

Πτυχιακή Εργασία

Ανάπτυξη της παραμετρικής γραφικής διεπαφής χρήστη της αστρονομικής υπηρεσίας Remote Interface for Science Analysis

The Remote Interface for Science Analysis (RISA) parameter GUI development

Σπουδάστρια: Ειρήνη Κομνηνού
ΣΤΕΦ - Τμ. Ηλεκτρολογίας
Τεχνολογικό Εκπαιδευτικό Ιδρυμα Κρήτης

Επιβλέπων καθηγητής: Δρ. Ευστράτιος Γεωργίου
ΣΤΕΦ - Τμ. Ηλεκτρολογίας
Τεχνολογικό Εκπαιδευτικό Ιδρυμα Κρήτης



Τεχνολογικό Εκπαιδευτικό Ιδρυμα Κρήτης (ΤΕΙΚ)

Μάρτιος 2010

CONTENTS

I INTRODUCTION	v
1 INTRODUCTION	1
1.1 Αντί προλόγου	1
1.2 General Introduction	4
1.3 Introduction to XMM-Newton	6
1.3.1 X-ray telescope optics	7
1.3.2 X-ray cameras	7
1.3.3 X-ray high resolution spectroscopy	8
1.3.4 The Optical Monitor (OM)	9
II XMM-NEWTON DATA ANALYSIS	11
2 SAS AND RISA	13
2.1 A brief introduction to the Science Analysis System (SAS)	13
2.2 RISA: The new, versatile approach to data reduction and analysis	15
2.2.1 RISA advantages	17
2.2.2 Limitations of the current RISA implementation	19
III THESIS PROJECT	21
3 THE STUDENT'S THESIS PROJECT	23
3.1 The new RISA parameter GUI	23
3.1.1 The new parameter GUI structure	23
3.1.2 The RISA task tree	28
3.1.3 Producing windows dynamically	30
3.1.4 Improving the source code	33
3.1.5 Architecture visualization	39
3.2 Results and future work	41
3.2.1 Scientific importance	41
3.2.2 Hardware technology used	42
3.2.3 Software technology used	42
3.2.4 Remaining work	44
3.2.5 Lessons learned	45
Bibliography	46

CONTENTS

IV APPENDICES	49
A SAS Tasks XML file	51
B TaskParser.java	55
C TaskTree.java	59
D Params.java	65
E Tasks.java	69
F LYTReader.java	73
G BuildPanel class	101
H BuildParameter class	105
I BuildTab class	109
J BuildTabPage class	111
K BuildComboBox class	115
L BuildCheckBox class	117
M LayoutPane interface	119
N PassWindowReader class	121
O Evselect .par file	131
P Evselect .lyt file	143

Part I
INTRODUCTION

1

INTRODUCTION

1.1 αντί προλόγου

Το Σεπτέμβριο του 2008, η γράφουσα ξεκίνησε την πρακτική της άσκηση στο Ευρωπαϊκό Κέντρο Αστρονομίας και Διαστημικής (European Space Astronomy Centre - ESAC) της Ευρωπαϊκής Υπηρεσίας Διαστήματος (European Space Agency - ESA) το οποίο εδρεύει στην Μαδρίτη. Παράλληλα αναπτύχθηκε ένα νέο λογισμικό στα πλαίσια της πτυχιακής εργασίας, υπό την εποπτεία του Δρ. Ευστράτιου Γεωργίου (ΤΕΙ Κρήτης). Για την εκπόνηση του τεχνικού μέρους της πτυχιακής εργασίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού Θαα¹, η οποία μπορεί να θεωρηθεί πιθανότατα ως η πιο διαδεδομένη αντικειμενοστραφής γλώσσα προγραμματισμού σήμερα. Κατά τη διάρκεια της βμηνής περιόδου πρακτικής άσκησης αποκομίσθηκαν νέες γνώσεις και δεξιότητες στον τομέα επιστήμης και τεχνολογίας υπολογιστών κυρίως, με την καθοδήγηση των Δρ. Κάρολος Γκαμπριέλ (ESAC) και Αιτόρο Ιμπάρρα (ESAC).

Έχοντας υπόβαθρο Ηλεκτρολόγου μηχανικού με έμφαση στην μελέτη και ανάπτυξη υλικού και με εμπειρία σε δομημένο προγραμματισμό (γλώσσα C) μόνο, η διαδικασία εκμάθησης και χρήσης μιας αντικειμενοστραφούς γλώσσας προγραμματισμού όπως η Java αποτέλεσε πρόκληση δεδομένου του περιορισμένου χρονικού περιθωρίου που προσφέρει η περίοδος πρακτικής άσκησης.

Οι μέλοις της ομάδας του Επιστημονικού Κέντρου Επιχειρίσεων (Science Operations Centre) του δορυφόρου XMM-Newton,

1 Η Θαα είναι γλώσσα προγραμματισμού που αναπτύχθηκε το 1995 από την εταιρία Sun Microsystems σαν δομικό στοιχείο της πλατφόρμας Θαα. Η σύνταξη της γλώσσας προέρχεται από την σύνταξη των γλωσσών “·”+“·” με αρκετές απλοποιήσεις και λιγότερες λειτουργίες χαμηλού επιπέδου (λογ λεελ). Οι εφαρμογές Θαα μεταφράζονται μέσω του σομπιλερ σε σύνολα εντολών ονόματι ζλασς και μπορούν να τρέξουν σε οποιοδήποτε Θαα ίρτυαλ Μαζηνε (ΘΜ)

INTRODUCTION

οι κύριες υποχρεώσεις μου περιελάμβαναν μεταξύ άλλων την μελέτη και ανάπτυξη μιας νέας παραμετρικής γραφικής διεπαφής χρήστη για την αστρονομική υπηρεσία **Remote Interface for Science Analysis (RISA)**, και αποτελεί την πτυχιακή μου εργασία. Η νέα γραφική διεπαφή χρήστη συμβάλλει στην εύκολη διαχείριση των δυνατοτήτων του RISA από έμπειρους και άπειρους χρήστες, κάθιστόντας την εφαρμογή πιο ελκυστική και προσιτή σε ένα ευρύ φάσμα επιστημόνων του χώρου.

Η υπηρεσία RISA που άρχισε να αναπτύσσεται από την ESA πρόσφατα, αποτελείται από ένα σύστημα *client/server* που έχει την δυνατότητα να προσφέρει όλες τις λειτουργίες του ήδη υπάρχοντος αλλά πεπαλαιωμένου **Science Analysis Software (SAS)**² καθιστόντας εφικτή την χρησιμοποίηση όλων των δυνατοτήτων του SAS χωρίς να απαιτείται τοπική εγκατάσταση του προγράμματος.

Το RISA έχει πολύ μικρές απαιτήσεις πόρων συστήματος και μπορεί να δουλέψει με σύνδεση διαδικτύου οποιασδήποτε ταχύτητας. Λαμβάνοντας υπόψιν τους πόρους³ που απαιτεί το SAS σε σύγκριση με τις απαιτήσεις του RISA, η ESA θεωρεί επιβεβλημένη την αντικατάσταση του SAS από το RISA.

Οι πηγές που χρησιμοποιήθηκαν για την ανάπτυξη της παρούσας εργασίας ήταν αρχικά μερικές επιστημονικές εργασίες σαν εισαγωγή, με κύρια πηγή πληροφοριών το διαδίκτυο. Η διαδικασία έρευνας μέσω διαδικτύου υπήρξε πολύ διδακτική, οδήγησε την γράφουσα στην ανάπτυξη ερευνητικής και ανεξαρτητης σκέψης

² Το SAS αναπτύχθηκε σαν συνοδευτικό λογισμικό της αποστολής XMM-Newton. Αποτελεί το λογισμικό απλοποίησης δεδομένων της αποστολής και σχεδιάστηκε με σκοπό να μετατρέπει τα ανεπεξέργαστα δεδομένα των παρατηρήσεων του δορυφόρου σε ορθά βαθμονομημένα και επεξεργασμένα επιστημονικά δεδομένα έτοιμα προς ανάλυση. Για παράδειγμα το SAS προσφέρει στους αστρονόμους εικόνες, φασματικές αναλύσεις, καμπύλες φωτός κ.ά. Χρησιμοποιείται από ολούς τους αστρονόμους που πραγματεύονται με παρατηρήσεις ακτίνων X και επιλέγουν να χρησιμοποιήσουν τα δεδομένα του δορυφόρου XMM-Newton για την έρευνά τους

³ Αφένός απαιτείται χρονοβόρα και πολύπλοκη διαδικασία εγκατάστασης του κυρίως προγράμματος και των συνοδευτικών υποπρογραμμάτων από τον χρήστη. Επίσης απαιτείται μεγάλη υπολογιστική ισχύ κατά τις διαδικασίες ανάλυσης των παρατηρήσεων του δορυφορίου. Αφέταίρου απαιτούνται αυξημένοι πόροι για την ανάπτυξη και συντήρηση της εφαρμογής SAS με αποτέλεσμα την αύξηση του συνολικού προϋπολογισμού της αποστολής XMM-Newton

1.1 αντί προλόγου

και μεθοδολογίας εργασίας.

Ο σκοπός της παρούσας εργασίας είναι να παρουσιαστεί η τεχνολογία, η μεθοδολογία (συμπεριλαμβανομένου του πηγαίου κώδικα) και η φιλοσοφία που χρησιμοποιήθηκε για την ανάπτυξη της νέας γραφικής διεπαφής του RISA. Κάθε κομμάτι πηγαίου κώδικα περιέχει σχόλια με λεπτομερείς επεξηγήσεις τα οποία καθοδηγούν τον μέσο αναγνώστη που έχει βασικό υπόβαθρο θεωρίας προγραμματισμού. Γιαυτό το λόγο, η εργασία χωρίζεται σε δύο μέρη: Το κυρίως κείμενο που δίδει δομικές πληροφορίες και επεξήγηση της μεθοδολογίας που ακολουθήθηκε και ο πηγαίος κώδικας –μαζί με τα συνοδευτικά σχόλια– που βρίσκεται στα παραρτήματα.

Προτείνεται να γίνει παράλληλη ανάγνωση του κυρίως κειμένου και των παραρτημάτων που αντιστοιχούν σε κάθε κεφάλαιο.

INTRODUCTION

1.2 GENERAL INTRODUCTION

In September 2008, the author of this thesis started working as a trainee at the European Space Astronomy Centre, the European Space Agency centre for astronomy and space science. During the six month placement period and under the tutorship of Dr. Carlos Gabriel (ESAC) and Aitor Ibarra (ESAC), new skills and aspects of computer technology were mastered. Furthermore a new piece of software was developed serving as the author's thesis project, under the tutorship of Dr. Efstratios Georgiou (TEIC). *Java*⁴ –probably the most popular object oriented language as we speak– was used for the development of the source code of this project.

Having an Electrical engineering background with emphasis on hardware analysis & design and experience in imperative programming (C programming language) but no experience in any object oriented programming languages, learning and manipulating a high level object oriented language like *Java* was a challenge, given the tight time frame.

Working as part of the XMM-Newton Science Operations Centre software development team, the author's main responsibilities included developing a new parameter Graphical User Interface (GUI) for the Remote Interface for Science Analysis (RISA) astrophysical service as her thesis project, making RISA practical and easy to use, thus more appealing to the end user. RISA is a client/server interface offering all the functionalities available in Science Analysis System⁵ (SAS), making it possible for anyone to uti-

⁴ Java is a programming language originally developed at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture.

⁵ SAS was developed as part of the XMM-Newton's mission infrastructure. It is XMM-Newton's data reduction software designed to convert raw data to scientifically useful products, like calibrated event lists, images, spectra, lightcurves, source lists, etc. It's used by all X-ray astronomers who need to utilize XMM-Newton data for scientific research and analysis.

lize all SAS functions remotely without the need of a local installation. RISA requires minimal processing power and other computer resources (RAM etc) and will work with any type of internet connection. Given the very low resource requirements of RISA and the fact that developing and maintaining SAS is resource consuming⁶, RISA is scheduled to substitute SAS soon. This thesis project will be implemented to RISA on the following version release.

A few papers served as references initially but the real knowledge derived almost entirely from the internet, with the ESAC tutors acting more like advisors rather than reference points. This approach was helpful and didactic, leading to a new more self sufficient way of thinking and working even though the learning curve became quite steep and the development process more time consuming. It also proved that the internet is a vast source of information, providing high level information to people ranging from hobbyists, to students and professionals.

This document aims in describing the technology, methodology, technical aspects and work philosophy applied that led to the successful development of the new RISA parameter GUI, i.e this thesis project. The entire source code can be found in the Appendices on the last chapter of the thesis. Each *Java* class contains explanation notes wherever needed, able to guide any programmer with basic *Java* syntax knowledge. Thus the main text contains program structure information and methodology explanation rather than detailed explanations of the source code itself. Parallel reading of the thesis text along with the source code –included in the Appendices– is recommended by the author.

⁶ From the end user's point of view: Performing a local installation of SAS is time consuming, running SAS is an intensive task requiring many computer resources. From the developers' point of view: SAS requires manpower, appropriate hardware and a corresponding budget, thus raising the overall XMM-Newton mission maintenance costs

INTRODUCTION

1.3 INTRODUCTION TO XMM-NEWTON

XMM-Newton (Figure 1.1) is the most sensitive X-ray satellite ever built, and with a total length of 10 meters, the largest satellite ever launched by ESA. It has been operating as an open observatory since the beginning of 2000, providing X-ray scientific data through three X-ray imaging cameras and two X-ray spectrometers, as well as visible and UV images through an optical telescope. The large amount of data collected by XMM-Newton is due to its unprecedented effective area in the X-ray domain in combination with the simultaneous operation of all its instruments. The latter is not only of quantitative, but also of eminent qualitative importance given the intrinsically variable nature of most of the phenomena associated with cosmic X-ray generation.

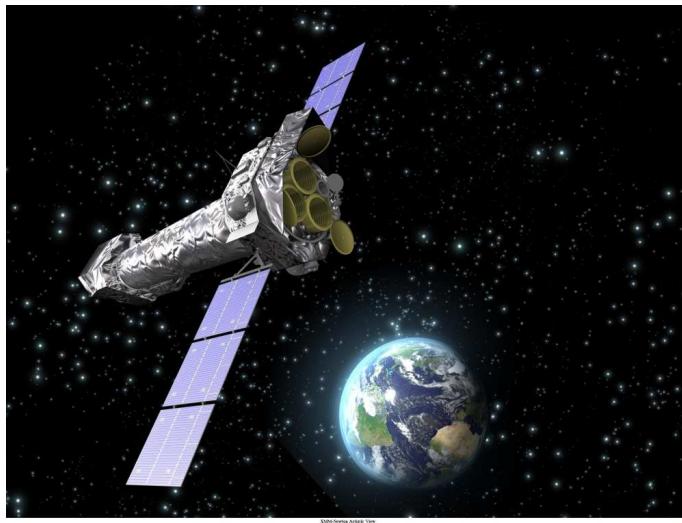


Figure 1.1: XMM-Newton - Artist's impression

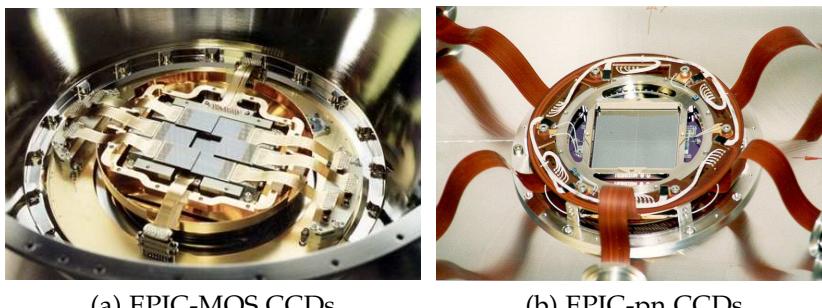
1.3 INTRODUCTION TO XMM-NEWTON

1.3.1 X-ray telescope optics

The XMM-Newton X-ray telescope composed of three barrel-shaped mirror modules, each with 58 "Wolter-II" type concentric gold-coated electroformed nickel mirror shells, covering through grazing incidence reflection the spectral range of [0.1-12]keV. The large total collecting area of 4300 cm² at 1.5 keV is complemented by moderate angular resolution of ca. 5" Full Width Half Maximum (FWHM) (14" Half Energy Width (HEW)).

1.3.2 X-ray cameras

XMM-Newton carries three X-ray imaging CCD cameras: the European Photon Imaging Cameras (EPIC) each of them in the focal plane of one of the X-ray telescopes. The cameras are of two different types, one of them using a new type of CCD (pn) (Figure 1.2b) especially developed for X-ray missions. Since all three work in single-photon register mode, and can register also energy and arrival time of each incoming X-ray photon, they provide simultaneously moderate spectroscopic and timing capabilities. Different operating modes can be used to optimize the observations according to the brightness of the target and the purpose of the observation.



(a) EPIC-MOS CCDs

(b) EPIC-pn CCDs

Figure 1.2: EPIC instruments

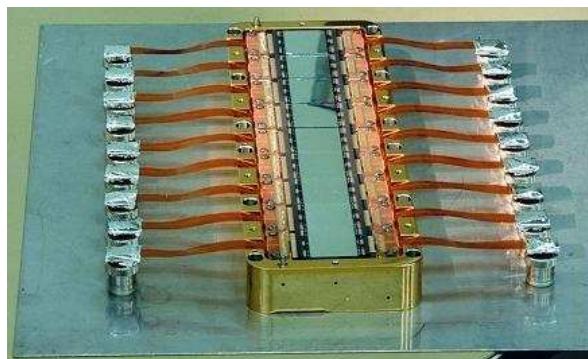
INTRODUCTION

1.3.3 X-ray high resolution spectroscopy

Behind two of the three X-ray telescopes, diffraction gratings (Figure 1.3) intercept around 50% of the incoming light, dispersing by reflection onto the CCDs (Figure 1.4) of the Reflection Grating Spectrometer (RGS) with a spectral resolution ($\lambda / \Delta \lambda$) of around 200 in first order dispersion in the soft X-ray domain ([0.3-2.4] keV).



Figure 1.3: The Reflection Grating Array (RGA)



XMM-Newton Focal Plane Camera (RFC)
Image courtesy of EKV Ltd., SRON, Paul
Scherzer Institute European Space Agency

Figure 1.4: The RGS Focal Plane Camera (RFC)

1.3 INTRODUCTION TO XMM-NEWTON

1.3.4 The Optical Monitor (OM)

Co-aligned with the X-ray telescopes, the OM (Figure 1.5) gives XMM-Newton a multi-wavelength capability, operating in the range [1600-6600] Å. The OM camera can work also in photon counting mode, providing therefore time-resolved information in addition to visible and UV images. It offers a FOV of 17' and a limiting sensitivity (5σ) of 20.7 mag for an integration time of 1000 seconds. Additional optical and UV grisms provide XMM-Newton with a moderate spectral resolution, also in this range.

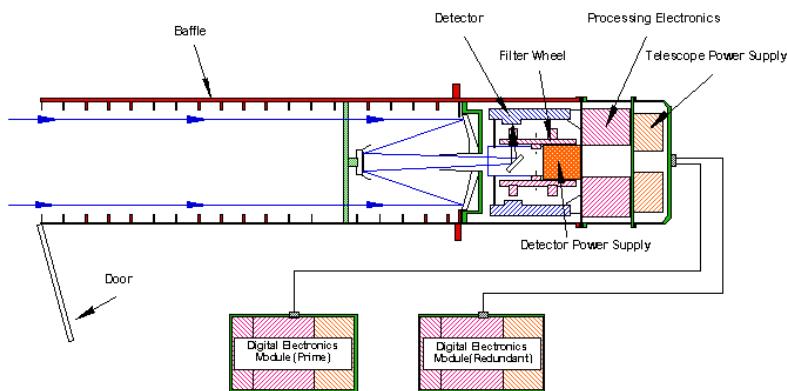


Figure 1.5: A schematic of the Optical Monitor

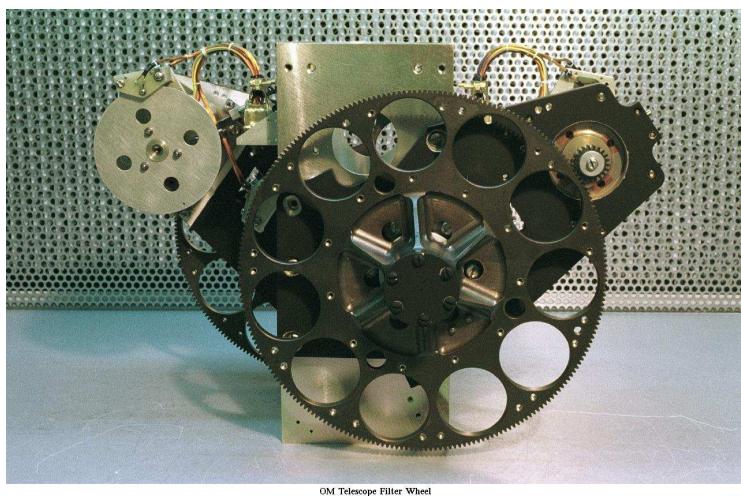


Figure 1.6: The OM Telescope Filter Wheel

Part II

XMM-NEWTON DATA ANALYSIS

2

SAS AND RISA

2.1 A BRIEF INTRODUCTION TO THE SCIENCE ANALYSIS SYSTEM (SAS)

The Science Analysis System (SAS) is the XMM-Newton data reduction software able to convert raw data to scientifically useful products, like calibrated event lists, images, spectra, lightcurves, source lists, etc. It is used regularly by almost 2000 astronomers and has helped to do the scientific analysis by the vast majority of more than 2200 refereed scientific publications based on XMM-Newton data so far.

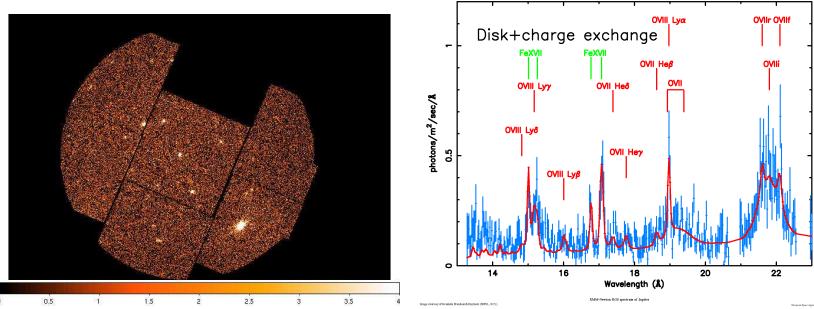
Mainly coded in C++ and Fortran 90, SAS is a complex software application dealing with data from diverse X-ray and optical / UV instruments, designed to run on a stand-alone computer, and heavily dependent on third party software and libraries.

To process XMM-Newton raw data using SAS and convert them to scientifically useful products, the end user has to perform the following steps:

- Download and install SAS and calibration Database.
- Download and install third party software (perl, cfitsio, ds9, grace...).
- Search and download the data from the XMM-Newton archive. (<http://xmm.esac.esa.int/xsa/>)
- Process the data locally, create clean event files.
- Derive images, spectra, light curves and source lists. (See figures below)

SAS AND RISA

Figures 2.1a, 2.1b and 2.2 seen below contain some of the SAS products:



(a) XMM-Newton image taken from the MOS1 camera – Credits: Nuria Fonseca and ESA
 (b) XMM-Newton RGS spectrum of Jupiter – Credits: G.Branduardi-Raymont and ESA

Figure 2.1: SAS products

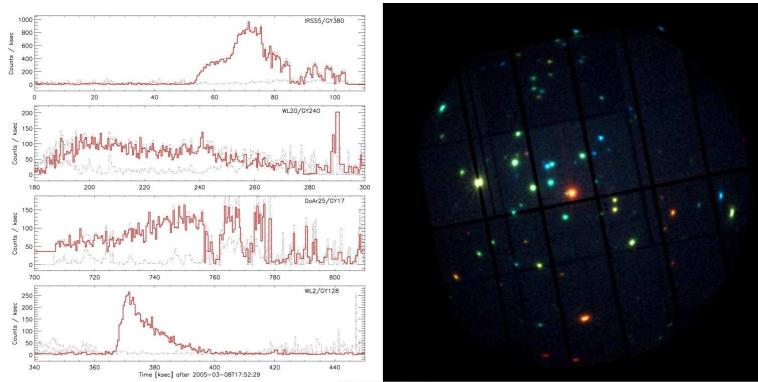


Figure 2.2: XMM-Newton EPIC-pn Lightcurves and Image of YSOs in the ρ Oph Star Forming Region – Credits: S.Sciortino and ESA

SAS binaries are distributed for several OSs and flavors, which have to be produced, forced by the high dependency from internal libraries. This requires more manpower, increased cost and time between SAS releases. A brief introduction to the new SAS approach is following.

