ΤΕΙ Κρήτης

Τεχνολογικό Εκπαιδευτικό
Ίδρυμα Κρήτης

ΠΤΥΧΙΑΚΉ ΕΡΓΑΣΊΑ

ΘΕΜΑ : Συνοπτικός, τουριστικός οδηγός για το νομό Ηρακλείου.

ΣΠΟΥΔΑΣΤΈΣ : Συμεουδάκης Χαράλαμπος - 3311
Καλωσυνάκης Ευάγγελος - 2711

Ευχαριστήριο Σημείωμα

Θα θέλαμε να ευχαριστήσουμε τον κύριο Νικόλαο Παπαδάκη για την καθοδήγηση που μας πρόσφερε κατά την εκπόνηση αυτής της εργασίας καθ'όλη την διάρκεια της αναπτυξής της.

Σύνοψη

Η εργασία αυτή και εν συνεχεία η εφαρμογή που αναπτύξαμε, δίνει την δυνατότητα σε οποιοδήποτε άτομο επιθυμεί να επισκεφθεί τον νομό Ηρακλείου, να έχει μία πλήρη επισκόπηση των πιο σημαντικών και γνωστών, σημείων ενδιαφέροντος, ώστε να μπορεί να αποφασίσει γρήγορα εαν το Ηράκλειο είναι ο προορισμός που επιθυμεί πραγματικά και αν αυτό ισχύει, να μπορεί γρήγορα να βρίσκει μέρη και δραστηριότητες που τον ενδιαφέρουν στον νομό μας.

Abstract

This project and in extend, this application, gives anyone who wants to visit Heraklion, the ability to have a complete overview of the most significant and known points of interest, which will help him/her decide quickly if Heraklion is, in fact, the destination which, really, he/she wants to visit and further help him/her decide which of the activities are of interest to him/her.

Περιεχόμενα

1. ΠΕΡΙΓΡΑΦΉ

1. ΠΕΡΙΛΗΨΗ

Το θέμα που επιλέξαμε να αναπτύξουμε, είναι ένας σύντομος και συγκεντρωτικός, πληροφοριακός οδηγός γιά το νομό Ηρακλείου. Ο οδηγός αυτός περιέχει πληροφορίες σχετικά με το που μπορεί ο επισκέπτης του νομού, να βρεί κάποιο μερος που προσφέρει φαγητό, καταστήματα διασκέδασης και αναψυχής, αθλητικές δραστηριότητες καθώς και άλλων ειδών και ιστορικές τοποθεσίες καθώς και την τοποθεσία όπου βρίσκονται όλα όσα προαναφέραμε.

2. ΛΟΓΟΣ

Αρχικά, ο λόγος που επιλέξαμε το συγκεκριμμένο θέμα είναι, διότι ο τουρισμός έχει την μεγαλύτερη συμβολή στο ΑΕΠ[1] της Ελλάδας, το οποίο φαίνεται παρακάτω στην εικόνα 1 οτι ανέρχεται στο ~25% του ετήσιου ΑΕΠ για το 2015. Επίσης, από τις περιοχές της Ελλάδας, η Κρήτη έχει την μεγαλύτερη συμβολή στο ΑΕΠ περιφερειας, όπως φαίνεται στην εικόνα 2, το οποίο μας δίνει μία ιδέα για το ποσό τουρισμού που επισκέπτεται την Κρήτη και εν συνεχεία τον νομό Ηρακλείου.

Κατηγορία Δαπάνης	2013	2014