2.2 RISA: THE NEW, VERSATILE APPROACH TO DATA REDUCTION AND ANALYSIS

2.2 RISA: THE NEW, VERSATILE APPROACH TO DATA REDUCTION AND ANALYSIS

The Remote Interface for Science Analysis (RISA) is an interface, under development, to all SAS functionalities through a Client / Server application running SAS workflows in a Grid Architecture. It is developed in Java and makes use of AJAX and SOAP technologies for web interfacing as well as message exchange between the Server and the Client applications. GridWay¹ is used as a Grid meta-scheduler². You can see the RISA flow of work in Figure 2.3.

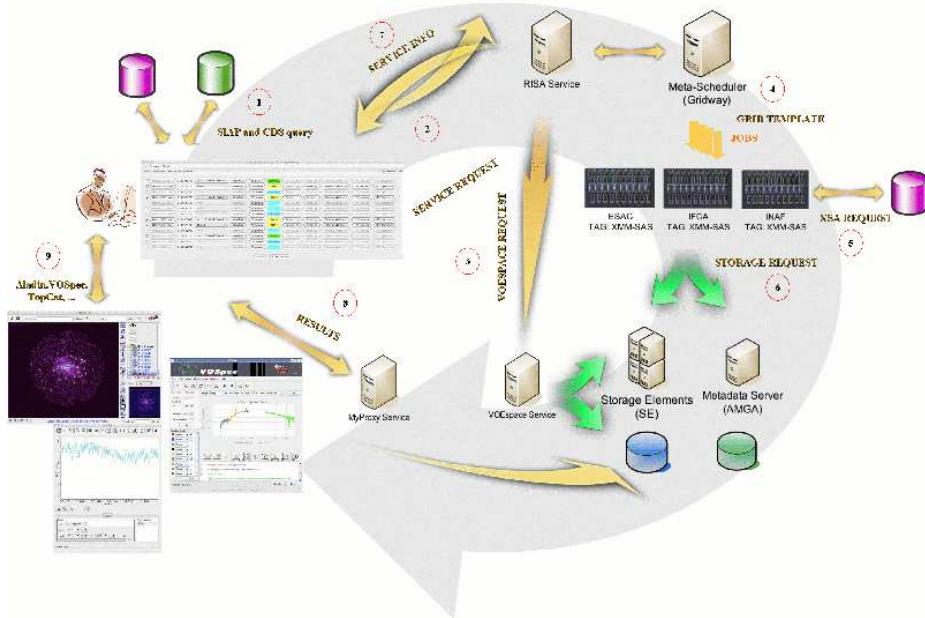


Figure 2.3: RISA flow of work. Each of the steps is explained right below

¹ GridWay (www.gridway.org) is an open source meta-scheduling technology that enables large-scale, secure, reliable and efficient sharing of computing resources (clusters, computing farms, servers, supercomputers...), managed by different Distributed Resource Management (DRM) systems, such as Sun Grid Engine, Condor, PBS, Load Sharing Facility..., within a single organization (enterprise grid) or scattered across several administrative domains (partner or supply-chain grid).

² Meta-scheduling or Super scheduling is a computer software technique of optimizing computational workloads by combining an organization's multiple Distributed Resource Managers into a single aggregated view, allowing batch jobs to be directed to the best location for execution.

The following points explain in detail the RISA flow of work:

- 1 SIAP query to retrieve information about the XMM-Newton observation (Observation Start Time, Exposure Time, Instrument Modes). CDS query to resolve astronomical names.
- 2 Service request. The client serializes the RISA input workflow together with the parameters corresponding to each workflow.
- 3 The RISA service contacts the VOEspace service to create data node(s), where the results will be stored once the observation(s) are reduced on the Grid. The VOEspace service creates the data node(s) and returns a set of GridFTP URL (endpoints) which RISA can use for uploading the results.
- 4 RISA sends the job(s) to the Grid through DRMAA OGF standard (Grid-Way implementation) specifying the output file(s) to be stored at the end-point(s) that VOEspace returned in the previous step.
- 5 Each node in the GRID makes a requests to the XMM-Newton Science Archive (XSA) to retrieve the XMM-Newton ODF (Observation Data File) data set, corresponding to a given observation ID, using the AIOClient (ESAC archive team tool to automatically download XMM-Newton data). application.
- 6 Once the data processing has finished, the results are taken to the Storage Element (GridFTP endpoints returned by VOEspace) automatically by GridWay. The RISA service then knows that the observations have been processed and informs the user (job DONE).
- 7 The RISA service sends information to the Client about the job status.
- 8 The user clicks a button for downloading results which connects to MyProxy service to download temporary credentials (needed to interact with the Grid Storage Elements) and performs the copy using GridFTP.
- 9 The results can be displayed with Aladin / VOSpec / TopCat / ... opened through PLASTIC interface.

2.2 RISA: THE NEW, VERSATILE APPROACH TO DATA REDUCTION AND ANALYSIS

Due to this approach, RISA is able to:

- Avoid or at least reduce the Operating System libraries dependencies and compilers evolution³ -- Virtualization (SAS on demand)⁴.
- Reduce the installation of third party software on the user's side.
- Easily increase the computing resources -- Scalability⁵.
- Make use of Virtual Organization infrastructures in a Grid Computing environment.
- Leave a legacy application for a long scientific mission⁶.

2.2.1 *RISA advantages*

The RISA webservice allows the remote execution of a SAS Workflow in a Grid environment. This design has the following advantages:

- No need for a local SAS installation or local maintenance of calibration files or third party software is required.
- Easy and Fast way of accessing large amounts of data (e.g. no download required).
- User's platform independent SAS execution.
- No need for large number of SAS integration platforms.
- Easy access to large computing resources thanks to the Grid technology (virtual organizations).

³ Maintaining SAS will be much easier. For the time being there are 16 SAS versions produced, corresponding to 16 OSs, every time a new SAS release is made. With RISA, only one SAS release needs to be produced, corresponding to the server OS.

⁴ With the use of system virtual machines, the end user is able to install and use the available SAS release under the user's computer OS.

⁵ RISA can be run on one or more GRIDs thus increasing the processing power if necessary.

⁶ RISA requires minimal maintenance and will be able to produce scientific products from the XMM-Newton data even many years after the end of the mission.

SAS AND RISA

Future developments and improvements of this design will focus on the following points:

- Standardization of SAS Workflows⁷.
- Access to all SAS functionalities (SAS parameters)⁸.
- SAS error handling.
- General VO type access to SAS routines.
- Storage data products: MySpace, VOSpace⁹.
- On the fly generator of Virtual Observatory compliant products.

⁷ A SAS workflow is a chain of calls to several tasks performing a certain type of data reduction.

⁸ RISA provides a categorized view of all SAS functionalities as well as a graphical interface for each SAS task's parameters, enabling the end user to easily and fully parametrise the final SAS product according to the user's needs.

⁹ VOSpace is an International Virtual Observatory Alliance (IVOA) standard describing the interfaces to implement for a Virtual Observatory (VO) distributed storage service. It is just the visible part of the implementation which specifies how VO agents and applications can store and exchange data in a standard way. It can be considered as an access point, through a SOAP Web Service, to a distributed storage network.

2.2.2 *Limitations of the current RISA implementation*

Even though RISA is a pioneering concept providing the astrophysicists and astronomers with a very handy SAS implementation via the web, the initial RISA parameter GUI (Figure 2.4) was less user friendly due to its very basic design.

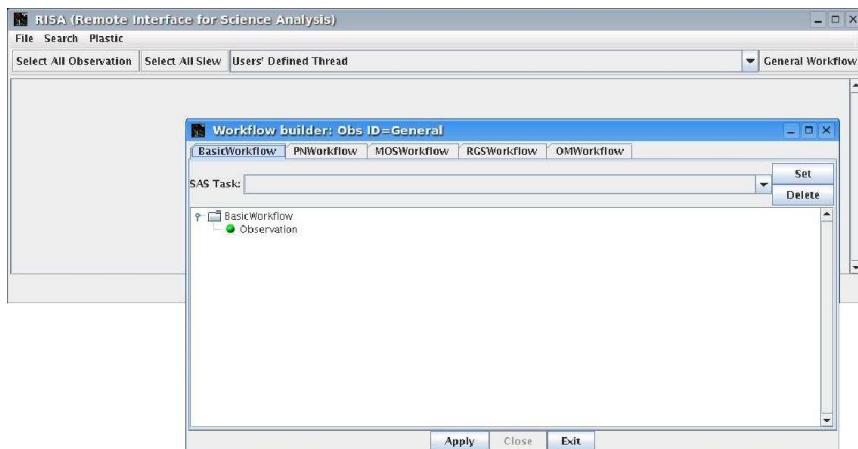


Figure 2.4: The old RISA parameter GUI

The need for a user friendly parameter GUI (Graphical User Interface), similar to the one implemented in SAS was recognized from the very beginning. It should be able to select SAS tasks and set or change the default values of the task parameters using a graphical interface similar to the one encountered in SAS which is very familiar to the X-ray scientists that deal with data analysis.

The following chapter will introduce the new RISA parameter GUI concept which constitutes this thesis, developed during the period 09/2008 - 03/2009.

Part III
THESIS PROJECT

3

THE STUDENT'S THESIS PROJECT

3.1 THE NEW RISA PARAMETER GUI

The aim of this thesis project was to improve the functionality and usability of RISA by implementing a new parameter Graphical User Interface. Since SAS is a well known and widely used application for X-ray astronomy, the SAS GUI served as a guideline and therefore greatly influenced the overall RISA GUI philosophy.

In general, producing a simple yet practical GUI simplifies the usage of a program, thus making it more appealing to all possible end users. When it comes to scientific applications, the existence of a clear, simple, practical GUI is essential since the vast majority of scientists counts reliability and practicality over appearance.

3.1.1 *The new parameter GUI structure*

At first, a search for new graphical interface concepts was done. The pros and cons were discussed and the final decision on how the new RISA interface would look was taken. The new RISA parameter GUI consists of a simple tree containing nodes and leafs. Nodes are categorized based on the respective XMM-Newton satellite instruments -for example EPIC, MOS, PN, RGS- or by the scope of the task -i.e GENERAL and THREAD- and leafs contain all the tasks that are applicable for the respective instrument data as seen in Figure 3.1. Whereas the original SAS GUI (Figure 3.2), which is rarely used due to its low practicality, contains all the tasks listed uncategorized.

THE STUDENT'S THESIS PROJECT

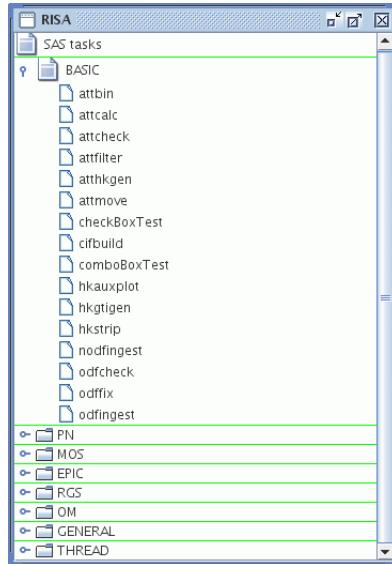


Figure 3.1: The new RISA task selector

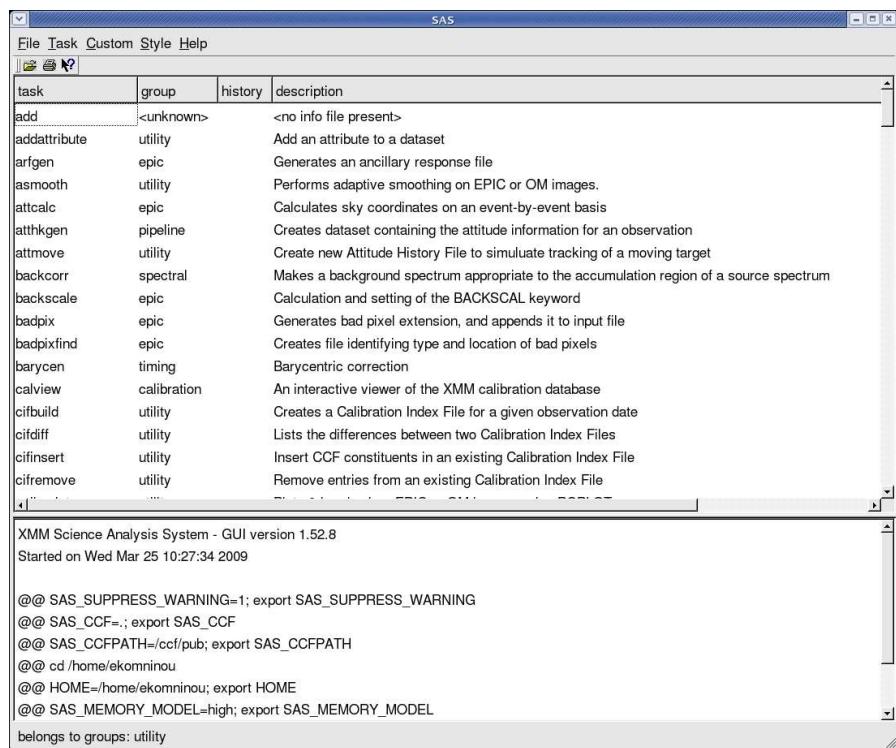


Figure 3.2: The SAS task selector

Whenever the user clicks on any task of the RISA task selector, a new window pops up containing all the parameters of this task. It is easily noticeable that the new RISA parameter GUI has a very strong resemblance to the classical

3.1 THE NEW RISA PARAMETER GUI

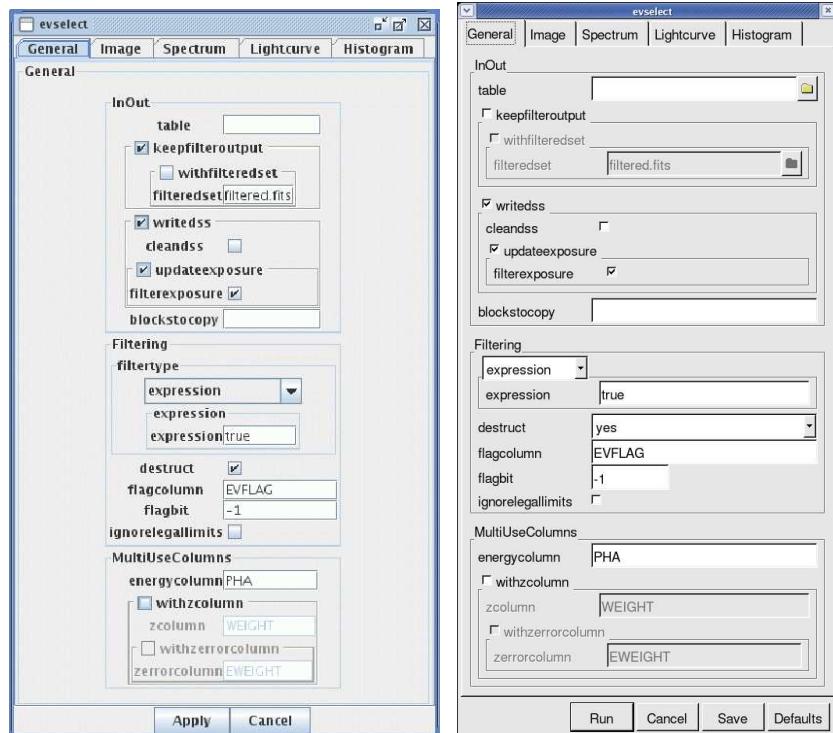
SAS GUI -which is considered a standard in the X-ray astronomical community- and all parameters are categorised depending on their function. Whereas the old RISA parameter GUI contains all the parameters listed uncategorized. In the next page, you can see a SAS task example and its RISA counterpart. The *evselect*¹ task window of the new RISA parameter GUI compared to the SAS one (Figure 3.3) and the evselect task window of the old RISA parameter GUI (Figure 3.4).

¹ *Evselect* is of central significance among the suite of individual SAS tasks. It serves two complementary purposes:

- 1 Filter event list data controlled by user-specified selection criteria. [Calibrated event list files consist of calibrated ODF (Observation Data File) sets, which are used as input for further detailed scientific analysis. ODFs are raw observation data produced by the respective XMM-Newton scientific instruments. Calibration means to apply all the transformations, which allow to convert instrument quantities into physical ones].
- 2 Extract images, spectra, and time series from the filtered event list obtained in the first step. The filtered list is selectable for output as well.

The availability of both features within a single task allows the end user to generate useful products from an event list (e.g. create a rates file in the energy band 0.5 - 3 keV from the X-ray source lying within a circle of 10 arcsec radius in the center of the FOV (Field Of View) in a convenient manner.

THE STUDENT'S THESIS PROJECT



(a) The **new** RISA evselect task window (b) The SAS evselect task window

Figure 3.3: RISA vs SAS task windows

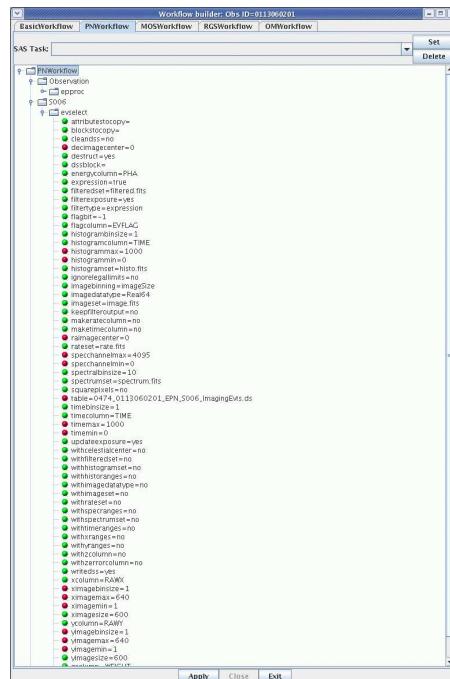


Figure 3.4: The **old** RISA evselect task window

3.1 THE NEW RISA PARAMETER GUI

The new RISA parameter GUI resemblance with the classical SAS one is easily noted in the screenshots above (Figure 3.3 (a) and (b)). The reason of designing a GUI identical to the original SAS one was simple: For the past 10 years this interface has served the researchers very well and it has become a well known standard among astronomers / astrophysicists dealing with X-ray astronomy like mentioned above. With simplicity and practicality in mind, there was no reason for altering or completely changing the very satisfactory SAS GUI for something more sophisticated or modern in appearance.

3.1.2 The RISA task tree

The SAS task data is included in an XML file containing all the available *workflows* as well as the corresponding *tasks* of each workflow (Figure 3.5).

```
<?xml version="1.0" encoding="UTF-8"?>
<BODY>
<Workflow value="BASIC">
    <Task level="Observation">odfingest</Task>
    <Task level="Observation">nodfingest</Task>
    <Task level="Observation">cifbuild</Task>
    <Task level="Observation">odffix</Task>
    <Task level="Observation">attmove</Task>
    <Task level="Observation">atthkgen</Task>
    <Task level="Observation">attbin</Task>
    <Task level="Observation">attfilter</Task>
    <Task level="Observation">hkgtigen</Task>
    <Task level="Observation">attcalc</Task>
    <Task level="Observation">odfcheck</Task>
    <Task level="Observation">attcheck</Task>
    <Task level="Observation">hkstrip</Task>
    <Task level="Observation">hkauxplot</Task>
</Workflow>
<Workflow value="PN">
    <Task level="Observation">epchain</Task>
    <Task level="Observation">epreject</Task>
    <Task level="Observation">eposcorr</Task>
```

Figure 3.5: SAS task data in XML format (see *Appendix A* for full XML code)

Therefore, a *parser* class called [TaskParser](#)² had to be developed that is able to read the XML data and pass it to the graphical interface, using tools from the JDOM³ API containing the well known SAX⁴ among others.

Also, a *Tree interface* class was developed, namely [TaskTree](#)⁵. This interface is able to read the SAS XML parsed data and pass them as individual or grouped nodes in a tree using the *JTree*⁶ Java class. This *Tree interface* is flexible enough since it's able to dynamically change the tree

² see *Appendix B* for full Java code

³ <http://www.jdom.org/> Application Programming Interface (API) – JDOM provides a complete, Java-based solution for accessing, manipulating, and outputting XML data from Java code.

⁴ <http://www.saxproject.org/> – SAX is the Simple API for XML, originally a Java-only API. SAX was the first widely adopted API for XML in Java, and is a “de facto” standard.

⁵ see *Appendix C* for full Java code

⁶ A control that displays a set of hierarchical data as an outline

contents whenever the actual XML file needs to be altered for some reason (e.g new SAS tasks released, obsolete SAS tasks removed by the SAS developing team).

The combination of parsing the SAS tasks XML file with passing the parsed data to the *Tree* interface resulted in a new simple yet effective *task selector* containing all SAS tasks categorized based on the respective XMM-Newton instrument and in alphabetical order (instruments and tasks starting with *a* are first, instruments and tasks starting with *z* are last).

After constructing the new RISA *task selector* interface, the second thing that had to be taken care of was to construct a *parser* which would be able to read the SAS graphical interface files and *dynamically*⁷ produce *modal windows*⁸ similar to the SAS ones.

It should be noted that SAS offers approximately 170 tasks⁹ (and rising) for all kinds of astrophysical data analysis, i.e 170+ different windows containing a combination of 6 types of widgets (also known as controls) depending on the task. Producing more than 170 different static GUIs is out of the question. It can be a very time and resource consuming procedure that's not sophisticated at all thus leaving us with the choice of writing code that will be able to produce such a variety of windows *dynamically*. This was a big challenge for a novice *Java* programmer –it can be challenging even for an experienced programmer– that

⁷ Each SAS GUI is composed dynamically based on two files' contents: the respective *.lyt* file contents which define the final graphical interface output and the *.par* file contents which set the default values of all parameters of each SAS task. This is done via numerous loops that "decide" which widget should be produced and where should it be put depending on the content of the respective task. Both *.lyt* and *.par* files are heritage of SAS and are used for producing all the existing GUIs of SAS.

⁸ A modal window is a child window that requires the user to interact with it before they can return to operating the parent application, thus preventing the workflow on the application main window. Modal windows are often called heavy windows or modal dialogs because the window is often used to display a dialog box.

⁹ A list of all tasks can be found at <http://xmm.vilspa.esa.es/sas/6.5.0/doc/packageList.html>

the student had to face.

3.1.3 *Producing windows dynamically*

The SAS graphical interface is originally generated by parsing and graphically interpreting two types of documents using the *Qt toolkit*¹⁰ : *.lyt* and *.par* files. Each SAS task has a corresponding *.lyt* and *.par* file. Both file types were developed during the early '90s as part of the SAS graphical interface infrastructure. The first file type is used for setting the graphical interface of each SAS task window by indicating the right widgets and parameters that should be contained in each task and where they should be placed. The latter contains information regarding the default values of each parameter contained in the respective SAS task window. In SAS both files are read through a parser written in C++ and interpreted into windows containing a layout that's composed of a set of widgets –depending on the chosen task– where each widget includes the default parameters of this particular task.

Since RISA is been developed as a client / server application with many elements and complex procedures running in the background, the choice of *Java* as the main programming language was essential, therefore making it impossible to use the original C++ SAS parsers for generating the original SAS graphical interfaces. This means that a new approach had to be followed for reading the data included in the *.lyt* and *.par* files of each SAS task and generating a fully functional SAS task window using the RISA infrastructure. In order to be able to read the corresponding *.lyt* and *.par* files of each SAS task, new *Java* code had to be developed which would be able to:

¹⁰ <http://qt.nokia.com/> Qt (pronounced as the English word "cute") is a cross-platform application development framework, widely used for the development of GUI programs (in which case it is known as a widget toolkit), and also used for developing non-GUI programs such as console tools and servers. Qt is most notably used in Adobe Photoshop Album, Google Earth, KDE, Opera, OPIE, Skype, Qt Extended, VLC media player and VirtualBox.

3.1 THE NEW RISA PARAMETER GUI

```
Cards
  Page General
    Frame InOut
      Parameter table
      Enable keepfilteroutput
        Enable withfilteredset
          Parameter filteredset
        end
      end
      Enable writedss
        Parameter cleandss
        Enable updateexposure
          Parameter filterexposure
        end
      end
      Parameter blockstocopy
    end
  Frame Filtering
    Choice filtertype
      Page expression
        Parameter expression
      end
      Page dataSubspace
        Parameter dssblock
      end
    end
    Parameter destruct
    Parameter flagcolumn
    Parameter flagbit
    Parameter ignorelegallimits
  end
```

Figure 3.6: Sample code of a *.lyt* file (See *Appendix P* for full code)

- 1 read each *.par* file and temporarily store the included default values of each parameter.
- 2 read the corresponding *.lyt* file.
- 3 create a layout (=various widgets produced dynamically depending on the *.lyt* contents) which includes all the parameters of the corresponding SAS task.
- 4 create a window (=container) that will contain the layout
- 5 add the layout to the window.

At first, a baseline was constructed consisting of a dozen new *Java* classes approximately. Initially this new set of classes was able to produce a basic GUI, including at least a few widgets in each window.

THE STUDENT'S THESIS PROJECT

```
<FILE>
<CONFIG>

<PARAM id="table" type="table" mandatory="yes">
    <DESCRIPTION> Name of the table to be filtered </DESCRIPTION>
</PARAM>

<PARAM id="keepfilteroutput" type="bool" default="no">
    <DESCRIPTION> Keep output of filtering process? </DESCRIPTION>
    <CASE>
        <ITEM value="no" > </ITEM>
        <ITEM value="yes">
            <PARAM id="withfilteredset" type="bool" default="no">
                <DESCRIPTION> Create a filtered event list </DESCRIPTION>
                <CASE>
                    <ITEM value="no" > </ITEM>
                    <ITEM value="yes">
                        <PARAM id="filteredset" type="dataset" default="filtered.fits">
                            <DESCRIPTION> Name of file for filtered event list </DESCRIPTION>
                        </PARAM>
                    </ITEM>
                </CASE>
            </PARAM>
        </ITEM>
    </CASE>
</PARAM>
```

Figure 3.7: Sample code of a *.par* file (See Appendix O for full code) – note the XML resemblance

The *.par* files are being processed using two classes, namely **Param**¹¹ and **Tasks**¹². **Param** parses each *.par*'s file contents whereas **Tasks** sorts the parsed content of each *.par* file. The *.lyt* files are being parsed using the **LYTReader**¹³ class .

Lots of new additions, improvements and corrections had to be done on the initial set of classes, in order for the RISA tasks' GUI to be considered fully functional. Many widgets were malfunctioning, others overlapped each other, others weren't visible at all and others were situated off frame. Therefore it was crucial to ensure that the *.lyt* and *.par* files would be parsed correctly resulting in correctly formed windows, fully functional drop down menus, checkboxes and so on, the window constraints would be properly set in order to add each widget to the correct position, the methods and elements in the source code would be correctly set.

After going through the *Java* code, it was obvious that **LYTReader** was only able to read the vary basic *.lyt* files (developed only for testing the project functionality) correctly since they don't contain complex expressions like normal SAS *.lyt* files do. Therefore reforming the **LYTReader** was the first priority

¹¹ see Appendix D for full Java code

¹² see Appendix E for full Java code

¹³ see Appendix F for full Java code

3.1.4 Improving the source code

After rewriting some parts of the [LYTReader](#) code and setting the layout constraints correctly the overlapping widgets problem was solved, but the missing widgets issue as well as blank windows and so on still had to be taken care of since these issues were mainly caused by the way the *.lyt* files were parsed. Therefore the method of reading and parsing the *.lyt* files had to be revised and improved so that blank spaces and blank lines will be omitted, comments will be ignored and words between quotes will be considered as one string.

The main problem was that while reading each *.lyt* line by line, the line contents were extracted and buffered by using blank spaces as string split indicators even though the content of each line contained more complex strings like "*some text and whitespaces*" or comment (#) characters, tab spaces, whitespaces and so on. This forced us to search for a different way of splitting the components of each line before extraction. The regular expressions feature of Java was used which is powerful and flexible enough to ignore comments (#), tab spaces, whitespaces and read entire phrases as one string therefore conserving the original format of each task's GUI included in the respective *.lyt* file.

Like mentioned above, each *.lyt* (and the corresponding *.par*) file is being parsed using a fairly complex class ([LYTReader](#)) containing some of the powerful features of Java such as Lists, Iterators, BufferedReaders, FileReaders, Regular expressions, ActionListeners and many others.

The working philosophy behind this particular class is the following:

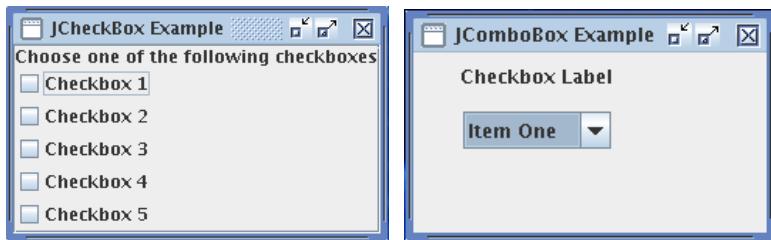
Firstly the *.lyt* file is read line by line using a FileReader. Once a line is buffered using a BufferedReader, the corresponding *.par* data is also buffered. If the line contains one or more keywords specifically set by the programmer, the corresponding method is triggered therefore producing the right widget frame.

The same line is read once more by the aforementioned reg-

ular expressions method, scanning for regular expressions that the programmer has set like “*Text and whitespaces*” for example. Once the regular expression has been located, it is corrected by subtracting the quotes or any other non necessary symbol(s) and is then passed along with the already processed *.lyt* and *.par* data to the corresponding widget where the full widget is finally added to the overall layout container.

If a line starts or contains a # (comment symbol), the regular expression method makes sure to omit that line and proceeds to the next one.

Like mentioned above, there are 6 different types of widgets available for the RISA parameter GUI depending on the functions that a SAS task can provide to the end user. Namely: [BuildPanel](#)¹⁴, [BuildParameter](#)¹⁵, [BuildTab](#)¹⁶, [BuildTabPage](#)¹⁷, [BuildComboBox](#)¹⁸, [BuildCheckBox](#)¹⁹ are the classes that produce panels, text fields and labels, tabs, tab pages, drop down menus and check boxes respectively. All 6 classes are heavily based on the Swing²⁰ API elements.



(a) An example of a Swing checkbox widget (b) An example of a Swing drop down menu widget

Figure 3.8: Swing widgets

¹⁴ see *Appendix G* for full java code

¹⁵ see *Appendix H* for full java code

¹⁶ see *Appendix I* for full java code

¹⁷ see *Appendix J* for full java code

¹⁸ see *Appendix K* for full java code

¹⁹ see *Appendix L* for full java code

²⁰ Swing is a widget toolkit for Java. It is part of Sun Microsystems' Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs. <http://java.sun.com/javase/6/docs/technotes/guides/swing/>

There are quite a few implementations of Swing available. *Jigloo*²¹ was eventually chosen, a user friendly Swing implementation containing a GUI that allows the programmer to construct any kind of static graphical interface by dragging and dropping widgets on a panel. When dragging and dropping widgets on the main panel, *Jigloo* generates the GUI *Java* code automatically therefore letting the programmer focus on coding the functionality behind the GUI rather than the GUI itself. Unfortunately this feature was not useful for the RISA parameter GUI project since the only standard feature used in all 170+ RISA implementations of the SAS tasks is a container where all widgets lay on, the rest of the features get to be produced dynamically like mentioned above. This added to the overall difficulty of the project but was very didactic in terms of the *Java* programming potentials and coding techniques.

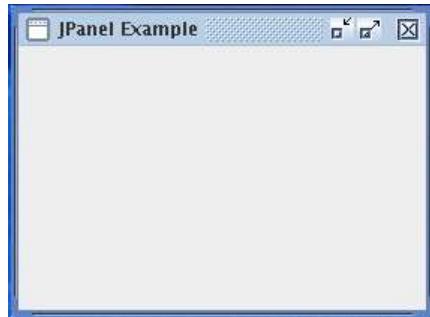


Figure 3.9: JPanel, a widget used as a "frame" surrounding other Swing controls

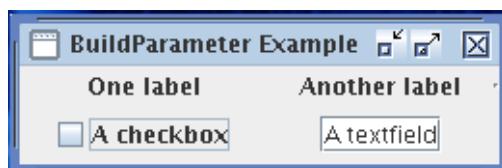


Figure 3.10: A BuildParameter widget - it consists of JLabels, JTextFields and JCheckboxes

Three of the aforementioned widget classes, `BuildParameter`, `BuildPanel` and `BuildTabPage` are included in the `LYTReader`'s methods. Every time a *.lyt* line is parsed, one of these three classes is called depending on the type of

²¹ www.cloudgarden.com/jigloo/

data the line contains. Information is passed to the class and the corresponding widget is formed. Each widget is placed in the layout, with respect to the constraints set, and when the last line of the *.lyt* file is read and the layout is complete it is then added to the main frame thus forming the task window.

Furthermore `BuildPanel` includes another three widget classes –`BuildParameter`, `BuildCheckBox` and `BuildComboBox`– in its methods. Everytime `BuildPanel` is called from a `LYTReader` method, it constructs the appropriate panel based on the data buffered from reading each *.lyt* line and passes it to the overall layout list containing the sum of widgets produced for assembling the GUI corresponding to the chosen task.

Also `BuildPanel`, `BuildCheckBox`, `BuildComboBox` and `BuildTab` implement an interface²² called `LayoutPane`²³.

A minor detail that had to be taken into consideration – which makes a noticeable difference though– was the scrollbar. Each window is able to be resized depending on the end user's needs, therefore a scrollbar has to be present once the window becomes smaller than the default value set by the programmer. This allows the window contents to be fully accessible no matter of the size of the window itself.

At first adding a scrollbar to the tasks' GUI wasn't feasible because of the Swing's API design constraints. In order for a scrollbar to be added to a window, the programmer has to set a valid viewport that will be accessible by the scrollbar. But the Swing API applies some constraints on each control. One of these constraints actually doesn't allow a container to be hosted inside another container.

Initially both the *window* and the *layout* were registered as

²² An interface in computer science is a set of named operations that can be invoked by clients. In the Java programming language, an interface is an abstract type that is used to specify an interface (in the generic sense of the term) that classes must implement. In programming languages, an abstract type is a type in a nominative type system which is declared by the programmer –where a nominative type system actually is a major class of type system, in which compatibility and equivalence of data types is determined by explicit declarations and/or the name of the types.

²³ See Appendix M for full java code

containers, thus making it impossible to incorporate the scrollbar to the *layout* and the *layout* to the *window*. For that reason the code had to be slightly altered in order to meet Swing's API criteria regarding scrollbars. The overall procedure was not overly complicated, still finding out the exact Swing constraints and making the proper code adjustments was a challenge.

Before the end of the traineeship period in March 2009 the RISA parameter GUI was fully functional, it displayed all the elements of each SAS task and only a few minor flaws had to be sorted out like correcting the spacing between the window frame and each layout etc in order for the GUI to be considered complete. Adjusting the layout constraints slightly –the GridBagConstraints²⁴ class is mainly used throughout the entire project– solved this issue.

The last step that had to be taken was to come up with a way to pass all the information included in each window (labels, textfields and their content, checkboxes and their boolean status, tabs and their contents etc) in an XML file which will contain all the untouched parameter values including the new values entered by the end user.

Passing information from the task window to an XML file was a slightly complicated procedure. A new class was developed called `PassWindowReader`²⁵ which is able to extract all useful data off of a task window and store them in an XML file with the help of another very helpful characteristic found in Java, the *Java logging API*²⁶. For

²⁴ The GridBagConstraints class specifies constraints for components that are laid out using the GridBagLayout class. <http://java.sun.com/j2se/1.5.0/docs/api/java.awt/GridBagConstraints.html>

²⁵ see Appendix N for full Java code

²⁶ The logging API is part of J2SE as of JDK 1.5, and it ships with the JDK. It is designed to let a Java program, servlet, applet, EJB, etc. produce messages of interest to end users, system administrators, field engineers, and software developers. Especially in production situations, where things can't be run in a debugger, or if doing so masks the problem that is occurring (because it is timing related, for example), such logs are frequently the greatest (and sometimes the only) source of information about a running program.

that reason a Logger²⁷ and a Handler²⁸ were utilized. The Logger was correctly set both in terms of syntax and logging level (amount of information to be logged) and the Handler was instructed to export all logged messages to a specific .xml file called *RISALog.xml*. Finally, a listener was introduced to each window's Apply button, which would pass all the information included in the current window to any tool that could handle and extract this XML data set (e.g the Logger).

What still needs to be finished is the output XML syntax of the Logger, containing the parameters of the task the end-user is using in a format compatible to the SAS XML parser. This can be done by tweaking the current XML logger used. The optimized XML logging system will be added to the new RISA configuration shortly.

-
- 27 A Logger object is used to log messages for a specific system or application component. Loggers are normally named, using a hierarchical dot-separated namespace. Logger names can be arbitrary strings, but they should normally be based on the package name or class name of the logged component, such as java.net or javax.swing. <http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/Logger.html>
- 28 A Handler object takes log messages from a Logger and exports them. It might for example, write them to a console or write them to a file, or send them to a network logging service, or forward them to an OS log and so on. <http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/Handler.html>

3.1.5 *Architecture visualization*

A good way to visualize a system's architectural blueprints in software engineering is to utilize the UML (Unified Modeling Language)²⁹. UML data models provide an overview of the system architecture and element interconnections (e.g class interconnections), which offers a very helpful insight of the overall system. For that reason a project UML diagram was constructed (Figure 3.11) which depicts the project class' interconnections. On this diagram you can see how classes depend on each other by calling one another while the code is executed. Also you can see which classes depend on the [LayoutPane](#) interface. That way you can get a good idea of the way all classes are linked with each other which gives a good overview of the project's source code architecture.

²⁹ The Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. It is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as programming language statements and so on

THE STUDENT'S THESIS PROJECT

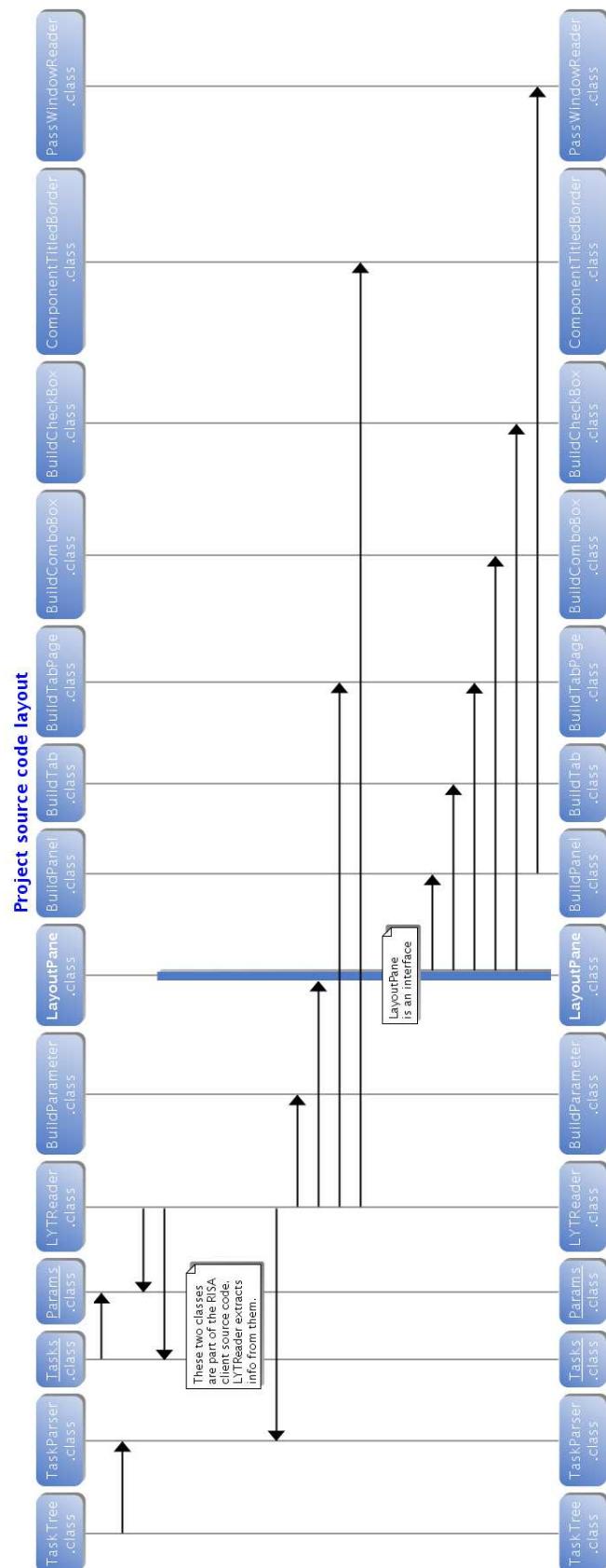


Figure 3.11: The UML data model of the project

3.2 RESULTS AND FUTURE WORK

3.2 RESULTS AND FUTURE WORK

By the end of the placement period, a new Graphical User Interface was produced for manipulating all the SAS tasks that are available via the RISA service. The new RISA parameter GUI was built with simplicity, practicality and reliability in mind, making it easier for the end user to control all the parameters and tasks provided in RISA. Since RISA will substitute SAS soon, developing a practical and easy to use parameter GUI was a crucial step for helping RISA evolve into the next standard for the X-ray astronomy community which utilizes XMM-Newton data for scientific research and analysis.

3.2.1 *Scientific importance*

Apart from developing a new parameter GUI for the RISA application, which improved its usability, the current thesis work gave RISA another strong feature: flexibility. The new GUI was developed in such a way that enables RISA to dynamically produce any kind of SAS task GUI, as long as a corresponding *.lyt* and *.par* file is provided for generating the new task's GUI. So once a new task is developed and implemented in SAS, the corresponding RISA application will be able to automatically generate the new task GUI on the fly, without the need of any alterations or patches to the actual RISA source code, just by gaining read-only access to the new task's corresponding *.lyt* and *.par* file.

Given the flexibility and the significant improvement of the actual parameter GUI which makes RISA very user friendly, RISA is constantly gaining ground among the astronomical / astrophysical X-ray research community. This combined with the advanced technologies RISA utilizes (client-server interface, GRID technology, multiplatform and architecture-independent application etc) makes RISA a very appealing solution for X-ray astronomers / astrophysicists. It is also a very appealing concept for other applications to follow, proving that web-technologies in combination with the power of object-oriented programming languages like *Java* and advanced technologies like GRID computing can bring exquisite results such as very powerful applications

working under low tech hardware, low bandwidth Internet connections but delivering high quality results nevertheless.

Therefore, while this thesis project is far from being considered a world class breakthrough in software technology, it is a valuable component greatly improving the usability of a largely growing and very promising astrophysical application.

3.2.2 *Hardware technology used*

The entire thesis code was developed under Linux using low end hardware. In particular the main tool of work was an affordable Asus EeePC 900 netbook with the following specs: Intel Celeron 900MHz CPU, 1Gb RAM, 8Gb Solid State Disk, Intel GMA 950 GPU operating under *Xandros Linux Light*³⁰ OS.

Despite the low end hardware technology included in this particular type of personal computers, the processing power was more than enough for the job. For that reason this thesis can also be considered as a technological demonstration of the ability to produce very satisfactory results using high end software technology combined with low end hardware technology which is nowadays considered satisfactory mainly for surfing the internet (ASUS calls the EeePC a "mobile Internet device").

3.2.3 *Software technology used*

The entire thesis source code was developed using open-source software tools. As stated above, the OS used was a Debian based distribution of Linux. The IDE (Integrated Development Environment) used was *Eclipse*³¹ *Ganymede 3.4*. The JVM (Java Virtual Machine) runtime used was the one included in the *JDK* (Java Developer's Kit) *1.5.0*.

³⁰ Debian-based OS developed by the Xandros company
www.xandros.com

³¹ www.eclipse.org

Furthermore, apart from the technical side, open-source software was also used for the documentation of the thesis work. \LaTeX ³² was used for typesetting the text, *Texmaker*³³ was used for writing the thesis text, *Ubuntu*³⁴ 9.04 was the OS under which the documentation part was realized.

Therefore this thesis also underlines the importance of open-source Operating Systems and software tools. A good example of this is the fact that *Eclipse* and *Netbeans*³⁵ are dominant in the developers' world, major technology / research centres like ESAC use Linux for all the desktops and servers on site, \LaTeX is widely used for documenting scientific or technological research results in universities and research centres and so on.

Sophisticated open-source software allows users and developers to make use of all the available resources such as CPU power and RAM quantity in a very efficient way. An example which proves the dominance of Unix-based Operating Systems is the fact that any *Windows Vista* 32bit version has a 3Gb RAM max limit (64bit advanced versions like Vista Business can support up to 128Gb of RAM), *Windows 7* 32bit version has a 4Gb RAM max limit (64bit versions can support up to 192Gb of RAM)³⁶. Linux distributions face a similar restriction unless the end-user installs the Physical Address Extension (PAE) aware kernel³⁷, which is a very simple and straightforward procedure if you use apt-get or similar package managers. That way a 32bit OS can actually address and handle its RAM like 64Bit OSs do hence overcoming all RAM limitations.

³² \LaTeX is a document markup language and document preparation system for the TeX typesetting program. It is most widely used by mathematicians, scientists, engineers, philosophers, economists and other scholars in academia and the commercial world, and other professionals.

³³ Texmaker is a cross-platform open source \LaTeX editor that integrates many tools needed to develop documents with \LaTeX , in just one application. <http://www.xm1math.net/texmaker/>

³⁴ www.canonical.com

³⁵ www.netbeans.org

³⁶ More information at <http://news.softpedia.com/news/Windows-7-Maximum-Supported-RAM-192-GB-RAM-119101.shtml>

³⁷ More information at <http://www.cyberciti.biz/faq/ubuntu-linux-4gb-ram-limitation-solution/>

3.2.4 Remaining work

By the end of the traineeship period, the development of the source code of the project was almost complete with only a few minor details to be taken care of. Once the coding was finished, there were a few ideas about further improving the final product. They were originally scheduled to be implemented within the next few months after the end of the traineeship period before or while working on the documentation of the thesis project.

Due to an unfortunate incident in mid-2009, the thesis work saved in the author's computers and backup hardware was lost since all computer hardware was stolen. This led to a major delay in the completion of the project.

This major delay led to documenting the thesis work based on very few resources like a few screenshots of the final product, non-functional pieces of *Java* code and so on while following a long and time-consuming procedure of gathering every lost piece of information again. Furthermore, given the fact that a very satisfactory postgraduate opportunity came up, starting on March 2010, the time remaining in order to improve or expand any features of the thesis product was very little making it very difficult to implement all the planned improvements and new features.

Some of the improvements that are required in order for the final product to be considered complete:

- 1 Complete the XML logging system so that it will pass all the necessary information to an *.xml* log file following the XML syntax required by the SAS parser in order to process the document correctly.
- 2 Deal with the GUI window contents refresh bug. Some widgets are visible only after refreshing the window contents more than once. – Solution: The window contents should be refreshed at least once right after the frame being generated.
- 3 Deal with the window frame decoration bug. The first time any task window pops up, the window frame decoration follows the Operating System's theme

3.2 RESULTS AND FUTURE WORK

colours and style. Normally the frame decoration is standard and independent of the OS theme colours and style. – Solution: Unknown for the time being.

- 4 Deal with the frame default dimensions bug. Some times frames appear smaller or much bigger instead of being adjusted on the frame contents. – Solution: Force the frame to be resized depending on the window contents.
- 5 Deal with the RISA tree mouse listener improvement. For the time being whenever the cursor is situated on a task leaf, any mouse button will trigger the action of the task window to pop up. – Solution: Set the mouse listener trigger to work only when left-clicking on the leaf once (or twice depending on the end-users' preference).

New features that should be implemented to the final product:

- 1 Add a (possibly numbering) sequence leading the end user through the correct steps before proceeding to requesting certain SAS products³⁸.
- 2 Add explanatory notes right next to each task leaf on the task tree. That way a new or inexperienced user will know what each task is capable of.
- 3 Add explanatory pop up tabs of each widget's elements, a very helpful feature for new SAS users who haven't read the SAS manual extensively yet and a good reminder for the experienced SAS users.

3.2.5 *Lessons learned*

The overall experience of developing this thesis which included working at the European Space Agency on a soft-

³⁸ Each XMM-Newton data set needs to be processed using some particular preparatory SAS tasks before it can be considered ready for scientific analysis. Therefore there should be some kind of guide leading the inexperienced user through the first few preparatory steps before proceeding to any kind of scientific analysis. This can be done by following a numbered pattern or possibly different techniques which indicate the correct task sequence that has to be followed before proceeding to any other tasks which conduct analysis and data extraction.

THE STUDENT'S THESIS PROJECT

ware based project, trying to develop part of an astrophysical application, was more than just didactic. It involved several new concepts and ways of work such as learning an object oriented programming language in a very tight time-frame, working independently, working on a multinational professional (i.e non academic) environment, improvement of proactive skills, communication skills and working on a multidisciplinary field.

From the programming point of view, new knowledge on object oriented programming and particularly in *Java* was acquired.

A new way of thinking was also acquired, leading to acquiring a broader technological spectrum and a deeper understanding of software engineering mentality.

From the working methodology point of view, a more independent way of working and thinking was developed by extracting and validating information and knowledge from various sources whenever needed as well as experimenting on and testing new techniques with little or no guidance from the tutors.

Overall the knowledge obtained and the experience acquired throughout the process of developing this thesis contributed to a deeper understanding of different fields such as software engineering and astronomy, as well as acquiring a more liberal way of thinking.

BIBLIOGRAPHY

- [1] Aitor Ibarra, Ignacio de la Calle, Carlos Gabriel, Jesús Salgado, Pedro Osuna, *XMM-Newton Science Analysis Software: How to bring new technologies to long-life satellite missions*. ADASS XVII 2008.
- [2] Aitor Ibarra, Ignacio de la Calle, Daniel Tapiador, Nora Loiseau, Carlos Gabriel, Jesús Salgado, Pedro Osuna, *Remote Interface to Science Analysis Tools for Grid Architecture: The XMM-Newton SAS Case*. ADASS XVI 2006.
- [3] Carlos Gabriel, Matteo Guainazzi, Leo Metcalfe, *The ESA XMM-Newton Science Operations Centre: Making Basic Space Science Available to the Whole Scientific World*.
- [4] *XMM-Newton Image Gallery*:
http://xmm.esac.esa.int/external/xmm_science/gallery/public/index.php
- [5] *XMM-Newton: A Technical Description*:
http://xmm.esac.esa.int/external/xmm_user_support/documentation/technical/
- [6] *XMM-Newton Science Analysis System*
<http://xmm.vilspa.esa.es/sas/6.5.0/doc/index.html>
- [7] Matteo Guainazzi
An introduction to XMM-Newton data analysis and the SAS grand-scheme http://xmm.esac.int/external/xmm_data_analysis/sas_workshops/sas_ws9_files/presentations/Matteo_SAS_introduction.pdf
- [8] *Enabling a robust VOSpace based on iRODS for astronomers*
<https://www.irods.org/index.php/VOSpace>
- [9] *JDKTM 5.0 Documentation*
<http://java.sun.com/j2se/1.5.0/docs/>
- [10] *Java examples (example source code)*
<http://www.java2s.com/>

Bibliography

- [11] *Wikipedia - The Free Encyclopedia*
[http://en.wikipedia.org/wiki/Java_\(programming_language\)](http://en.wikipedia.org/wiki/Java_(programming_language))
- [12] *Wikipedia - The Free Encyclopedia*
<http://en.wikipedia.org/wiki/GridWay>
- [13] *Wikipedia - The Free Encyclopedia*
http://en.wikipedia.org/wiki/Modal_window
- [14] *Wikipedia - The Free Encyclopedia*
[http://en.wikipedia.org/wiki/Interface_\(computer_science\)](http://en.wikipedia.org/wiki/Interface_(computer_science))
- [15] *Wikipedia - The Free Encyclopedia*
http://en.wikipedia.org/wiki/Unified_Modeling_Language
- [16] *Wikipedia - The Free Encyclopedia*
<http://en.wikipedia.org/wiki/LaTeX>

Part IV

APPENDICES

A

SAS TASKS XML FILE

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <BODY>
3 <Workflow value="BASIC">
4   <Task level="Observation">odfingest</Task>
5   <Task level="Observation">nodfingest</Task>
6   <Task level="Observation">cifbuild</Task>
7   <Task level="Observation">odffix</Task>
8   <Task level="Observation">attmove</Task>
9    <Task level="Observation">atthkgen</Task>
10   <Task level="Observation">attbin</Task>
11   <Task level="Observation">attfilter</Task>
12   <Task level="Observation">hkgtigen</Task>
13   <Task level="Observation">attcalc</Task>
14   <Task level="Observation">odfcheck</Task>
15   <Task level="Observation">attcheck</Task>
16   <Task level="Observation">hkstrip</Task>
17   <Task level="Observation">hkauxplot</Task>
18 </Workflow>
19 <Workflow value="PN">
20   <Task level="Observation">epchain</Task>
21   <Task level="Observation">epreject</Task>
22   <Task level="Observation">eposcorr</Task>
23   <Task level="Observation">epauxcomb</Task>
24   <Task level="Observation">epframes</Task>
25   <Task level="Observation">epevents</Task>
26   <Task level="Observation">epexposure</Task>
27   <Task level="Observation">epproc</Task>
28   <Task level="Observation">epiclccorr</Task>
29 </Workflow>
30 <Workflow value="MOS">
31   <Task level="Observation">emchain</Task>
32   <Task level="Observation">emframes</Task>
33   <Task level="Observation">emeventsproj</Task>
34   <Task level="Observation">emevents</Task>
35   <Task level="Observation">emenergy</Task>
36   <Task level="Observation">emdiag</Task>
37   <Task level="Observation">embadpixfind</Task>
38   <Task level="Observation">emproc</Task>
39 </Workflow>
```

SAS TASKS XML FILE

```

41 | <Workflow value="EPIC">
41 |   <Task level="Instrument">emldetect</Task>
43 |   <Task level="Instrument">epatplot</Task>
43 |   <Task level="Instrument">ebadpixupdate</Task>
45 |   <Task level="Instrument">badpixfind</Task>
45 |   <Task level="Instrument">gtimerge</Task>
47 |   <Task level="Instrument">eboxdetect</Task>
47 |   <Task level="Instrument">arfgen</Task>
49 |   <Task level="Instrument">eexpchipmap</Task>
49 |   <Task level="Instrument">esplinemap</Task>
51 |   <Task level="Instrument">rmfgen</Task>
51 |   <Task level="Instrument">backscale</Task>
51 |   <Task level="Instrument">edetect_chain</Task>
53 |   <Task level="Instrument">especget</Task>
53 |   <Task level="Instrument">eregionanalyse</Task>
55 |   <Task level="Instrument">etimeget</Task>
55 |   <Task level="Instrument">echeckregion</Task>
57 |   <Task level="Instrument">eexpmap</Task>
57 |   <Task level="Instrument">lccorr</Task>
59 |   <Task level="Instrument">specadd</Task>
59 |   <Task level="Instrument">movecalc</Task>
61 |   <Task level="Instrument">lcplot</Task>
61 |   <Task level="Instrument">gtialign</Task>
63 |   <Task level="Instrument">ewavelet</Task>
63 |   <Task level="Instrument">evalcorr</Task>
65 |   <Task level="Instrument">especplot</Task>
65 |   <Task level="Instrument">esensmap</Task>
67 |   <Task level="Instrument">emask</Task>
67 |   <Task level="Instrument">elcplot</Task>
69 |   <Task level="Instrument">efftplot</Task>
69 |   <Task level="Instrument">inthist</Task>
71 |   <Task level="Instrument">implot</Task>
71 |   <Task level="Instrument">evarimgen</Task>
73 |   <Task level="Instrument">esrcselect</Task>
73 |   <Task level="Instrument">esrcmerge</Task>
75 |   <Task level="Instrument">esrcfetch</Task>
75 |   <Task level="Instrument">esources</Task>
77 |   <Task level="Instrument">esky2det</Task>
77 |   <Task level="Instrument">emosaic</Task>
79 |   <Task level="Instrument">ekstest</Task>
79 |   <Task level="Instrument">dpssflag</Task>
81 |   <Task level="Instrument">evproject</Task>
81 |   <Task level="Instrument">elcbuild</Task>
83 |   <Task level="Instrument">region</Task>
83 |   <Task level="Instrument">colimplot</Task>
85 |   <Task level="Instrument">evigweight</Task>
85 |   <Task level="Instrument">esensitivity</Task>
87 |   <Task level="Instrument">econvolverprep</Task>
87 |   <Task level="Instrument">sensitivity</Task>
89 |   <Task level="Instrument">flspec</Task>
89 |   <Task level="Instrument">flmask</Task>

```

SAS TASKS XML FILE

```

91  <Task level="Instrument">srcdisplay</Task>
92  <Task level="Instrument">backcorr</Task>
93 </Workflow>
94 <Workflow value="RGS">
95  <Task level="Observation">rgsoffsetcalc</Task>
96  <Task level="Observation">rgsbkgmodel</Task>
97  <Task level="Observation">rgsbkgmodelTest</Task>
98  <Task level="Observation">rgsevconvert</Task>
99  <Task level="Instrument">rgsspectrum</Task>
100 <Task level="Instrument">rgssources</Task>
101 <Task level="Instrument">rgsrnfgen</Task>
102 <Task level="Instrument">rgsmcrgen</Task>
103 <Task level="Instrument">rgsregions</Task>
104 <Task level="Instrument">rgslinepos</Task>
105 <Task level="Observation">rgsframes</Task>
106 <Task level="Instrument">rgsfluxer</Task>
107 <Task level="Observation">rgsfilter</Task>
108 <Task level="Observation">rgsevents</Task>
109 <Task level="Observation">rgsenergy</Task>
110 <Task level="Observation">rgscombine</Task>
111 <Task level="Observation">rgsbadpix</Task>
112 <Task level="Instrument">rgsspecplot</Task>
113 <Task level="Instrument">rgsimplot</Task>
114 <Task level="Observation">rgsauxtable</Task>
115 <Task level="Observation">rgsangles</Task>
116 <Task level="Observation">rgssuperrmf</Task>
117 <Task level="Observation">rgsproc</Task>
118 <Task level="Instrument">rgslccorr</Task>
119 <Task level="Observation">rgsprods</Task>
120 <Task level="Observation">htrframes</Task>
121 </Workflow>
122 <Workflow value="OM">
123  <Task level="Observation">omthlcplot</Task>
124  <Task level="Observation">omgrismplot</Task>
125  <Task level="Observation">omflatindex</Task>
126  <Task level="Observation">omcomb</Task>
127  <Task level="Observation">omthconv</Task>
128  <Task level="Observation">omsrclistcomb</Task>
129  <Task level="Observation">omregion</Task>
130  <Task level="Observation">omprep</Task>
131  <Task level="Observation">ommosaic</Task>
132  <Task level="Observation">ommodmap</Task>
133  <Task level="Observation">ommag</Task>
134  <Task level="Observation">omlcbuild</Task>
135  <Task level="Observation">omgrism</Task>
136  <Task level="Observation">omgprep</Task>
137  <Task level="Observation">omflatgen</Task>
138  <Task level="Observation">omflatfield</Task>
139  <Task level="Observation">omfastshift</Task>
140  <Task level="Observation">omfastflat</Task>
141  <Task level="Observation">omdrifthist</Task>

```

SAS TASKS XML FILE

```

143   <Task level="Observation">omdetect</Task>
144   <Task level="Observation">omcosflag</Task>
145   <Task level="Observation">omatt</Task>
146   <Task level="Observation">omphkgen</Task>
147   <Task level="Observation">omslewchain</Task>
148   <Task level="Observation">omsource</Task>
149   <Task level="Observation">omichain</Task>
150   <Task level="Observation">omgsource</Task>
151   <Task level="Observation">omgchain</Task>
152   <Task level="Observation">omfchain</Task>
153   <Task level="Observation">rudiframetime</Task>
154   <Task level="Observation">movecalc</Task>
155   <Task level="Observation">lcpplot</Task>
156   <Task level="Observation">gtialign</Task>
157   <Task level="Observation">implot</Task>
158   <Task level="Observation">colimplot</Task>
159   <Task level="Observation">srcdisplay</Task>
160 </Workflow>
161 <Workflow value="GENERAL">
162   <Task level="Instrument">merge</Task>
163   <Task level="Instrument">evselect</Task>
164   <Task level="Instrument">dstoplot</Task>
165   <Task level="Instrument">dsplot</Task>
166   <Task level="Instrument">badpix</Task>
167   <Task level="Instrument">phasecalc</Task>
168   <Task level="Instrument">tabgtigen</Task>
169   <Task level="Instrument">orbit</Task>
170   <Task level="Instrument">colsmooth</Task>
171   <Task level="Instrument">asmooth</Task>
172   <Task level="Instrument">calview</Task>
173   <Task level="Instrument">timeappend</Task>
174   <Task level="Instrument">evlistcomb</Task>
175   <Task level="Instrument">statsget</Task>
176 </Workflow>
177 <Workflow value="Thread">
178   <Task level="Observation">epic_event_thread</Task>
179   <Task level="Instrument">epic_lightcurve_thread</Task>
180   <Task level="Instrument">epic_spectrum_thread</Task>
181   <Task level="Instrument">epic_edetectchain_thread</Task>
182   <Task level="Instrument">epic_analysis_thread</Task>
183   <Task level="Observation">rgs_thread</Task>
184     <Task level="Observation">epic_slew_thread</Task>
185 </Workflow>
</BODY>

```

B

TASKPARSER.JAVA

```
1 package layoutCreator;  
3 import java.util.ArrayList;  
import java.util.Collections;  
5 import java.util.HashMap;  
import java.util.Iterator;  
7 import java.util.List;  
import org.jdom.Document;  
9 import org.jdom.Element;  
import org.jdom.input.DOMBuilder;  
11 import org.jdom.input.SAXBuilder;  
import com.sun.org.apache.xerces.internal.parsers.  
    DOMParser;  
13  
14 public class TaskParser {  
15     SAXBuilder builder;  
16     Document doc;  
17     Element xmlRootElement;  
18     static Element child;  
19     static Element ch;  
20     static Element root;  
21     static Element childe;  
22  
23  
24     private HashMap<String,List<String>> _myMap = new  
25         HashMap<String,List<String>>(); //Create a  
26         HashMap containing a mapping of 2 types of  
27         elements: a string and a list of strings  
28  
29     public void readXMLFile(String file)  
30     {  
31         System.out.println("Reading ");  
32         file = "SASData_v2.xml"; //Set the file that's  
            about to be parsed  
33         DOMBuilder builder = new DOMBuilder();  
         DOMParser parser = new DOMParser();  
34  
35         // Read the entire document into memory
```

TASKPARSER.JAVA

```
35  try {
36      parser.parse(file);
37      org.w3c.dom.Document domDoc = parser.
38          getDocument();
39      org.jdom.Document jdomDoc = builder.build
40          (domDoc);
41      Element root = jdomDoc.getRootElement();
42      final List<?> allChildren = root.
43          getChildren(); //Create a list of all
44          the parsed words
45
46      Iterator<?> itr = allChildren.iterator();
47          //Add an iterator to allChildren
48
49      while (itr.hasNext())
50      {
51          List<String> taskList =new ArrayList<
52              String>();
53          Element child = (Element) itr.next(); //
54              Return the next element
55          child.getAttributeValue("value"); //Get
56          each workflow name
57
58          final List<?> childe = child.getChildren
59          (); //Add each workflow name to childe
60          Iterator<?> it = ((List<?>) childe).
61              iterator(); //Add an iterator to
62              childe
63
64          while (it.hasNext())
65          {
66              Element ch = (Element) it.next(); //
67                  Return the next element
68              ch.getText(); //Get each task name
69              ch.getAttributeValue("level"); //Get
70                  the task level (Observation or
71                  Instrument)
72              taskList.add(ch.getText()); //Add the
73                  task name to taskList
74              Collections.sort(taskList); //Sort
75                  taskList by alphabetical order
76              System.out.println("Tasks "+ch.getText
77                  ());
78          }
79
80          _myMap.put(child.getAttributeValue("value
81              "),taskList); //Add each workflow and
82              its corresponding tasks in the HashMap
83      }
84  }
85  catch (Exception e) {
```

TASKPARSER.JAVA

```
67    }
69
70    public HashMap<String, List<String>> getInfo()
71    {
72        System.out.println("My Map: "+_myMap);
73        return _myMap; //Return the contents of _myMap
74    }
75 }
```


C

TASKTREE.JAVA

```
1 import java.awt.Color;
2 import java.awt.Dimension;
3 import java.awt.event.MouseAdapter;
4 import java.awt.event.MouseEvent;
5 import java.util.HashMap;
6 import java.util.List;
7 import javax.swing.ImageIcon;
8 import javax.swing.JFrame;
9 import javax.swing.JRootPane;
10 import javax.swing.JScrollPane;
11 import javax.swing.JTree;
12 import javax.swing.UIManager;
13 import javax.swing.event.TreeSelectionListener;
14 import javax.swing.tree.DefaultMutableTreeNode;
15 import javax.swing.tree.MutableTreeNode;
16 import javax.swing.tree.TreeSelectionModel;
17 import layoutCreator.LYTReader;
18 import layoutCreator.TaskParser;
19
20 public class TaskTree {
21
22     DefaultMutableTreeNode parent = null;
23     static JTree tree = null;
24
25     public TaskTree()
26     {
27         //Read the parsed info
28         TaskParser myParse = new TaskParser();
29         myParse.readXMLFile("SASData_v2.xml");
30
31         //Create a tree , tree nodes and a frame
32         DefaultMutableTreeNode basic = new
33             DefaultMutableTreeNode("BASIC");
34         DefaultMutableTreeNode nBasic;
35
36         DefaultMutableTreeNode pn = new
37             DefaultMutableTreeNode("PN");
38         DefaultMutableTreeNode nPn;
```

TASKTREE.JAVA

```
39         DefaultMutableTreeNode mos = new
40             DefaultMutableTreeNode("MOS");
41             DefaultMutableTreeNode nMos;
42
43         DefaultMutableTreeNode epic = new
44             DefaultMutableTreeNode("EPIC");
45             DefaultMutableTreeNode nEpic;
46
47         DefaultMutableTreeNode rgs = new
48             DefaultMutableTreeNode("RGS");
49             DefaultMutableTreeNode nRgs;
50
51         DefaultMutableTreeNode om = new
52             DefaultMutableTreeNode("OM");
53             DefaultMutableTreeNode nOm;
54
55         DefaultMutableTreeNode gen = new
56             DefaultMutableTreeNode("GENERAL");
57             DefaultMutableTreeNode nGen;
58
59         DefaultMutableTreeNode thread = new
60             DefaultMutableTreeNode("THREAD");
61             DefaultMutableTreeNode nThread;
62
63 //Create a HashMap containing a mapping of 2
64 //types of elements: a string and a list of
65 //strings
66 final HashMap<String, List<String>> info =
67     myParse.getInfo();
68
69 //Extract each workflow separately
70 for (int k = 0 ; k < info.get("BASIC").size()
71     ; k++)
72 {
73     nBasic = new DefaultMutableTreeNode(info.get(
74         "BASIC").get(k));
75     basic.add(nBasic);
76 }
77
78 for (int k = 0 ; k < info.get("PN").size() ; k
79    ++)
80 {
81     nPn = new DefaultMutableTreeNode(info.get(
82         "PN").get(k));
83     pn.add(nPn);
84 }
85
86 for (int k = 0 ; k < info.get("MOS").size() ;
87     k++)
88 {
89 }
```

TASKTREE.JAVA

```

    nMos = new DefaultMutableTreeNode(info.get
        ("MOS").get(k));
    mos.add(nMos);
}
for (int k = 0 ; k < info.get("EPIC").size() ;
    k++)
{
    nEpic = new DefaultMutableTreeNode(info.
        get("EPIC").get(k));
    epic.add(nEpic);
}
for (int k = 0 ; k < info.get("RGS").size() ;
    k++)
{
    nRgs = new DefaultMutableTreeNode(info.get
        ("RGS").get(k));
    rgs.add(nRgs);
}
for (int k = 0 ; k < info.get("OM").size() ; k
++)
{
    nOm = new DefaultMutableTreeNode(info.get
        ("OM").get(k));
    om.add(nOm);
}
for (int k = 0 ; k < info.get("GENERAL").size
() ; k++)
{
    nGen = new DefaultMutableTreeNode(info.
        get("GENERAL").get(k));
    gen.add(nGen);
}
for (int k = 0 ; k < info.get("Thread").size()
; k++)
{
    nThread = new DefaultMutableTreeNode(info
        .get("Thread").get(k));
    thread.add(nThread);
}
//Build the tree and add the extracted data in it
parent = new DefaultMutableTreeNode("SAS tasks",
    true);
parent.add(basic);
parent.add(pn);

```

TASKTREE.JAVA

```
113     parent.add(mos);
115     parent.add(epic);
117     parent.add(rgs);
119     parent.add(om);
121     parent.add(gen);
123     parent.add(thread);

127 //Improve the looks of the window containing the
129 tree = new JTree(parent);
131 UIManager.put("Tree.line", Color.GREEN);
133 UIManager.put("Tree.openIcon",new ImageIcon("config/postM.png"));
135 tree.putClientProperty("JTree.lineStyle", "Horizontal");
137 tree.updateUI();

141 //Create a frame
143 final JFrame frame = new JFrame("RISA");

145 frame.add(tree);
147 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
149 frame.setLocationRelativeTo(null);
151 Dimension minimumSize = new Dimension();
153 minimumSize.setSize(180,210);
155 frame.setUndecorated(true);
157 frame.getRootPane().setWindowDecorationStyle(
159     JRootPane.FRAME);
161 frame.setMinimumSize(minimumSize);
163 frame.pack();
165 frame.setVisible(true);
167 frame.setLocale(new java.util.Locale("en", "US"));

171 //Create a scrollbar whenever needed
173 JScrollPane scrollPane = new JScrollPane();
175 scrollPane.setViewport().add(tree);
177 frame.add(scrollPane);

181 //Mouse listener
183 tree.addMouseListener(new MouseAdapter() {
185     public void mouseClicked(MouseEvent e) {
187         doMouseClicked(e);
189     }
191 });
193 }

195 //Create a main and instruct it to "read" the
197 ListenerTest class
```

TASKTREE.JAVA

```
157 public static void main(String[] args) {
159     new TaskTree();
161 }
163 //Mouse listener function. It which will give
164     output only when a tree leaf is clicked with
165     any mouse button
166 /*Possible improvement: Change this to "left
167     mouse button only".*/
168     static void doMouseClicked(MouseEvent e) {
169
170         DefaultMutableTreeNode node = (
171             DefaultMutableTreeNode)
172             tree.getLastSelectedPathComponent();
173
174         if (node == null) return;
175
176         Object nodeInfo = node.getUserObject();
177
178         if (node.isLeaf()) {
179
180             //Read and show the windows (based on the .
181             //lyt and .par files of SAS) corresponding
182             //to each SAS task
183             LYTReader reader = new LYTReader();
184
185             boolean flag = reader.open(nodeInfo.
186                 toString());
187             if (flag)
188             {
189                 reader.parse();
190
191                 reader.show();
192             }
193         }
194     }
```


D

PARAMS.JAVA

```
1 package sasTask;  
3 import java.util.HashMap;  
import java.util.TreeMap;  
5 import java.util.logging.Level;  
import java.util.logging.Logger;  
7  
import org.w3c.dom.Node;  
9 import org.w3c.dom.NodeList;  
import org.apache.xerces.parsers.*;  
11  
13 public class Params { //This class parses all the  
info included in each .par file  
15 private TreeMap<String, String> params = null;  
private HashMap<String, String> mandatoryParams =  
new HashMap<String, String>();  
17 private HashMap<String, String> _paramType = new  
HashMap<String, String>();  
19 public Params()  
{  
21     params = new TreeMap<String, String>(); //  
Constructs a new, empty map of strings,  
sorted according to the keys' natural order  
}  
23  
25 public TreeMap<String, String> readSASPParam(String  
SASTask)  
{  
27     String file = ("config/" + SASTask + ".par"); //  
Load the .par file  
//Initiate parsing  
DOMParser parser = new DOMParser(); //Xerces-  
specific parser class  
29  
31     try {  
// Read the entire .par file into memory
```

PARAMS.JAVA

```
33         parser.parse(file.toString());
35
36     org.w3c.dom.Document domDoc = parser.
37         getDocument();
38
39     params = new TreeMap<String, String>(); // 
40         Constructs a new, empty map of strings,
41         sorted according to the keys' natural
42         order containing the parameters of each .
43         par
44
45     NodeList nodes = domDoc.getElementsByTagName(
46         "PARAM"); //Returns a NodeList of all the
47         Elements in document order with a given
48         tag name (PARAM) and are contained in the
49         document
50
51     for (int i = 0; i < nodes.getLength(); i++) {
52         Node element = (Node) nodes.item(i);
53
54         String attribute = "";
55         if (element.getAttributes().
56             getNamedItem("default") == null)
57             attribute = "";
58         else
59             attribute = element.getAttributes().
60                 getNamedItem("default").
61                 getNodeValue();
62
63         params.put(element.getAttributes().
64             getNamedItem("id").getNodeValue(),
65             attribute);
66
67         if (element.getAttributes().getNamedItem
68             ("mandatory") != null)
69             mandatoryParams.put(element.
70                 getAttributes().getNamedItem("id").
71                 getNodeValue(), "true"); //Add the
72                 values "element+attributes+id+
73                 idvalue" and "true" to the
74                 mandatoryParams HashMap
75         else
76             mandatoryParams.put(element.
77                 getAttributes().getNamedItem("id").
78                 getNodeValue(), "false"); //Add the
79                 values "element+attributes+id+
80                 idvalue" and "false" to the
81                 mandatoryParams HashMap
82
83         _paramType.put(element.getAttributes().
84             getNamedItem("id").getNodeValue(),
```

PARAMS.JAVA

```

        element.getAttributes().getNamedItem(
            "type").getNodeValue();

59    Logger.getLogger("RISAWS").info("id=" +
        element.getAttributes().getNamedItem(
            "id").getNodeValue()
        + " default=" + attribute);
61}
63
65} catch (Exception e) { //It indicates a
    wrong format or validity error
    Logger.getLogger("RISAWS").log(Level.SEVERE
        ,file + " is not valid. ",e.getMessage()
    ); //Log the error encountered in the .
        par file and print it in the RISAWS
67}
69    return params;
71}
73public void setParam(String param, String value)
{
    Logger.getLogger("RISAWS").info("param "+param
        + " value "+value); //Log each parameter and
        its value
75    params.put(param, value); //Store each
        parameter and its default value to the
        TreeMap
77}
79public TreeMap<String, String> getParam()
{
    return params; //Return the parameters saved in
        the TreeMap
81}
83public String getType(String param)
{
85    return _paramType.get(param); //Return the type
        of parameters saved in the HashMap
87}
89public boolean isMandatory(String param)
{
91    String mandatory = mandatoryParams.get(param);
        //Extract the mandatoryParams parameters
93    if (mandatory == "true")
    {

```

PARAMS.JAVA

```
95 }      return true;  
96 }  
97 else      return false;  
98 }  
99 }
```

E

TASKS.JAVA

```
1 package sasTask;
2
3
4 import java.util.Iterator;
5 import java.util.TreeMap;
6
7
8 public class Tasks {
9
10    private Params _params = null;
11
12    private TreeMap<String, Params> _tasksMap = null;
13
14    private TreeMap<String, String> _levels = null;
15
16    public Tasks()
17    {
18        _params = new Params(); //Calls the Params
19                    method of Params.java
20        _tasksMap = new TreeMap<String, Params>(); //Constructs a new TreeMap containing a string
21                    and the Params class output
22        _levels = new TreeMap<String, String>(); //Constructs a new, empty map, sorted
23                    according to the keys' natural order
24    }
25
26    public Params getTasks(String task)
27    {
28        return _tasksMap.get(task); //Returns the task
29                    that's saved in the _tasksMap.
30    }
31
32    public TreeMap<String,Params> getTasks()
33    {
34        return _tasksMap; //Returns the _tasksMap
35                    containing all the tasks
36    }
37
38}
```

TASKS.JAVA

```
34     public void readParams(String taskName)
35     {
36         _params.readSASParam(taskName);
37     }
38
39     public void printInfo()
40     {
41         Iterator<?> myWorkflowIterator = _tasksMap.
42             keySet().iterator(); //Iterate the keys of
43             the _tasksMap TreeMap
44         while(myWorkflowIterator.hasNext()) { //For all
45             the elements included in the _tasksMap
46             String entry = (String) myWorkflowIterator.
47                 next();
48             System.out.println("TASKS: "+entry); //Print
49                 each element of the _tasksMap
50         }
51     }
52
53     public void setTask(String task)
54     {
55         _tasksMap.put(task, _params); //Associate the
56             tasks included in the _tasksMap with the
57             parameters included in the _params
58         _params = new Params();
59     }
60
61     public void setLevel(String task, String level)
62     {
63         _levels.put(task, level); //Associate the tasks
64             included in the _tasksMap with a level
65     }
66
67     public void append(Tasks task)
68     {
69         Iterator<?> myWorkflowIterator = task.getTasks
70             ().keySet().iterator(); //Iterate the tasks
71         while(myWorkflowIterator.hasNext()) { //For all
72             the tasks
73             String entry = (String) myWorkflowIterator.
74                 next();
75             _tasksMap.put(entry, task.getTasks().get(
76                 entry)); //Associate each value with an
77                 iterator key
78         }
79
80         Iterator<String> levelsIt = task._levels.keySet
81             ().iterator(); //Iterate the levels
82         while(levelsIt.hasNext()) //For all the keys
83             contained in the _levels
```

TASKS.JAVA

```
70     {
71         String entryLevel = (String) levelsIt.next();
72         _levels.put(entryLevel,task._levels.get(
73             entryLevel)); //Associate each value with
74             an iterator key
75     }
76
77     public void filter(String level) {
78
79         Iterator<String> myLevelIterator = _levels.
80             keySet().iterator();
81         while(myLevelIterator.hasNext()) {
82             String taskLevel = (String) myLevelIterator.
83                 next();
84
85             if (!_levels.get(taskLevel).equals(level)) ///
86                 If the iterated string in _levels is not
87                 equal to a level
88             _tasksMap.remove(taskLevel); //Remove this
89                 particular string
90         }
91     }
92 }
```


F

LYTREADER.JAVA

```
1 package layoutCreator;
2
3 import java.awt.BorderLayout;
4 import java.awt.Component;
5 import java.awt.GridBagConstraints;
6 import java.awt.event.ActionListener;
7 import java.io.BufferedReader;
8 import java.io.FileReader;
9 import java.io.IOException;
10 import java.util.ArrayList;
11 import java.util.Iterator;
12 import java.util.List;
13 import java.util.TreeMap;
14 import java.util.regex.Matcher;
15 import java.util.regex.Pattern;
16 import java.awt.event.ActionEvent;
17 import java.awt.event.ItemEvent;
18 import java.awt.event.ItemListener;
19 import javax.swing.BorderFactory;
20 import javax.swing.JButton;
21 import javax.swing.JCheckBox;
22 import javax.swing.JFrame;
23 import javax.swing.JScrollPane;
24 import sasTask.Params;
25 import sasTask.Tasks;
26
27 public class LYTReader{
28
29     private static final long serialVersionUID = 1L;
30
31     private JFrame _frame = null;
32     private BufferedReader _in;
33     private boolean _ok = true;
34     private static int i;
35     private boolean _pageFlag = false;
36     private JButton jButton = null;
37     private JButton jButton1 = null;
38     public String s = null;
```

LYTREADER.JAVA

```
40  private List<LayoutPane> _frameList = new
     ArrayList<LayoutPane>();
41  private List<BuildPanel> _choiceList = new
     ArrayList<BuildPanel>();
42  private List<BuildPanel> _enableList = new
     ArrayList<BuildPanel>();

44  private List<BuildParameter> _ParamList = new
     ArrayList<BuildParameter>();
45  List<String> _result = new ArrayList<String>();
46  private String inputLine = null;

48  LayoutPane _layout = null;

50  private Tasks _task = null;
51  private Params _params = null;
52
53  private String windowName;
54
55  public LYTReader()
56  {
57      super();
58      initialize();
59      i=0;
60  }
61
62  private void initialize() {
63
64  }

66  public boolean open(String fileName)
67  {
68      //Read the .lyt and .par files
69      try {
70          _in = new BufferedReader(new FileReader(
71              config+"/"+fileName+".lyt")); //Read the .
72          // lyt file named "filename"
73          _task = new Tasks();
74          windowName = fileName; //Where "filename" is
75          // actually the name of the task the end-user
76          // chooses
77
78          _task.readParams(fileName);
79          _task.setTask(fileName);
80          _task.setLevel(fileName, "observation"); //
81          // Task level "observation" (see Appendix A)
82          TreeMap<String, String> paramsHM = null;
83          _params = _task.getTasks(fileName);
84          paramsHM = _params.getParam(); //Read the .
85          // par file default parameter values
```

LYTREADER.JAVA

```

    System.out.println("paramsHM "+paramsHM);
82
    Iterator<String> myTaskIterator = paramsHM.
        keySet().iterator(); //Iterate all default
        parameter values
84
    while(myTaskIterator.hasNext()) {
        System.out.println("PARAMS "+myTaskIterator
            .next());
86
    }

88
    return true;
} catch (IOException e) {
    System.out.println("ERROR Reading Layout file
        ");
    return false;
92
}

94
public void parseLine()
{
96
    //parse each line of the .lyt file
98
    boolean status = true;
try {
100
    while (status){
        inputLine = _in.readLine();
102
        if (inputLine.trim().length() != 0) //Omit
            the blank lines
        {
104
            if (!readAgain(inputLine)) //If there is
                no comment in the line
            {
106
                breakup(inputLine); //Split the line
                    considering the regular expression
                    as one word
                status = false;
108
            }
            else
110
                status = true;
            }
112
        else
            status = true;
114
    }
116
} catch (IOException e) {
    e.printStackTrace();
}
118

120
public void breakup(String inputLine) {
122
    //Create a regular expression and search for it
        through each line of the .lyt file

```

LYTREADER.JAVA

```
124     String val = null;
125     _result.clear(); //Clear the List containing
126         regular expressions
127     String[] lineString = inputLine.trim().split(" "
128 ); //Split words by whitespaces
129     _result.add(lineString[0]); //Add the first
130         word of the parsed line to the List _result
131     System.out.println("Result contents: "+_result)
132         ;
133
134     String regex = "(\\\".*\\\")"; //This is a regular
135         expression recognizing "whatever(characters ,
136         whitespaces , numbers)" as one word
137     Pattern regexp = Pattern.compile(regex);
138     Matcher matcher = regexp.matcher(inputLine);
139     matcher.matches();
140
141     if (matcher.find()) {
142         for (int j = 0; j < matcher.groupCount(); j
143             ++ ) {
144             System.out.print("[ " + matcher.group(j) + " "
145             "]");
146             System.out.println();
147             val = (matcher.group(j));
148             val = (val.substring(1, val.length()-1));
149                 //We trim the " " from the regular
150                     expression found
151             _result.add(val); //Add the trimmed regular
152                 expression to the List _result
153         }
154     } else
155     {
156         if (lineString.length > 1) //If there is a
157             second word in the line we're parsing
158             _result.add(lineString[1]); //then add the
159                 second word to the _result
160
161     }
162
163     matcher.reset(); //reset the matcher in order
164         to look for the next regular expression
165     System.out.println("Let's see what we've got :
166         "+_result);
167
168 }
169
170 private boolean readAgain(String inputLine) {
171     // Read the line again in order to find
172         comments (#) and ignore them
173     boolean status = false;
```

```

158     if (inputLine.trim().length() != 0) //If the
159         line is not blank
160     {
161         //Check for comments
162         String patternComment = "(#.*)";
163         Pattern pattern = Pattern.compile(
164             patternComment);
165         Matcher matcher = pattern.matcher(inputLine
166             );
167         status = matcher.matches(); //Comment
168         line found
169         if (status) {
170             System.out.println("Comment line found -
171                 ignore : " + matcher.group());
172             return true;
173         }
174     }
175
176     return status;
177 }
178
179 public boolean parse()
180 {
181     boolean status = true;
182     parseLine(); //Read the contents of each line
183         of the .lyt file , find the regular
184         expressions (and extract them) , find the
185         comments (and ignore them)
186
187     if (_result.contains("Parameter"))
188         status = parseParameter(); //If the line
189         contains the keyword Parameter then go to
190         the parseParameter method
191     else if (_result.contains("Cards"))
192         status = parsePagedLayout(); //If the line
193         contains the keyword Cards then go to the
194         parsePagedLayout method
195     else if (_result.contains("Frame"))
196         status = parseFrameLayout(); //If the line
197         contains the keyword Frame then go to the
198         parsePagedLayout method
199     else if (_result.contains("Enable"))
200         status = parseEnableLayout(); //If the line
201         contains the keyword Enable then go to the
202         parseEnableLayout method
203     else if (_result.contains("Choice"))
204         status = parseChoiceLayout(); //If the line
205         contains the keyword Choice then go to the
206         parseChoiceLayout method
207     else if (_result.contains("Row"))
208
209 }
```

LYTREADER.JAVA

```
    status = parseRowLayout(); //If the line
    contains the keyword Row then go to the
    parseRowLayout method
192    else if (_result.contains("Column"))
        status = parseColumnLayout(); //If the line
        contains the keyword Column then go to the
        parseColumnLayout method
194    else if (_result.contains("end"))
        status = false; //If the line contains the
        keyword end then break the loop
196    else {
        System.out.println("Layout keyword not
            recognised:" + _result);
198        _ok = false;
        return false;
200    }
        return status;
202}

204 private boolean parseColumnLayout() {
206    if(_layout == null) //If the _layout List is
        null
    {
208        _layout = new BuildPanel("FIRST PANEL"); //Create a new panel
    }
210    System.out.println("Creating Column Panel");

212    BuildPanel panel = new BuildPanel("Column");
        panel.setName("Column");
214    String tooltip = "TOOLTIP"+i;
        panel.setToolTipText(tooltip);
216    _frameList.add(panel); //Add the panel to the
        _framelist List

218    while(parse()) //While searching for keywords
        (Parameter, Cards etc) in each line of
        the .lyt
220    {
        if (_ParamList.size() != 0) //If the
            _ParamList is not empty
        //add a component to the panel, with
            respect to the GridBagConstraints set
            right below
        {
222            GridBagConstraints gridBagConstraints =
                new GridBagConstraints();
                gridBagConstraints.gridx = 0;
```

```

226     gridBagConstraints.gridx =
227         GridBagConstraints.RELATIVE;
228     gridBagConstraints.fill =
229         GridBagConstraints.BOTH;
230     panel.add(getBuildParameter(),
231             gridBagConstraints);
232     System.out.println("NUMBER OF COMPONENTS
233                     "+ panel.getComponentCount());
234 }
235
236     System.out.println("ITEM ADDED!!! ");
237 }
238
239 if (_frameList.size()==1) //If the _frameList
240     List contains one item
241     //add another item with respect to the
242     //GridBagConstraints set right below
243 {
244     GridBagConstraints gridBagConstraints = new
245         GridBagConstraints();
246     gridBagConstraints.gridx = 0;
247     gridBagConstraints.gridy = GridBagConstraints
248         .RELATIVE;
249     gridBagConstraints.fill = GridBagConstraints.
250         BOTH;
251     System.out.println("Adding the frame to the
252                     current panel!!! "+i);
253
254     _layout.add((BuildPanel) getFrame(),
255             gridBagConstraints); //Add the frame to
256     //the _layout with respect to the
257     //gridBagConstraints set right above
258
259 /*
260 * At this point, the size of _frameList
261     must be zero. That means that we have
262     finished with
263     * one nest.
264 */
265 }
266 else
267 {
268     System.out.println("Number of frames inside
269                     _frameList "+_frameList.size());
270     BuildPanel last = (BuildPanel) getFrame();
271     BuildPanel prev = (BuildPanel) getFrame();
272     GridBagConstraints gridBagConstraints = new
273         GridBagConstraints();
274     gridBagConstraints.gridx = GridBagConstraints
275         .RELATIVE;
276     gridBagConstraints.gridy = 0;

```

LYTREADER.JAVA

```
        gridBagConstraints.fill = GridBagConstraints.  
        BOTH;  
260        prev.add(last,gridBagConstraints); //Add  
        the frame named "last" to the frame  
        named "prev" with respect to the  
        gridBagConstraints set right above  
        _frameList.add(prev);  
262    }  
264    System.out.println("Column Done !!!!");  
266    i++;  
268    return _ok;  
270}  
272 private boolean parseRowLayout() {  
273     if(_layout == null) //If the _layout List is  
        null  
274     {  
275         _layout = new BuildPanel("FIRST PANEL"); //  
            Create a new panel  
276     }  
277     System.out.println("Creating Row Panel");  
278     BuildPanel panel = new BuildPanel("Row");  
279     panel.setName("Row");  
280     String tooltip = "TOOLTIP"+i;  
281     panel.setToolTipText(tooltip);  
282     _frameList.add(panel); //Add the panel to the  
        _framelist List  
284  
286  
287     while(parse()) //While searching for keywords  
        (Parameter , Cards etc) in each line of  
        the .lyt  
288     {  
289         if (_ParamList.size() != 0) //If the  
            _ParamList is not empty  
290             //add a components to the panel , with  
                respect to the GridBagConstraints set  
                below  
291         {  
292             GridBagConstraints gridBagConstraints =  
                 new GridBagConstraints();  
293             gridBagConstraints.gridx = 0;  
294             gridBagConstraints.gridy =  
                 GridBagConstraints.RELATIVE;
```

```

296     gridBagConstraints.fill =
297         GridBagConstraints.BOTH;
298     panel.add(getBuildParameter(),
299             gridBagConstraints);
300     System.out.println("NUMBER OF COMPONENTS
301                     "+ panel.getComponentCount());
302 }
303
304     System.out.println("ITEM ADDED!!!");
305 }
306
307 if(_frameList.size()==1) //If the _frameList
308     List contains one item
309     //add another item with respect to the
310     //GridBagConstraints set below
311 {
312     GridBagConstraints gridBagConstraints = new
313         GridBagConstraints();
314     gridBagConstraints.gridx = 0;
315     gridBagConstraints.gridy = GridBagConstraints
316         .RELATIVE;
317     gridBagConstraints.fill = GridBagConstraints.
318         BOTH;
319     System.out.println("Adding the frame to the
320                     current panel!!! "+i);
321
322     _layout.add((BuildPanel)getFrame(),
323                 gridBagConstraints);
324
325 /*
326  * At this point, the size of _frameList
327  * must be zero. That means that we have
328  * finished with
329  * one nest.
330  */
331 }
332 else
333 {
334     System.out.println("Number of frames inside
335                     _frameList "+_frameList.size());
336     LayoutPane last = getFrame();
337     LayoutPane prev = getFrame();
338     GridBagConstraints gridBagConstraints = new
339         GridBagConstraints();
340     gridBagConstraints.gridx = 0;
341     gridBagConstraints.gridy = GridBagConstraints
342         .RELATIVE;
343     gridBagConstraints.fill = GridBagConstraints.
344         BOTH;
345     prev.add((BuildPanel) last,
346             gridBagConstraints); //Add the frame

```

LYTREADER.JAVA

```
            named "last" to the frame named "prev"
            with respect to the GridBagConstraints
            set right above
            _frameList.add(prev);
        }

        System.out.println("Frame Done !!!!");

        i++;
        return _ok;
    }

    private boolean parseChoiceLayout() {
        if(_layout == null) //If the _layout List is
                           null
        {
            _layout = new BuildPanel("FIRST PANEL"); //Create a new panel
        }

        System.out.println("New Choice Panel create !!")
        ;

        BuildPanel panel = new BuildPanel(_result.get
                                         (1)); //Name the panel after the word
                                         situated next to each "Choice" keyword in
                                         the .lyt line
        _choiceList.add(panel); //Add the panel to the
                               _choicelist List
        _frameList.add(panel); //Add the panel to the
                               _framelist List

        GridBagConstraints gridBagConstraints = new
                                              GridBagConstraints();
        gridBagConstraints.fill = GridBagConstraints.
                                   VERTICAL;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.gridx = 0;

        //A JComboBox represents a drop down menu
        panel.add(panel.getJComboBox(),
                  gridBagConstraints); //add each JComboBox
                                         created, to the current panel
        final BuildComboBox combo = panel.getJComboBox()
                                         (); //get each JComboBox that's created

        combo.addItemListener(new ItemListener() {
            //create a listener which applies to all the
             JComboBoxs' created
```

```

//and reports an action every time a choice
  is clicked from the user
364  public void itemStateChanged(ItemEvent evt) {
      System.out.println("ItemChanged "+evt);

366    BuildPanel panel = getChoice(combo.
      getSelectedItem().toString()); //call
      the getChoice method, get the selected
      JComboBox's item and return a String
      containing the item's name

368

370    List<BuildPanel> bpList = panel.
      getFrameStuff(); //Create a List with
      the frame components

372    for( int i = 0; i<bpList.size(); i++){
374        System.out.println("COMPONENT!!!");

376        if (panel.getJComboBox().getSelectedItem()
          ().toString().equals(bpList.get(i).
          getName())) //If the name of the
          JComboBox component equals the name of
          one of the bpList names
          //which means that one of the JComboBox
          items is chosen by the user
378        {
379            System.out.println("binSize ....");
380            bpList.get(i).setVisible(true); //Set
              the items of the JComboBox chosen as
              visible
381        }
382        else
383        {
384            System.out.println("imageSize ....");
385            bpList.get(i).setVisible(false); //Set
              the items of the JComboBox chosen as
              hidden
386        }
387    }
388  );
389
390  while(parseChoice())
391  {
392      System.out.println("Parse Choice Page
          finished !!!");
393  }
394  System.out.println("Page Panel created !!!");
395
396

```

LYTREADER.JAVA

```
398     if (_frameList.size() == 1) //If the _frameList
        List contains one item
        //add another item with respect to the
        GridBagConstraints set below
    {
400     GridBagConstraints gridBagConstraints1 = new
        GridBagConstraints();
        gridBagConstraints1.gridx = 0;
402     gridBagConstraints1.gridy =
        GridBagConstraints.RELATIVE;
        gridBagConstraints1.fill = GridBagConstraints
        .BOTH;
404     System.out.println("Adding the frame to the
        current panel!!! "+i);

406     _layout.add((BuildPanel) getFrame(),
        gridBagConstraints1);
    /*
408     * At this point, the size of _frameList
        must be zero. That means that we have
        finished with
        * one nest.
    */
410 }
412 else
{
414     System.out.println("Number of frames inside
        _frameList "+_frameList.size());
416     LayoutPane last = getFrame();
        LayoutPane prev = getFrame();
        GridBagConstraints gridBagConstraints2 = new
            GridBagConstraints();
418     gridBagConstraints2.gridx = 0;
        gridBagConstraints2.gridy =
            GridBagConstraints.RELATIVE;
420     gridBagConstraints2.fill = GridBagConstraints
            .BOTH;
        prev.add((BuildPanel) last,
            gridBagConstraints2); //Add the frame
            named "last" to the frame named "prev"
            with respect to the gridBagConstraints
            set right above
422     _frameList.add(prev);

424 }
426 /*
427 * At this point, the size of _frameList must
        be zero. That means that we have finished
        with
        * one nest.
    */
428 }
```

```

430     return _ok;
431 }
432 private boolean parseChoice()
433 {
434     boolean status = true;
435     parseLine(); //Read the contents of each line
436         of the .lyt file , find the regular
437         expressions , find the comments (and ignore
438         them)
439
440     if (_result.get(0).contains("Page"))
441     {
442         System.out.println("Adding an Item to the
443             ComboBox "+_result);
444
445         BuildPanel panel = (BuildPanel) getFrame();
446         panel.getJComboBox().addItem(_result.get(1));
447             //Add the second word of the line to the
448             JComboBox and then to the panel
449         _frameList.add(panel); //Add the panel to the
450             List _frameList
451
452         BuildPanel frame = new BuildPanel(_result.get
453             (1)); //Create a frame named after the
454             second word of the line that is read
455             _frameList.add(frame); //Add the frame to
456             the List _frameList
457
458         while (parse()) {
459             //While searching for keywords like
460             Parameter , Cards etc
461             if (_ParamList.size() != 0) { //If the List
462                 _ParamList is not empty
463                 //add a frame with respect to the
464                 GridBagConstraints listed right below
465                 GridBagConstraints gridBagConstraints =
466                     new GridBagConstraints();
467                     gridBagConstraints.gridx = 0;
468                     gridBagConstraints.gridy =
469                         GridBagConstraints.RELATIVE;
470                     gridBagConstraints.fill =
471                         GridBagConstraints.BOTH;
472                     BuildParameter bp = getBuildParameter();
473                     frame.setLayout(bp);
474                     frame.add(bp, gridBagConstraints); //add
475                         the parameters to the frame with
476                         respect to the GridBagConstraints
477                     System.out.println("NUMBER OF COMPONENTS"
478                         "
```

LYTREADER.JAVA

```
        + frame.getComponentCount());
462        System.out.println("ITEM ADDED!!!");
464    }
466    System.out.println("Number of frames inside
467        _frameList "+_frameList.size());
468    LayoutPane last = getFrame();
469    LayoutPane prev = getFrame();
470    GridBagConstraints gridBagConstraints2 = new
471        GridBagConstraints();
472    gridBagConstraints2.gridx = 0;
473    gridBagConstraints2.gridy =
474        GridBagConstraints.RELATIVE;
475    gridBagConstraints2.fill = GridBagConstraints
476        .BOTH;

477    ((BuildPanel)prev).addFrame((BuildPanel) last
478        , gridBagConstraints2); //Add the frame
479        named "last" to the frame named "prev"
480        with respect to the gridBagConstraints set
481        right above
482    _frameList.add(prev); //Add both frames to
483        the List _frameList
484    i++;
485    status = _ok;
486}
487else if(_result.get(o).contains("end")) //If
488    the line contains the word "end", break the
489    loop
490{
491    System.out.println("Type "+_result.get(o));
492    status = false;
493}
494
495private boolean parseEnableLayout() {
496    if(_layout == null) //If the _layout List is
497        null
498    {
499        _layout = new BuildPanel("FIRST PANEL"); //
500            Create a new panel
501    }
502
503    boolean status = true;
504    System.out.println("Creating Enabled Panel");
505}
```

LYTREADER.JAVA

```

498  if (_params.getType(_result.get(1)).equals("bool")) //If the second word of the _result
      List contains a boolean parameter
{
500    final JCheckBox checkBox = new JCheckBox(
      _result.get(1), true); //Create a
      JCheckBox named after the second word of
      the line
    checkBox.setFocusPainted(false);
    checkBox.setOpaque(true);

504    BuildPanel panel = new BuildPanel(_result.get
      (1)); //Name the panel after the word
      situated next to each "Enable" keyword in
      the .lyt line
    panel.setName(_result.get(1));
    String tooltip = "TOOLTIP"+i;
    panel.setToolTipText(tooltip);

508    _enableList.add(panel); //Add the panel to
      the List _enableList
510    _frameList.add(panel); //Add the panel to the
      List _frameList

512    while(parse())
{
514      System.out.println("enable parameter!!!");

516      if (_ParamList.size() != 0) //If the
        _ParamList is not empty
          //add a components to the panel, with
          respect to the GridBagConstraints
          set below
{
518        GridBagConstraints gridBagConstraints =
          new GridBagConstraints();
519        gridBagConstraints.gridx = 0;
520        gridBagConstraints.gridy =
          GridBagConstraints.RELATIVE;
521        gridBagConstraints.fill =
          GridBagConstraints.BOTH;
522        BuildParameter bp = getBuildParameter();
523        panel.setLayout(bp);
524        panel.add(bp,gridBagConstraints); //Add
          the bp to the panel with respect to
          the GridBagConstraints
526      }
527    }
528    System.out.println("NUMBER OF COMPONENTS "+
      panel.getComponentCount());

```

LYTREADER.JAVA

```
    //Set the look and feel of the border of each  
    components on the layout  
530  
531  
532     ComponentTitledBorder componentBorder =  
533         new ComponentTitledBorder(checkBox, panel  
534             , BorderFactory.createEtchedBorder());  
535     checkBox.addActionListener(new ActionListener  
536         () { //Add a listener to the checkbox  
537             public void actionPerformed(ActionEvent e){  
538                 boolean enable = checkBox.isSelected();  
539  
540                 System.out.println("Title name "+  
541                     checkBox.getText());  
542                 BuildPanel panel = getEnable(checkBox.  
543                     getText());  
544                 List<BuildParameter> bpList = panel.  
545                     getLayoutStuff();  
546                 for(int i = 0; i<bpList.size(); i++){  
547                     System.out.println("COMPONENT!!!");  
548                     bpList.get(i).setDisable(enable);  
549                 }  
550             }  
551         });  
552     panel.setBorder(componentBorder);  
553  
554     /*  
555      * Move the last frame into the previous one  
556      */  
557  
558     if(_frameList.size()==1)  
559     {  
560         GridBagConstraints gridBagConstraints = new  
561             GridBagConstraints();  
562         gridBagConstraints.gridx = 0;  
563         gridBagConstraints.gridy =  
564             GridBagConstraints.RELATIVE;  
565         gridBagConstraints.fill =  
566             GridBagConstraints.BOTH;  
567         System.out.println("Adding the frame to  
568             the current panel!!! "+i);  
569  
570         _layout.add((BuildPanel) getFrame(),  
571             gridBagConstraints);  
572         /*  
573          * At this point, the size of _frameList  
574          must be zero. That means that we have  
575          finished with  
576          * one nest.  
577          */  
578     }  
579     else
```

```

568     {
569         System.out.println("Number of frames
570             inside _frameList "+_frameList.size())
571             ;
572         LayoutPane last = getFrame();
573         LayoutPane prev = getFrame();
574         GridBagConstraints gridBagConstraints = new
575             GridBagConstraints();
576         gridBagConstraints.gridx = o; /*This helps
577             with the frame width problem*/
578         gridBagConstraints.gridy =
579             GridBagConstraints.RELATIVE;
580         gridBagConstraints.fill =
581             GridBagConstraints.BOTH;
582         prev.add((BuildPanel) last,
583             gridBagConstraints); //Add the frame
584             named "last" to the frame named "prev"
585             with respect to the
586             gridBagConstraints set right above
587             _frameList.add(prev);
588     }
589
590     System.out.println("Enable Done!!!!");
591
592     i++;
593     status = _ok;
594 }
595 else
596 {
597     System.out.println("Type "+_result.get(o));
598     status = false;
599 }
600 return status;
601 }

602 private boolean parseFrameLayout() {
603     if(_layout == null) //If the _layout is empty
604     {
605         _layout = new BuildPanel("FIRST PANEL"); //
606             Add a new panel
607     }
608     System.out.println("Creating Frame Panel");
609
610     BuildPanel panel = new BuildPanel(_result.get
611         (1));
612
613     panel.setName(_result.get(1)); //Get the second
614         string of _result and set it as the name of
615         the panel
616     String tooltip = "TOOLTIP"+i;

```

LYTREADER.JAVA

```
606         panel.setToolTipText(tooltip);
607
608         _frameList.add(panel); //Add panel to
609             _framelist
610
611         while(parse())
612     {
613             if (_ParamList.size() != 0) //If _ParamList
614                 is empty
615             { //Set new GridBagConstraints
616                 GridBagConstraints gridBagConstraints =
617                     new GridBagConstraints();
618                 gridBagConstraints.gridx = o;
619                 gridBagConstraints.gridy =
620                     GridBagConstraints.RELATIVE;
621                 gridBagConstraints.fill =
622                     GridBagConstraints.BOTH;
623                 panel.add(getBuildParameter(),
624                     gridBagConstraints); //Add the output
625                     of the getBuildParameter method to
626                     panel with respect to the
627                     gridBagConstraints set right above
628                     System.out.println("NUMBER OF COMPONENTS
629                         "+ panel.getComponentCount());
630             }
631
632             System.out.println("PARAMETER ADDED!!! ");
633         }
634         System.out.println("Frame END rechead");
635
636         if(_frameList.size()==1) //If _frameList is
637             not empty
638         {
639             //Set new GridBagConstraints
640             GridBagConstraints gridBagConstraints = new
641                 GridBagConstraints();
642             gridBagConstraints.gridx = o;
643             gridBagConstraints.gridy = GridBagConstraints
644                 .RELATIVE;
645             gridBagConstraints.fill = GridBagConstraints.
646                 BOTH;
647             System.out.println("Adding the frame to the
648                 current panel!!! "+i);
649
650             _layout.add((BuildPanel) getFrame(),
651                 gridBagConstraints); //Add frame to
652                 _layout with respect to the
653                 gridBagConstraints set right above
654             /*
655             * At this point, the size of _frameList
656             must be zero. That means that we have
657             finished with
```

```

636         * one nest.
637         */
638     }
639     else
640     {
641         System.out.println("Number of frames inside
642                         _frameList "+_frameList.size());
643         LayoutPane last = getFrame();
644         LayoutPane prev = getFrame();
645         GridBagConstraints gridBagConstraints = new
646             GridBagConstraints();
647         gridBagConstraints.gridx = GridBagConstraints.
648             RELATIVE; //This keeps all tabs one next
649             to the other
650         gridBagConstraints.gridy = GridBagConstraints.
651             RELATIVE;
652         gridBagConstraints.fill = GridBagConstraints.
653             BOTH;
654         prev.add((BuildPanel) last,
655             gridBagConstraints); //Add last panel to
656             previous with respect to the
657             gridBagConstraints set right above
658             _frameList.add(prev);
659     }
660     System.out.println("Frame Done !!!!");
661     i++;
662     return _ok;
663 }
664
665 private boolean parsePagedLayout() {
666     System.out.println("New Card Panel create!!!");
667
668     if (_layout == null) //If _layout is empty
669     {
670         _layout = new BuildTabPage("FIRST PANEL"); ///
671             Add a new Tab
672     }
673
674     while (parsePage())
675     {
676         System.out.println("ParsePage false !!!!");
677     }
678
679     System.out.println("Page Panel created !!!");
680
681     return _ok;
682 }
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
```

LYTREADER.JAVA

```
678     private boolean parsePage()
679     {
680         boolean status = true;
681
682         parseLine();
683
684         if (_result.get(0).contains("Page")) //If the
685             first string of _result contains the word "
686             Page"
687         {
688             System.out.println("Creating Page Panel");
689
690             LayoutPane panel = null;
691             if (_pageFlag == true)
692             {
693                 panel = new BuildTabPage(_result.get(1));
694                 //Add a Tab to the panel named as the
695                 second string of _result
696             }
697             else
698             {
699                 panel = new BuildPanel(_result.get(1)); //
700                 Add a panel named as the second string
701                 of _result
702             }
703
704             panel.setName(_result.get(1)); //Set the name
705             of the panel as the second string of
706             _result
707             String text = "TOOLTIP"+i;
708             panel.setToolTipText(text);
709
710             _frameList.add(panel); //Add panel to
711             _framelist
712
713             while(parse())
714             {
715                 if (_ParamList.size() != 0) //If
716                     _ParamList is empty
717                 { //Set new GridBagConstraints
718                     GridBagConstraints gridBagConstraints =
719                         new GridBagConstraints();
720                     gridBagConstraints.gridx = 0;
721                     gridBagConstraints.gridy =
722                         GridBagConstraints.RELATIVE;
723                     gridBagConstraints.fill =
724                         GridBagConstraints.BOTH;
725                     panel.add(getBuildParameter(),
726                         gridBagConstraints); //Add the
727                         output of the getBuildParameter
728                         method to the panel with respect to
```

```

    the GridBagConstraints set right
    above
712     System.out.println("NUMBER OF
    COMPONENTS "+ panel.
        getComponentCount());
    }

714     System.out.println("ITEM ADDED!!! ");
716 }

718     if (_frameList.size()==1) //If _frameList is
        not empty
720     { //Set new GridBagConstraints
721         GridBagConstraints gridBagConstraints = new
            GridBagConstraints();
722         gridBagConstraints.gridx = 0;
723         gridBagConstraints.gridy =
            GridBagConstraints.RELATIVE;
724         gridBagConstraints.fill =
            GridBagConstraints.BOTH;
725         System.out.println("Adding the page frame
            to the current panel!!! "+_layout.
                getNumberofItems());
726         LayoutPane kk = getFrame();
727         if(kk instanceof BuildTabPage) //If kk is
            a Tab
728             ((BuildTabPage) _layout).add((
                BuildTabPage) kk,gridBagConstraints)
                ; //Add kk as Tab to _layout with
                    respect to the gridBagConstraints
                    set right above
729         else
730             _layout.add((BuildPanel) kk,
                gridBagConstraints); //Add kk as
                    panel to _layout with respect to the
                    gridBagConstraints set right above
731         /*
732             * At this point, the size of _frameList
733             must be zero. That means that we have
734             finished with
735             * one nest.
736             */
737         System.out.println("Adding the page frame
            to the current panel!!! "+_layout.
                getNumberofItems());
    }
738     else
    {
        /*
739             * Move the last frame into the previous
740             one

```

LYTREADER.JAVA

```
740         */
741         System.out.println("Number of frames
742             inside _frameList "+_frameList.size())
743             ;
744             LayoutPane last = getFrame();
745             LayoutPane prev = getFrame();
746             GridBagConstraints gridBagConstraints = new
747                 GridBagConstraints();
748                 gridBagConstraints.gridx = 0;
749                 gridBagConstraints.gridy =
750                     GridBagConstraints.RELATIVE;
751                 gridBagConstraints.fill =
752                     GridBagConstraints.BOTH;
753
754             if(last instanceof BuildTabPage) //If
755                 last is a Tab
756                 ((BuildTabPage) prev).add((BuildTabPage)
757                     last,gridBagConstraints); //Add
758                     last as Tab to prev with respect to
759                     the gridBagConstraints set right
760                     above
761             else
762                 ((BuildPanel) prev).add((BuildPanel)
763                     last,gridBagConstraints); //Add last
764                     as panel to prev with respect to
765                     the gridBagConstraints set right
766                     above
767             _frameList.add(prev); //Add prev to
768             _frameList
769
770     }
771
772     System.out.println("Page Done !!!!");
773
774     i++;
775     status = _ok;
776 }
777
778 else if(_result.get(0).contains("end")) //If
779     the first string of _result is "end"
780 {
781     System.out.println("Type "+_result.get(0));
782     status = false; //Set boolean status to
783         false
784
785 }
786
787 return status;
788 }
789
790 private boolean parseParameter() {
791 }
```

```

        BuildParameter bp = new BuildParameter(_params.
    getType(_result.get(1))); //The
    BuildParameter class will get the second
    string of _result and pass it to _params as
    the type of parameter
774
    bp.getLabel().setText(_result.get(1)); //The
    getLabel() method of BuildParameter class
    will set the second string of _result as
    label for bp
776
    bp.getTextField().setText(_params.getParam().
    get(_result.get(1))); //The getTextField()
    method of BuildParameter class will fetch
    the second string of _result, pass it to the
   getParam() method which will withdraw this
    parameter from the _params and pass it to bp
    as the default parameter value
778
    _ParamList.add(bp); /Add bp to _ParamList
    System.out.println("SimpleParameter DONE");
780
    return _ok;
782 }

784 private BuildParameter getBuildParameter() {
786
    System.out.println("Parameter List Size: " +
        _ParamList.size());
    return _ParamList.remove(_ParamList.size()-1);
    //Return the size of _ParamList minus one
788 }

790 private LayoutPane getFrame()
{
792
    return _frameList.remove(_frameList.size()-1);
    //Return the size of _frameList minus one
}
794
private BuildPanel getChoice(String item) { //
796
    BuildPanel bp = null; //Set bp empty
798
    for (int n = 0 ; n < _choiceList.size() ; n++)
        //For all the items of _choiceList
    {
800
        for (int m = 0 ; m < _choiceList.get(n).
            getItemCount(); m++) //For
            all the JComboboxes included in
            _choiceList
    {
802
        if( _choiceList.get(n).getJComboBox().
            getItemAt(m).toString().equals(item)) //
    }
}

```

LYTREADER.JAVA

```
    If the item no.n equals to the "item"
    string
    bp = _choiceList.get(n); //Get the no.n
    item of _choiceList and pass it to bp
804    }
805}
806    return bp;
807}
808
809 private BuildPanel getEnable(String name) {
810
811     BuildPanel bp = null;
812     for (int n = 0 ; n < _enableList.size() ; n++)
813         //For all the items of _enableList
814     {
815         if( _enableList.get(n).getName() .equals(name)
816             ) //If the name of item no.n equals the
817             string "name"
818             bp = _enableList.get(n); //Get item no.n
819             from _enableList and pass it to bp
820     }
821     return bp;
822 }
823
824 public void show() {
825
826     _frame = new JFrame(); //Create a new JFrame
827     _frame.setTitle(windowName); //Set the title as
828     the string "windowName"
829     _frame.setLocationRelativeTo(null);
830
831     JFrame.setDefaultLookAndFeelDecorated(true); //
832     Set the looks of the JFrame
833
834     BuildPanel buttonPanel = new BuildPanel(null);
835     //Add a new frame to JFrame, which will
836     contain the Apply and Cancel buttons
837
838     System.out.println("Add the _layout to the main
839     frame");
840     //Set the gridBagConstraints for the main frame
841     GridBagConstraints gridBagConstraints = new
842         GridBagConstraints();
843     gridBagConstraints.gridx = 0;
844     gridBagConstraints.gridy = 0;
845     gridBagConstraints.fill = GridBagConstraints.
846         BOTH;
847
848     //Set the gridBagConstraints for the Apply
849     button
```

```

838     GridBagConstraints gridBagConstraintsA = new
839         GridBagConstraints();
840     gridBagConstraintsA.gridx = 0;
841     gridBagConstraintsA.gridy = 1;
842     gridBagConstraints.fill = GridBagConstraints.
843         BOTH;
842     buttonPanel.add(getApplyButton() ,
843         gridBagConstraintsA);

844     //Set the gridBagConstraints for the Cancel
845     //button
846     GridBagConstraints gridBagConstraintsC = new
847         GridBagConstraints();
846     gridBagConstraintsC.gridx = 1;
847     gridBagConstraintsC.gridy = 1;
848     gridBagConstraints.fill = GridBagConstraints.
849         BOTH;
849     buttonPanel.add(getCancelButton() ,
850         gridBagConstraintsC);

850     //Set the gridBagConstraints for the main frame
852     GridBagConstraints gridBagConstraintsB = new
853         GridBagConstraints();
853     gridBagConstraintsB.gridx = 0;
854     gridBagConstraintsB.gridy = 1;
854     gridBagConstraintsB.fill = GridBagConstraints.
855         BOTH;

856     //Add a scrollbar to the _layout contained in
857     //the main frame
858     JScrollPane scrollpane = new JScrollPane();
859     scrollpane.setViewportView((Component) _layout)
860         ;
860     _frame.add(scrollpane,BorderLayout.CENTER);
861     _frame.add((Component) buttonPanel,BorderLayout
862         .SOUTH);
862     _frame.pack();
863     _frame.setVisible(true);

864     //—TEST— Extract the layout objects and "read"
865     //      " them
866     System.out.println("Number of objects: "+
867         _layout.getNumberOfItems());
868     _layout.printParameters();
868     //—TEST— Extract the layout objects and "read"
869     //      " them
870 }

871 private JButton getJButton() {
872     if (jButton == null) { //If there is no jButton
873         created

```

LYTREADER.JAVA

```
874     jButton = new JButton(); //Create one
875     jButton.setText("Apply"); //Set the text of
876     it as "Apply"
877     jButton.setSize(jButton.getSize()); //
878     Automatically resize the jButton depending
879     on the text
880     jButton.addActionListener(new java.awt.event.
881     ActionListener() { //Add a listener
882         public void actionPerformed(java.awt.event.
883             ActionEvent e) {
884             System.out.println("actionPerformed ()");
885             _frame.dispose(); //Close frame whenever
886             the button is pressed
887         }
888     });
889 }
890
891 /**
892 * This method initializes jButton1
893 */
894 private JButton getJButton1() {
895     if (jButton1 == null) { //If there is no
896         jButton1 created
897         jButton1 = new JButton(); //Create one
898         jButton1.setText("Cancel"); //Set the text of
899         it as "Cancel"
900         jButton1.setSize(jButton1.getSize()); //
901         Automatically resize the jButton depending
902         on the text
903         jButton1.addActionListener(new java.awt.event.
904             ActionListener() { //Add a listener
905                 public void actionPerformed(java.awt.event.
906                     ActionEvent e) {
907                     System.out.println("actionPerformed ()");
908                     _frame.dispose(); //Close frame whenever
909                     the button is pressed
910                 }
911             });
912 }
913
914 /**
915 * This method initializes jButton
916 */
917 private JButton getJButton() {
918     if (jButton == null) { //If there is no
919         jButton created
920         jButton = new JButton(); //Create one
921         jButton.setText("OK"); //Set the text of
922         it as "OK"
923         jButton.setSize(jButton.getSize()); //
924         Automatically resize the jButton depending
925         on the text
926         jButton.addActionListener(new java.awt.event.
927             ActionListener() { //Add a listener
928                 public void actionPerformed(java.awt.event.
929                     ActionEvent e) {
930                     System.out.println("actionPerformed ()");
931                     _frame.dispose(); //Close frame whenever
932                     the button is pressed
933                 }
934             });
935 }
936
937 /**
938 * This method initializes jButton2
939 */
940 private JButton getJButton2() {
941     if (jButton2 == null) { //If there is no
942         jButton2 created
943         jButton2 = new JButton(); //Create one
944         jButton2.setText("Cancel"); //Set the text of
945         it as "Cancel"
946         jButton2.setSize(jButton2.getSize()); //
947         Automatically resize the jButton depending
948         on the text
949         jButton2.addActionListener(new java.awt.event.
950             ActionListener() { //Add a listener
951                 public void actionPerformed(java.awt.event.
952                     ActionEvent e) {
953                     System.out.println("actionPerformed ()");
954                     _frame.dispose(); //Close frame whenever
955                     the button is pressed
956                 }
957             });
958 }
```

LYTREADER.JAVA

```
910     return getJButton1(); //Return the Cancel  
911         button  
912     }  
913  
914     public void close() { //Close the class  
915         try {  
916             _in.close(); //Close the .lyt filereader  
917         } catch (IOException e) {  
918             e.printStackTrace();  
919         }  
920     }
```


G

BUILDPANEL CLASS

```
1 package layoutCreator;
2
3 import java.awt.Component;
4 import java.awt.Container;
5 import java.awt.GridBagConstraints;
6 import java.awt.GridBagLayout;
7 import javax.swing.JPanel;
8 import javax.swing.BorderFactory;
9 import javax.swing.JComboBox;
10 import java.util.ArrayList;
11 import java.util.List;
12 import layoutCreator.BuildComboBox;
13 import javax.swing.border.TitledBorder;
14
15 public class BuildPanel extends JPanel implements
16     LayoutPane {
17
18     private static final long serialVersionUID = 1L;
19     private BuildComboBox jComboBox;
20     private List<BuildParameter> _addLayout = new
21         ArrayList<BuildParameter>();
22     private List<BuildPanel> _addFrame = new
23         ArrayList<BuildPanel>();
24     private String buildPanelName;
25     private BuildCheckBox jCheckBox;
26
27     /**
28      * This is the default constructor
29      */
30     public BuildPanel(String panelName) {
31         super();
32         initialize(panelName);
33     }
34
35     /**
36      * This method initializes this
37      */
38     private void initialize(String panelName) {
```

BUILDPANEL CLASS

```
36     this.setLayout(new GridBagLayout()); //Set
      layout to grid bag layout
37     this.setBorder(BorderFactory.createTitledBorder
      (null, panelName, TitledBorder.LEFT,
      TitledBorder.TOP, null, null)); //Set the
      border preferences
38     if (panelName != null) //If panelname is not
      empty
      this.setName(panelName); //Set the name of
      the panel according to panelname
40     buildPanelName = getName(); //Get the name of
      the panel
41     this.pack();
42 }

44     private void pack() {
45
46 }
47
48     public JPanel getPanel()
49 {
50     return this; //Return panel
51 }
52

54     public void addLayout(BuildParameter bp)
55 {
56     _addLayout.add(bp); //Add bp to _addLayout
57 }
58
59     public List<BuildParameter> getLayoutStuff()
60 {
61     return _addLayout;
62 }
63

64     public BuildComboBox getJComboBox() {
65         if(jComboBox == null) { //If jComboBox doesn't
            exist
66             jComboBox = new BuildComboBox(); //Build a
            new combobox
67         }
68         return jComboBox;
69     }
70
71     public BuildCheckBox getJCheckBox() {
72         if(jCheckBox == null) { //If jCheckBox doesn't
            exist
73             jCheckBox = new BuildCheckBox(); //Build a
            new checkbox
74         }
75         return jCheckBox;
76 }
```

BUILDPANEL CLASS

BUILDPANEL CLASS

```
112     Component cp = this.getComponent(i); //Get  
113     component no.i  
114     if (cp instanceof BuildParameter) //If  
115         component is a parameter  
116         ((BuildParameter) cp).getParameter()  
117             .printContent(); //Fetch the parameter  
118             and print the content of it  
119     else  
120     {  
121         System.out.println("Label name: " + ((  
122             Container) cp).getName());  
123         ((LayoutPane) cp).printParameters(); //  
124             Print the label of the parameter  
125             /*JComboBoxes and CheckBoxes are not read.  
126                 FIX IT*/  
127     }  
128 }
```

H

BUILDPARAMETER CLASS

```
1 package layoutCreator;
2
3 import javax.swing.JCheckBox;
4 import javax.swing.JPanel;
5 import javax.swing.JLabel;
6 import javax.swing.JTextField;
7 import javax.swing.SwingConstants;
8 import layoutCreator.*;
9 import java.awt.GridLayout;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 public class BuildParameter extends JPanel {
14
15     private static final long serialVersionUID = 1L;
16     private String _type = null; //Type of parameter
17     private JLabel jLabel = null; //The label of each
18         parameter
19     private JTextField jTextField = null; //The value
20         of each parameter
21     private JCheckBox jCheckBox = null; //Boolean
22         value of each checkbox
23
24     /**
25      * This is the default constructor
26      */
27     public BuildParameter(String type) {
28         super();
29         _type = type;
30         initialize();
31     }
32
33     /**
34      * This method initializes this
35      */
36     private void initialize() {
37         GridLayout gridLayout = new GridLayout(); //
38             Choose the layout
```

BUILDPARAMETER CLASS

```
gridLayout.setRows(1); //Set the number of rows  
to 1  
36   jLabel = new JLabel(); //Create a label for the  
parameter  
38   jLabel.setText("JLabel");  
39   jLabel.setHorizontalTextPosition(SwingConstants.  
.CENTER); //Set the text position  
40   jLabel.setHorizontalAlignment(SwingConstants.  
.CENTER); //Set the text alignment  
41   jLabel.getSize(); //Get the label text size  
42   jLabel.setSize(getSize()); //Set the label size  
equal to the length of the label text  
43  
44   this.setLayout(gridLayout); //Set the layout  
this.pack();  
45   this.add(jLabel, null);  
46   if (_type.equals("bool")) //If the _type equals  
to a boolean  
47       this.add(getJCheckBox(), null); //Create a  
checkbox  
48   else  
49       this.add(getJTextField(), null); //Create a  
textfield  
50 }  
51  
52 private void pack() {  
53     return;  
54 }  
55  
56 /**  
 * This method initializes jTextField  
 */  
57  
58 public JTextField getJTextField() {  
59     if (jTextField == null) { //If the jTextField  
is null  
60         jTextField = new JTextField(); //Create a new  
textfield  
61         jTextField.getSize(); //Get thetextfield  
text size  
62         jTextField.setSize(getSize()); //Set the  
textfield size equal to the text  
63         jTextField.setHorizontalAlignment(JTextField.  
.LEFT); //Align thetextfield  
64     }  
65     return jTextField;  
66 }  
67  
68 public JCheckBox getJCheckBox() {  
69     if (jCheckBox == null) { //If the jCheckBox is  
null  
70 }
```

BUILDPARAMETER CLASS

```

jCheckBox = new JCheckBox(); //Create a new
checkbox
72   jCheckBox.setHorizontalAlignment(JCheckBox.
      LEFT); //Align the checkbox
    jCheckBox.getSize(); //Get the size of it
74   jCheckBox.setSize(getSize()); //Set the size
      equal to it
}
76   return jCheckBox;
}
78
public JLabel getLabel()
80 {
  return this.jLabel; //Return the current label
}
82
84 public BuildParameter getParameter()
{
  return this; //Return the parameter
}
86
88 public void setDisable(boolean b) {
  this.jLabel.setEnabled(b); //Enable the label
    so it can be visible
  if(getJCheckBox() != null) //If the
    getJCheckBox is not empty
    getJCheckBox().setEnabled(b); //Enable the
      checkbox
  if(getJTextField() != null) //If the
    getJTextField is not empty
    getJTextField().setEnabled(b); //Enable the
      textfield and make it editable
}
90
92
94
96 public String toString(){
  return _type;
}
98
100 public void printContent()
{
  System.out.println("Parameter " + jLabel.
    getText() + " Value: " + jTextField.getText
    ());
}
102
104
106 }

```


I

BUILDTAB CLASS

```
1 package layoutCreator;
2
3 import java.awt.GridBagLayout;
4 import javax.swing.JPanel;
5 import javax.swing.JTabbedPane;
6 import java.awt.GridBagConstraints;
7
8 public class BuildTab extends JPanel {
9
10    private static final long serialVersionUID = 1L;
11    private JTabbedPane jTabbedPane = null;
12
13    /**
14     * This is the default constructor
15     */
16    public BuildTab(String name) {
17        super();
18        initialize(name);
19    }
20
21    /**
22     * This method initializes this
23     * @return void
24     */
25    private void initialize(String name) {
26        GridBagConstraints gridBagConstraints = new
27            GridBagConstraints(); //Use the
28            GridBagConstraints layout class
29        gridBagConstraints.fill = GridBagConstraints.
30            BOTH;
31        gridBagConstraints.gridx = 0;
32        gridBagConstraints.weightx = 1.0;
33        gridBagConstraints.weighty = 1.0;
34        gridBagConstraints.gridy = 0;
35        this.setName(name); //Set the name of the tab
36        this.pack();
37        this.setLayout(new GridBagLayout()); //Set the
38            tab layout constraints
```

BUILDTAB CLASS

```
      this . add( getJTabbedPane() ,  gridBagConstraints);
      //Add tabs to layout with respect to
      gridBagconstraints
36    }

38  private void pack() {
39    return;
40  }

42 /**
43 * This method initializes jTabbedPane
44 * @return javax.swing.JTabbedPane
45 */
46 public JTabbedPane getJTabbedPane() {
47   if (jTabbedPane == null) { //If jTabbedPane is
48     null
49     jTabbedPane = new JTabbedPane(); //Add a new
50     jTabbedPane
51   }
52   return jTabbedPane;
53 }
```

J

BUILDTABPAGE CLASS

```
1 package layoutCreator;  
2  
3 import javax.swing.JTabbedPane;  
4 import java.awt.Component;  
5 import java.awt.Container;  
6 import java.awt.GridBagConstraints;  
7  
8 public class BuildTabPage extends JTabbedPane  
9     implements LayoutPane {  
10  
11     private static final long serialVersionUID = 1L;  
12     private String tabPageName;  
13     /**  
14      * This is the default constructor  
15      */  
16     public BuildTabPage(String name) {  
17         super();  
18         initialize(name);  
19     }  
20  
21     /**  
22      * This method initializes this  
23      */  
24     private void initialize(String name) {  
25         this.pack();  
26         this.setName(name); //Set the tab page name  
27         tabPageName = getName(); //Get the tabPageName  
28         string  
29     }  
30  
31     private void pack() {  
32         return;  
33     }  
34  
35     public JTabbedPane getTabPane()  
36     {  
37         return this; //Return the JTabbedPane  
38     }
```

BUILDTABPAGE CLASS

```
37  public void add(BuildPanel panel,
40      GridBagConstraints gridBagConstraints) {
41      System.out.println("IMPLEMENTATION ADD METHOD (
42          BUILDTABPAGE) !!!! ");
43
44      getTabPane().add(panel,gridBagConstraints); // 
45          Add panel to JTabbedPane with respect to
46          gridBagConstraints
47  }
48
49  public int getNumberofItems()
50  {
51      return getTabPane().getComponentCount(); // 
52          Count the components number of the
53          JTabbedPane
54  }
55
56  public void add(BuildParameter panel,
57      GridBagConstraints gridBagConstraints) {
58      System.out.println("IMPLEMENTATION ADD METHOD (
59          BUILDTABPAGE PARAMETER) !!!! ");
60
61      getTabPane().add(panel,gridBagConstraints); // 
62          Add parameter panel to JTabbedPane with
63          respect to gridBagConstraints
64  }
65
66  public void printParameters()
67  {
68      System.out.println("Number of parameters: " +
69          this.getComponentCount());
70      for (int i = 0 ; i < this.getComponentCount() ;
71          ++i) //For the number of components
72      {
73          Component cp = this.getComponent(i); //Get
74              component no.i
75          if (cp instanceof BuildParameter) //If cp is
76              a parameter
77              ((BuildParameter) cp).getParameter().
78                  printContent(); //Print the parameter
79                  content
80          else
81          {
82              System.out.println("Tab name: " + ((
83                  Container) cp).getName()); //Get the
84                  name of cp and use it as the tab name
85              ((LayoutPane) cp).printParameters(); /* 
86                  JComboBoxes and CheckBoxes are not read.
87                  FIX IT*/
88          }
89  }
```

BUILDTABPAGE CLASS

69	}	
71	}	

K

BUILDCOMBOBOX CLASS

```
1 package layoutCreator;
2
3 import java.awt.Component;
4 import java.awt.Container;
5 import java.awt.GridBagConstraints;
6 import java.awt.event.ItemListener;
7 import javax.swing.JComboBox;
8 import javax.swing.JPanel;
9
10
11 public class BuildComboBox extends JComboBox
12     implements LayoutPane {
13
14     private static final long serialVersionUID = 1L;
15
16     public BuildComboBox()
17     {
18         super();
19     }
20
21     public void add(BuildPanel panel,
22                     GridBagConstraints gridBagConstraints) {
23
24     }
25
26     public void add(BuildParameter panel,
27                     GridBagConstraints gridBagConstraints) {
28
29     }
30
31     public int getNumberofItems() {
32         return this.getItemCount(); //Return the number
33             of parameters
34     }
35
36     public void setName(String name) { //Set the name
37         of each item
38 }
```

BUILDCOMBOPBOX CLASS

```
36    }
37
38    public void setToolTipText(String text) { //The
39        text displays when the cursor lingers over the
40        component
41    }
42
43    public void printParameters()
44    {
45        System.out.println("Number of parameters: " +
46            this.getNumberOfItems());
47        for (int i = 0 ; i < this.getNumberOfItems() ;
48            ++i) //For each parameter
49        {
50            System.out.println("ComboBox name: " + this
51                .getItemAt(i));
52        }
53    }
54}
```

L

BUILDCHECKBOX CLASS

```
1 package layoutCreator;
2
3 import java.awt.GridBagConstraints;
4 import javax.swing.JCheckBox;
5
6 public class BuildCheckBox extends JCheckBox
7     implements LayoutPane {
8
9     private static final long serialVersionUID = 1L;
10
11    public BuildCheckBox()
12    {
13        super();
14    }
15
16    public void add(BuildPanel panel,
17                    GridBagConstraints gridBagConstraints) { //Add
18        // a checkbox
19    }
20
21    public void add(BuildParameter panel,
22                    GridBagConstraints gridBagConstraints) { //Add
23        // the boolean value of the checkbox
24    }
25
26    public int getNumberOfItems() {
27        return this.getComponentCount(); //Return the
28        // number of checkboxes in the panel
29    }
30
31    public void setName(String name) { //Set the name
32        // of the checkbox
33    }
34}
```

BUILDCHECKBOX CLASS

```
32  public void setToolTipText(String text) { //The
      text displays when the cursor lingers over the
      checkbox
34 }
36  public int getComponentCount() {
      return this.getComponentCount(); //Return the
      number of components
38 }
40  public void printParameters() {
42
      System.out.println("Number of CheckBoxes: " +
            this.getNumberOfItems()); //Print the number
            of checkboxes in the panel
44  for (int i = 0 ; i < this.getNumberOfItems() ;
            ++i) //For each checkbox included in the
            panel
      System.out.println("CheckBox name: " + this
            .getComponent(i)); //print the name of
            each checkbox
46 }
}
```

M

LAYOUTPANE INTERFACE

```
1 package layoutCreator;  
3 import java.awt.GridBagConstraints;  
5 public interface LayoutPane { //This interface  
   contains all the methods commonly used by  
   BuildCheckBox , BuildPanel , BuildComboBox and  
   BuildTabPage classes  
7   void add(BuildPanel panel , GridBagConstraints  
             gridBagConstraints);  
9   void add(BuildParameter panel , GridBagConstraints  
             gridBagConstraints);  
11  int getNumberofItems();  
13  void setName(String name);  
15  void setToolTipText(String text);  
17  int getComponentCount();  
19  void printParameters();  
21 }
```


N

PASSWINDOWREADER CLASS

```
1 package regularExpressions;
2 import com.cloudgarden.layout.AnchorLayout;
3
4 import java.awt.Dimension;
5 import java.awt.GridBagConstraints;
6 import java.awt.GridBagLayout;
7
8 import java.awt.GridLayout;
9 import java.awt.Insets;
10 import java.awt.event.ItemEvent;
11 import java.awt.event.ItemListener;
12 import java.util.List;
13 import java.util.logging.FileHandler;
14 import java.util.logging.Level;
15 import java.util.logging.Logger;
16
17 import javax.swing.ComboBoxModel;
18 import javax.swing.DefaultComboBoxModel;
19 import javax.swing.GroupLayout;
20
21 import javax.swing.JButton;
22 import javax.swing.JCheckBox;
23 import javax.swing.JComboBox;
24 import javax.swing.JComponent;
25 import javax.swing.JFrame;
26 import javax.swing.JLabel;
27 import javax.swing.JPanel;
28 import javax.swing.JProgressBar;
29 import javax.swing.JTabbedPane;
30 import javax.swing.JTextField;
31 import javax.swing.LayoutStyle;
32
33 import javax.swing.WindowConstants;
34 import javax.swing.SwingUtilities;
35
36 import layoutCreator.BuildPanel;
```

39

PASSWINDOWREADER CLASS

```
public class PassWindowReader extends javax.swing.JFrame {
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel ButtonPanel;
    private JPanel jPanel1;
    private JComboBox myComboBox;
    private JTextField jTextField1;
    private JLabel jLabel1;
    private JCheckBox jCheckBox1;
    private JPanel jPanel2;
    private JButton CancelButton;
    private JButton ApplyButton;
    private String labelText;
    private static Object comboBoxtext;
    boolean b = false;
    private static String textField;
    private String checkBoxText;
    private JLabel comboBoxPopupLabel;
    private JTextField comboBoxTextField;
    private JPanel comboBoxPanel;
    private JPanel jPanel4;
    private JPanel jPanel3;
    private JTabbedPane jTabbedPane1;
    private String comboBoxPanelLabel;
    private String comboBoxTextField;
    private String comboBoxLabel;
    /**
     * Auto-generated main method to display this JFrame
     */
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                PassWindowReader inst = new PassWindowReader();
                inst.setLocationRelativeTo(null);
                inst.setVisible(true);
            }
        });
    }
    public PassWindowReader() {
        super();
        initGUI();
    }
}
```

PASSWINDOWREADER CLASS

```

89
90     }
91
92     private void initGUI() {
93         try {
94             GridBagLayout thisLayout = new GridBagLayout
95                 ();
96             thisLayout.rowWeights = new double[] {0.1};
97             thisLayout.rowHeights = new int[] {};
98             thisLayout.columnWeights = new double[] {};
99             thisLayout.columnWidths = new int[] {};
100            getContentPane().setLayout(thisLayout);
101            setDefaultCloseOperation(WindowConstants.
102                DISPOSE_ON_CLOSE);
103        {
104            ButtonPanel = new JPanel();
105            GridBagLayout ButtonPanelLayout = new
106                GridBagLayout();
107            getContentPane().add(ButtonPanel, new
108                GridBagConstraints(1, 1, 1, 1, 0.0, 0.0,
109                    GridBagConstraints.SOUTH,
110                    GridBagConstraints.NONE, new Insets(0,
111                        0, 0, 0), 0, 0));
112            Dimension buttonPanelDimension =
113                ButtonPanel.getSize();
114            ButtonPanel.setSize(buttonPanelDimension);
115            ButtonPanelLayout.rowWeights = new double[]
116                {0.1, 0.1, 0.1, 0.1};
117            ButtonPanelLayout.rowHeights = new int[]
118                {7, 7, 7, 7};
119            ButtonPanelLayout.columnWeights = new
120                double[] {0.1, 0.1, 0.1, 0.1};
121            ButtonPanelLayout.columnWidths = new int[]
122                {7, 7, 7, 7};
123            ButtonPanel.setLayout(ButtonPanelLayout);
124        {
125            ApplyButton = new JButton();
126            GroupLayout ApplyButtonLayout = new
127                GroupLayout((JComponent)ApplyButton);
128            ButtonPanel.add(ApplyButton, new
129                GridBagConstraints(0, 0, 1, 1, 0.0,
130                    0.0, GridBagConstraints.CENTER,
131                    GridBagConstraints.NONE, new Insets(0,
132                        0, 0, 0), 0, 0));
133            ApplyButton.setText("Apply");
134            ApplyButton.setLayout(ApplyButtonLayout);
135
136            //action listener assigned to Apply
137            //button
138            ApplyButton.addActionListener(new java.
139                awt.event.ActionListener() {

```

PASSWINDOWREADER CLASS

```
119         public void actionPerformed(java.awt.
120             event.ActionEvent e) {
121                 //If the textfield doesn't change,
122                 //print a blank space else print the
123                 //new value
124
125                 textField = jTextField1.getText();
126
127                 b = jCheckBox1.isSelected();
128                 String CheckBoxResult = checkBoxText+
129                     ":" +b+"\n";//Merge the checkbox
130                     label with the checkbox output
131                 String TextFieldResult = labelText+":
132                     "+textField+"\n";//Merge the
133                     textfield label with the textfield
134                     output
135
136                 //If the combotextfield doesn't
137                 //change, print a blank space else
138                 //print the new value
139
140                 comboBoxTextField = comboBoxTextField.
141                     getText();
142
143                 String comboBoxPanelResult =
144                     comboBoxLabel+": "+comboBoxTextField+
145                     "\n";//Merge the combobox label
146                     with the combobox textfield output
147
148                 System.out.println("Apply: \n"+
149                     TextFieldResult+CheckBoxResult+
150                     comboBoxPanelResult);
151
152                 //XML logging
153
154                 try {
155                     FileHandler handler = new
156                         FileHandler("RISALog.xml");
157                     Logger logger = Logger.
158                         getLogger("java2s.logging");
159                     logger.addHandler(handler);
160                     logger.log(Level.INFO,
161                         TextFieldResult+
162                         CheckBoxResult+
163                         comboBoxPanelResult);
164                 } catch (Exception ex) {
165                     ex.printStackTrace();
166                 }
167
168                 //end of XML logging
169             }
170         });
171     }
172 }
```

PASSWINDOWREADER CLASS

```

149         ApplyButtonLayout.setVerticalGroup(
150             ApplyButtonLayout.createParallelGroup(
151                 () );
152         ApplyButtonLayout.setHorizontalGroup(
153             ApplyButtonLayout.createParallelGroup(
154                 () );
155     }
156     {
157         CancelButton = new JButton();
158         ButtonPanel.add(CancelButton, new
159             GridBagConstraints(3, 0, 1, 1, 0.0,
160                 0.0, GridBagConstraints.CENTER,
161                 GridBagConstraints.NONE, new Insets(0,
162                     0, 0, 0), 0, 0));
163         CancelButton.setText("Cancel");
164
165         CancelButton.addActionListener(new java.
166             awt.event.ActionListener() {
167                 public void actionPerformed(java.awt.
168                     event.ActionEvent e) {
169                     System.out.println("Cancel");
170                     dispose();
171                 }
172             });
173     }
174     {
175         jTabbedPane1 = new JTabbedPane();
176         Dimension tabSize = jTabbedPane1.getSize();
177         jTabbedPane1.setSize(tabSize);
178         getContentPane().add(jTabbedPane1, new
179             GridBagConstraints(1, 0, 1, 1, 0.0, 0.0,
180                 GridBagConstraints.CENTER,
181                 GridBagConstraints.BOTH, new Insets(0,
182                     0, 0, 0), 0, 0));
183
184         JPanel1 = new JPanel();
185         GridBagLayout jPanel1Layout = new
186             GridBagLayout();
187         jTabbedPane1.addTab("jPanel1", null,
188             jPanel1, null);
189         jPanel1Layout.rowWeights = new double[]
190             {0.1, 0.1};
191         jPanel1Layout.rowHeights = new int[] {7,
192             7};
193         jPanel1Layout.columnWeights = new double
194             [] {0.1};
195         jPanel1Layout.columnWidths = new int[]
196             {7};
197         jPanel1.setLayout(jPanel1Layout);

```

PASSWINDOWREADER CLASS

```
181     {
182         jPanel3 = new JPanel();
183         Dimension panel3Size = jPanel3.getSize()
184             ();
185         jPanel3.setSize(panel3Size);
186         jPanel1.add(jPanel3, new
187             GridBagConstraints(1, 1, 1, 1, 0.0,
188                 0.0, GridBagConstraints.CENTER,
189                 GridBagConstraints.BOTH, new Insets
190                     (0, 0, 0, 0), 0, 0));
191         GridBagLayout jPanel3Layout = new
192             GridBagLayout();
193         jPanel3Layout.columnWidths = new int []
194             {7};
195         jPanel3Layout.rowHeights = new int []
196             {7, 7};
197         jPanel3Layout.columnWeights = new
198             double [] {0.1};
199         jPanel3Layout.rowWeights = new double []
200             {0.1, 0.1};
201         jPanel3.setLayout(jPanel3Layout);
202         {
203             jTextField1 = new JTextField();
204             jPanel3.add(jTextField1, new
205                 GridBagConstraints(0, 1, 1, 1,
206                     0.0, 0.0, GridBagConstraints.
207                         CENTER, GridBagConstraints.NONE,
208                         new Insets(0, 0, 0, 0), 0, 0));
209             jTextField1.setText("jTextField1");
210         }
211         {
212             jLabel1 = new JLabel();
213             jPanel3.add(jLabel1, new
214                 GridBagConstraints(0, 0, 1, 1,
215                     0.0, 0.0, GridBagConstraints.
216                         CENTER, GridBagConstraints.NONE,
217                         new Insets(0, 0, 0, 0), 0, 0));
218             AnchorLayout jLabel1Layout = new
219                 AnchorLayout();
220             jLabel1.setText("A random label");
221             jLabel1.setLayout(jLabel1Layout);
222             labelText = jLabel1.getText();
223         }
224     }
225     Dimension panel1Size = jPanel1.getSize();
226     jPanel1.setSize(panel1Size);
227     {
228         jPanel2 = new JPanel();
229         jTabbedPane1.addTab("jPanel2", null,
230             jPanel2, null);
231     }
```

PASSWINDOWREADER CLASS

```

211     GridBagConstraints jPanel2Layout = new
212         GridBagConstraints();
213     jPanel2Layout.rowWeights = new double []
214         {0.1, 0.1, 0.1};
215     jPanel2Layout.rowHeights = new int [] {7,
216         7, 7};
217     jPanel2Layout.columnWeights = new double
218         [] {0.1};
219     jPanel2Layout.columnWidths = new int []
220         {7};
221     Dimension panelSize = jPanel2.getSize();
222     jPanel2.setSize(panelSize);
223     jPanel2.setLayout(jPanel2Layout);
224     {
225         comboBoxPanel = new JPanel();
226         jPanel2.add(comboBoxPanel, new
227             GridBagConstraints(0, 2, 1, 1, 0.0,
228                 0.0, GridBagConstraints.PAGE_END,
229                 GridBagConstraints.BOTH, new Insets
230                     (0, 0, 0, 0), 0, 0));
231         GridBagConstraints comboBoxPanelLayout = new
232             GridBagConstraints();
233         comboBoxPanelLayout.columnWidths = new
234             int [] {7};
235         comboBoxPanelLayout.rowHeights = new
236             int [] {7};
237         comboBoxPanelLayout.columnWeights = new
238             double []
239                 {0.1};
240         comboBoxPanelLayout.rowWeights = new
241             double []
242                 {0.1};
243         comboBoxPanel.setLayout(
244             comboBoxPanelLayout);
245         Dimension comboBoxPanelSize =
246             comboBoxPanel.getSize();
247         comboBoxPanel.setSize(comboBoxPanelSize
248             );
249         comboBoxPanel.setVisible(false);
250         {
251             comboBoxPopupLabel = new JLabel();
252             comboBoxPanel.add(comboBoxPopupLabel,
253                 new GridBagConstraints(0, 2, 1,
254                     1, 0.0, 0.0, GridBagConstraints.
255                     CENTER, GridBagConstraints.BOTH,
256                     new Insets(0, 0, 0, 0), 0, 0));
257             comboBoxPopupLabel.setText("Another
258                 label");
259             comboBoxPopupLabel.setLayout(null);
260             comboBoxLabel = comboBoxPopupLabel.
261                 getText();
262         }
263     }

```

PASSWINDOWREADER CLASS

```
239         comboBoxTextField = new JTextField();
240         comboBoxPanel.add(comboBoxTextField,
241             new GridBagConstraints(0, 3, 1, 1,
242                 0.0, 0.0, GridBagConstraints.
243                 CENTER, GridBagConstraints.BOTH,
244                 new Insets(0, 0, 0, 0), 0, 0));
245         comboBoxTextField.setText(""
246             + jTextField2);
247     }
248     {
249         final ComboBoxModel myComboBoxModel =
250             new DefaultComboBoxModel(
251                 new String[] { "Something", "
252                     Something else" });
253         myComboBox = new JComboBox();
254         jPanel2.add(myComboBox, new
255             GridBagConstraints(0, 0, 1, 1, 0.0,
256                 0.0, GridBagConstraints.CENTER,
257                 GridBagConstraints.NONE, new Insets
258                 (0, 0, 0, 0), 0, 0));
259         myComboBox.setModel(myComboBoxModel);
260         comboBoxtext = myComboBox.
261             getSelectedItem().toString();
262
263         myComboBox.addItemListener(new
264             ItemListener() {
265                 public void itemStateChanged(
266                     ItemEvent evt) {
267                         System.out.println("Event: "+evt);
268
269                         if (myComboBox.getSelectedItem().
270                             toString().equals("Something
271                             else"))//If the name of the
272                             JComboBox component equals the
273                             name of one of the bpList names
274                             //which means that one of the
275                             JComboBox items is chosen by
276                             the user
277                         {
278                             System.out.println("Visible");
279                             comboBoxPanel.setVisible(true);
280                             //Set the items of the
281                             JComboBox chosen as visible
282                         }
283                         else
284                         {
285                             System.out.println("Not visible
286                             ");
287                         }
288                     }
289                 );
290             }
291         );
292     }
293 }
```

PASSWINDOWREADER CLASS

```
    comboBoxPanel.setVisible(false)
        ;//Set the items of the
        JComboBox chosen as hidden
267    }

269    }
271    }
273    jCheckBox1 = new JCheckBox();
274    AnchorLayout jCheckBox1Layout = new
275        AnchorLayout();
276    jPanel2.add(jCheckBox1, new
277        GridBagConstraints(0, 1, 1, 1, 0.0,
278            0.0, GridBagConstraints.CENTER,
279            GridBagConstraints.NONE, new Insets
280            (0, 0, 0, 0), 0, 0));
281    jCheckBox1.setText("My CheckBox");
282    jCheckBox1.setLayout(jCheckBox1Layout);
283    checkBoxText = jCheckBox1.getText();
284

285    }
286}

287}

288
289protected BuildPanel getChoice(String string) {
290    // TODO Auto-generated method stub
291    return null;
292}
293}
```


O

EVSELECT .PAR FILE

```
1 <FILE>
2 <CONFIG>
3
4 <PARAM id="table" type="table" mandatory="yes">
5   <DESCRIPTION> Name of the table to be filtered </
6   DESCRIPTION>
7 </PARAM>
8
9 <PARAM id="keepfilteroutput" type="bool" default="no">
10  <DESCRIPTION> Keep output of filtering process? </
11  DESCRIPTION>
12 <CASE>
13  <ITEM value="no"> </ITEM>
14  <ITEM value="yes">
15    <PARAM id="withfilteredset" type="bool"
16      default="no">
17      <DESCRIPTION> Create a filtered event list </
18      DESCRIPTION>
19    <CASE>
20      <ITEM value="no"> </ITEM>
21      <ITEM value="yes">
22        <PARAM id="filteredset" type="dataset"
23          default="filtered.fits">
24          <DESCRIPTION> Name of file for
25            filtered event list </DESCRIPTION>
26        </PARAM>
27      </ITEM>
28    </CASE>
29  </PARAM>
30
31 <PARAM id="destruct" type="bool" default="yes">
32   <DESCRIPTION> Destructive event selection? </
33   DESCRIPTION>
34 <CASE>
35   <ITEM value="no">
```

EVSELECT .PAR FILE

```
32      <PARAM id="flagcolumn" type="string" default=
         "EVFLAG">
         <DESCRIPTION> Name of integer column for
             event flagging </DESCRIPTION>
     </PARAM>

36      <PARAM id="flagbit" type="int" default="-1">
         <DESCRIPTION> Bit position in flagcolumn
             column to save select status </
             DESCRIPTION>
         <CONSTRAINTS>
             flagbit in [-1:31]
         </CONSTRAINTS>
     </PARAM>
</ITEM>
<ITEM value="yes"> </ITEM>
</CASE>
</PARAM>

46      <PARAM id="filtertype" type="string" default="
         expression">
         <DESCRIPTION> Type of filtering to use </
             DESCRIPTION>
<CASE>
50          <ITEM value="dataSubspace">
              <PARAM id="dssblock" type="block" default="">
                  <DESCRIPTION> Name of block containing the
                      Data Subspace specification to use for
                      filtering </DESCRIPTION>
              </PARAM>
</ITEM>
54          <ITEM value="expression">
              <PARAM id="expression" type="string" default=
                  "true">
                  <DESCRIPTION> Filtering expression </
                      DESCRIPTION>
              </PARAM>
</ITEM>
58          </ITEM>
</CASE>
60      </PARAM>

62      <PARAM id="writedss" type="bool" default="yes">
         <DESCRIPTION> Write data subspace information to
             the output files </DESCRIPTION>
<CASE>
66          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
              <PARAM id="cleandss" type="bool" default="no"
                  >
```

EVSELECT .PAR FILE

```

        <DESCRIPTION> Remove components from the
        data subspace which select no events </
        DESCRIPTION>
70      </PARAM>

72      <PARAM id="updateexposure" type="bool"
             default="yes">
        <DESCRIPTION> Update exposure information
        in the output files (XMM specific) </
        DESCRIPTION>
74      </PARAM>

76      <PARAM id="filterexposure" type="bool"
             default="yes">
        <DESCRIPTION> Filter exposure extensions
        with the GTIs which apply to them </
        DESCRIPTION>
78      </PARAM>
    </ITEM>
80  </CASE>
</PARAM>

82  <PARAM id="blockscopy" type="string" list="yes">
84    <DESCRIPTION> Extentions to copy to the extracted
        data sets </DESCRIPTION>
</PARAM>

86  <PARAM id="attributecopy" type="string" list="
     yes">
88    <DESCRIPTION> Attributes to copy from the input
        table to the output product table </
        DESCRIPTION>
</PARAM>

90  <PARAM id="energycolumn" type="string" default="PHA
     ">
92    <DESCRIPTION> Name of col for energy information
        for spectra and light curves </DESCRIPTION>
</PARAM>

94  <PARAM id="withzcolumn" type="bool" default="no">
96    <DESCRIPTION> Use zcolumn for product
        accumulation </DESCRIPTION>
<CASE>
98    <ITEM value="no"> </ITEM>
    <ITEM value="yes">
100      <PARAM id="zcolumn" type="string" default="
           WEIGHT">
        <DESCRIPTION> Column of values to
        accumulate in an image, lightcurve,
        spectrum or histogram </DESCRIPTION>

```

EVSELECT .PAR FILE

```
102      </PARAM>
104      <PARAM id="withzerrorcolumn" type="bool"
105          default="no">
106          <DESCRIPTION> Use zerrorcolumn as the error
107              on zcolumn </DESCRIPTION>
108          <CASE>
109              <ITEM value="no"> </ITEM>
110              <ITEM value="yes">
111                  <PARAM id="zerrorcolumn" type="string"
112                      default="EWEIGHT">
113                      <DESCRIPTION> Column of the error on
114                          the zcolumn value for a lightcurve
115                          , spectrum or histogram </
116                          DESCRIPTION>
117                  </PARAM>
118              </ITEM>
119          </CASE>
120      </PARAM>
121      <PARAM id="ignorelegallimits" type="bool" default="
122          no">
123          <DESCRIPTION> Ignore TLMIN/MAX values when
124              extracting data from columns? </DESCRIPTION>
125      </PARAM>
126      <PARAM id="withimageset" type="bool" default="no">
127          <DESCRIPTION> Extract an image file </DESCRIPTION>
128          <CASE>
129              <ITEM value="no"> </ITEM>
130              <ITEM value="yes">
131                  <PARAM id="imageset" type="dataset" default="
132                      image.fits">
133                      <DESCRIPTION> Name of image file to extract
134                          </DESCRIPTION>
135                  </PARAM>
136                  <PARAM id="xcolumn" type="string" default="
137                      RAWX">
138                      <DESCRIPTION> Name of X coord column for
139                          image creation </DESCRIPTION>
140                  </PARAM>
141                  <PARAM id="ycolumn" type="string" default="
142                      RAWY">
143                      <DESCRIPTION> Name of Y coord column for
144                          image creation </DESCRIPTION>
```

EVSELECT .PAR FILE

```

138  </PARAM>
140  <PARAM id="imagebinning" type="string"
        default="imageSize">
        <DESCRIPTION> Use bin sizes or image sizes
                      to determine binning factor </
        DESCRIPTION>
142  <CASE>
143    <ITEM value="binSize">
144      <PARAM id="ximagebinsize" type="real"
              default="1" mandatory="yes">
              <DESCRIPTION> Binning factor for x
                            axis in image creation </
              DESCRIPTION>
145      <CONSTRAINTS>
146        ximagebinsize in [0:]
147      </CONSTRAINTS>
148    </PARAM>
149
150    <PARAM id="yimagebinsize" type="real"
              default="1" mandatory="yes">
              <DESCRIPTION> Binning factor for y
                            axis in image creation </
              DESCRIPTION>
151      <CONSTRAINTS>
152        yimagebinsize in [0:]
153      </CONSTRAINTS>
154    </PARAM>
155
156  </ITEM>
157  <ITEM value="imageSize">
158    <PARAM id="squarepixels" type="bool"
              default="no">
              <DESCRIPTION> Force x and y bin size
                            to be the same when imagebinning=
                            imageSize </DESCRIPTION>
159    </PARAM>
160
161    <PARAM id="ximagesize" type="int"
              default="600">
              <DESCRIPTION> Image size in the x
                            coordinate (used to determine
                            binning factor) </DESCRIPTION>
162    <CONSTRAINTS>
163      ximagesize in [0:]
164    </CONSTRAINTS>
165  </PARAM>
166
167  <PARAM id="yimagesize" type="int"
              default="600">

```

EVSELECT .PAR FILE

```
172         <DESCRIPTION> Image size in the y  
173             coordinate (used to determine  
174                 binning factor) </DESCRIPTION>  
175             <CONSTRAINTS>  
176                 yimagesize in [0:]  
177             </CONSTRAINTS>  
178         </PARAM>  
179     </ITEM>  
180     </CASE>  
181 </PARAM>  
  
182     <PARAM id="withxranges" type="bool" default="no">  
183         <DESCRIPTION> Use min/max values for x axis  
184             image extraction </DESCRIPTION>  
185         <CASE>  
186             <ITEM value="no"> </ITEM>  
187             <ITEM value="yes">  
188                 <PARAM id="ximagemin" type="real"  
189                     default="1" mandatory="yes">  
190                     <DESCRIPTION> Lower limit of x axis  
191                         for image extraction </DESCRIPTION>  
192                     >  
193                 </PARAM>  
194             </ITEM>  
195         </CASE>  
196     </PARAM>  
  
197     <PARAM id="withyranges" type="bool" default="no">  
198         <DESCRIPTION> Use min/max values for x axis  
199             image extraction </DESCRIPTION>  
200         <CASE>  
201             <ITEM value="no"> </ITEM>  
202             <ITEM value="yes">  
203                 <PARAM id="yimagemin" type="real"  
204                     default="1" mandatory="yes">  
205                     <DESCRIPTION> Lower limit of y axis  
206                         for image extraction </DESCRIPTION>  
207                     >  
208                 </PARAM>  
209             </ITEM>  
210         </CASE>  
211     </PARAM>
```

EVSELECT .PAR FILE

```

206      <DESCRIPTION> Upper limit of y axis
207          for image extraction </DESCRIPTION>
208          >
209      </PARAM>
210  </ITEM>
211  </CASE>
212 </PARAM>

213 <PARAM id="withimagedatatype" type="bool"
214     default="no">
215     <DESCRIPTION> Use imagedatatype to set the
216         type of image created </DESCRIPTION>
217     <CASE>
218         <ITEM value="no"> </ITEM>
219         <ITEM value="yes">
220             <PARAM id="imagedatatype" type="string"
221                 default="Real64">
222                 <DESCRIPTION> Data type of the image
223                     to be created </DESCRIPTION>
224             <CASE>
225                 <ITEM value="Int16"> </ITEM>
226                 <ITEM value="Int32"> </ITEM>
227                 <ITEM value="Int8"> </ITEM>
228                 <ITEM value="Real32"> </ITEM>
229                 <ITEM value="Real64"> </ITEM>
230             </CASE>
231         </PARAM>
232     </ITEM>
233     </CASE>
234 </PARAM>

235 <PARAM id="withcelestialcenter" type="bool"
236     default="no">
237     <DESCRIPTION> Shift the center of the image
238         to the specified ra and dec </
239         DESCRIPTION>
240     <CASE>
241         <ITEM value="no"> </ITEM>
242         <ITEM value="yes">
243             <PARAM id="raimagecenter" type="real"
244                 default="0" mandatory="yes">
245                 <DESCRIPTION> Right ascension of the
246                     center of the image, in decimal
247                     degrees </DESCRIPTION>
248             </PARAM>
249
250             <PARAM id="decimagecenter" type="real"
251                 default="0" mandatory="yes">
252                 <DESCRIPTION> Declination of the
253                     center of the image, in decimal
254                     degrees </DESCRIPTION>

```

EVSELECT .PAR FILE

```

242          </PARAM>
243          </ITEM>
244      </CASE>
245      </PARAM>
246      </ITEM>
247      </CASE>
248  </PARAM>

250  <PARAM id="witspectrumset" type="bool" default="no"
251      ">
252      <DESCRIPTION> Extract a spectrum file </
253          DESCRIPTION>
254  <CASE>
255      <ITEM value="no"> </ITEM>
256      <ITEM value="yes">
257          <PARAM id="spectrumset" type="dataset"
258              default="spectrum.fits">
259          <DESCRIPTION> Name of spectrum file to
260              extract </DESCRIPTION>
261      </PARAM>
262
263      <PARAM id="spectralbinsize" type="int"
264          default="10">
265          <DESCRIPTION> Binning factor for spectrum
266              creation </DESCRIPTION>
267          <CONSTRAINTS>
268              spectralbinsize in [1:]
269          </CONSTRAINTS>
270      </PARAM>

271      <PARAM id="witspecranges" type="bool"
272          default="no">
273          <DESCRIPTION> Use min/max values for
274              spectral channels </DESCRIPTION>
275  <CASE>
276      <ITEM value="no"> </ITEM>
277      <ITEM value="yes">
278          <PARAM id="specchannelmin" type="real"
279              default="0" mandatory="yes">
280          <DESCRIPTION> Minimum channel for
281              spectrum creation </DESCRIPTION>
282          <CONSTRAINTS>
283              specchannelmin in [0:]
284          </CONSTRAINTS>
285      </PARAM>

286      <PARAM id="specchannelmax" type="real"
287          default="4095" mandatory="yes">
288          <DESCRIPTION> Maximum channel for
289              spectrum creation </DESCRIPTION>
290          <CONSTRAINTS>

```

EVSELECT .PAR FILE

```

specchannelmax in [o:]
  </CONSTRAINTS>
</PARAM>
</ITEM>
</CASE>
</PARAM>
</ITEM>
</CASE>
</PARAM>
<PARAM id="withrateset" type="bool" default="no">
  <DESCRIPTION> Extract a time series </DESCRIPTION>
  >
<CASE>
  <ITEM value="no"> </ITEM>
  <ITEM value="yes">
    <PARAM id="rateset" type="dataset" default="rate.fits">
      <DESCRIPTION> Name of time series file to
      extract </DESCRIPTION>
    </PARAM>

    <PARAM id="timecolumn" type="string" default="TIME">
      <DESCRIPTION> Name of col for time
      information </DESCRIPTION>
    </PARAM>

    <PARAM id="timebinsize" type="real" default="1">
      <DESCRIPTION> Size of time bins for time
      series files </DESCRIPTION>
      <CONSTRAINTS>
        timebinsize in [o:]
      </CONSTRAINTS>
    </PARAM>

    <PARAM id="withtimeranges" type="bool"
      default="no">
      <DESCRIPTION> Use min/max values for time
      series extraction </DESCRIPTION>
      <CASE>
        <ITEM value="no"> </ITEM>
        <ITEM value="yes">
          <PARAM id="timemin" type="time" default
            ="o" mandatory="yes">
            <DESCRIPTION> Lower limit for time
            series </DESCRIPTION>
          </PARAM>
        </ITEM>
      </CASE>
    </PARAM>
  </ITEM>
</CASE>
</PARAM>

```

EVSELECT .PAR FILE

```

320      <PARAM id="timemax" type="time" default
321          ="1000" mandatory="yes">
322          <DESCRIPTION> Upper limit for time
323              series </DESCRIPTION>
324          </PARAM>
325      </ITEM>
326  </CASE>
327 </PARAM>

328      <PARAM id="maketimecolumn" type="bool"
329          default="no">
330          <DESCRIPTION> Include a time column in the
331              time series extention </DESCRIPTION>
332      </PARAM>

333      <PARAM id="makeratecolumn" type="bool"
334          default="no">
335          <DESCRIPTION> Produce a lightcurve of rates
336              rather than counts </DESCRIPTION>
337      </PARAM>
338  </ITEM>
339 </CASE>
340 </PARAM>

341      <PARAM id="withhistogramset" type="bool" default="
342          no">
343          <DESCRIPTION> Extract a general histogram </
344              DESCRIPTION>
345      <CASE>
346          <ITEM value="no"> </ITEM>
347          <ITEM value="yes">
348              <PARAM id="histogramset" type="dataset"
349                  default="histo.fits">
350                  <DESCRIPTION> Name of the histogram file to
351                      extract </DESCRIPTION>
352              </PARAM>
353
354              <PARAM id="histogramcolumn" type="string"
355                  default="TIME">
356                  <DESCRIPTION> Name of col for histogram
357                      generation </DESCRIPTION>
358              </PARAM>
359
360              <PARAM id="histogrambinsize" type="real"
361                  default="1">
362                  <DESCRIPTION> Size of bins for histogram
363                      files </DESCRIPTION>
364                  <CONSTRAINTS>
365                      histogrambinsize in [0:]
366                  </CONSTRAINTS>
367              </PARAM>

```

EVSELECT .PAR FILE

```
358  <PARAM id="withhistoranges" type="bool"
      default="no">
      <DESCRIPTION> Use min/max values for
                    histogram extraction </DESCRIPTION>
360  <CASE>
      <ITEM value="no"> </ITEM>
362  <ITEM value="yes">
      <PARAM id="histogrammin" type="real"
            default="0" mandatory="yes">
            <DESCRIPTION> Lower limit for
                          histogram </DESCRIPTION>
      </PARAM>
366  <PARAM id="histogrammax" type="real"
            default="1000" mandatory="yes">
            <DESCRIPTION> Upper limit for
                          histogram </DESCRIPTION>
            </PARAM>
370  </ITEM>
372  </CASE>
374  </PARAM>
376  </CONFIG>
378  </FILE>
```


P

EVSELECT .LYT FILE

```
Cards
2   Page General
    Frame InOut
      Parameter table
      Enable keepfilteroutput
      Enable withfilteredset
      Parameter filteredset
      end
    end
    Enable writedss
      Parameter cleandss
      Enable updateexposure
      Parameter filterexposure
      end
    end
    Parameter blockstocopy
  end
18  Frame Filtering
  Choice filtertype
  Page expression
    Parameter expression
  end
  Page dataSubspace
    Parameter dssblock
  end
26  end
  Parameter destruct
  Parameter flagcolumn
  Parameter flagbit
  Parameter ignoregallimits
end
32  Frame MultiUseColumns
  Parameter energycolumn
  Enable withzcolumn
    Parameter zcolumn
    Enable withzerrorcolumn
      Parameter zerrorcolumn
    end
  end
38  end
```

EVSELECT .LYT FILE

```
40      end
41  end
42  Page Image
43      Enable withimageset
44          Parameter imageset
45          Frame Columns
46              Parameter xcolumn
47              Parameter ycolumn
48      end
49      Frame Ranges
50          Enable withxranges
51              Parameter ximageamin
52              Parameter ximageamax
53      end
54          Enable withyranges
55              Parameter yimageamin
56              Parameter yimageamax
57      end
58  end
59  Frame Binning
60      Choice imagebinning
61          Page imageSize
62              Parameter ximagesize
63              Parameter yimagesize
64              Parameter squarepixels
65      end
66          Page binSize
67              Parameter ximagebinsize
68              Parameter yimagebinsize
69      end
70  end
71  end
72      Enable withimagedatatype
73          Parameter imagedatatype
74      end
75      Enable withcelestialcenter
76          Parameter rimagecenter
77          Parameter decimagecenter
78      end
79  end
80  end
81  Page Spectrum
82      Enable withspectrumset
83          Parameter spectrumset
84          Parameter spectralbinsize
85      Enable withspecranges
86          Parameter specchannelmin
87          Parameter specchannelmax
88      end
89  end
90  end
```

EVSELECT .LYT FILE

```
91  Page Lightcurve
92    Enable withrateset
93      Parameter rateset
94      Parameter timecolumn
95      Parameter timebinsize
96      Parameter maketimecolumn
97      Parameter makeratecolumn
98    Enable withtimeranges
99      Parameter timemin
100     Parameter timemax
101   end
102   Enable withzcolumn
103     Parameter zcolumn
104   end
105 end
106
107 Page Histogram
108   Enable withhistogramset
109     Parameter histogramset
110     Parameter histogramcolumn
111     Parameter histogrambinsize
112   Enable withhistoranges
113     Parameter histogrammin
114     Parameter histogrammax
115   end
116 end
117 end
118 end
```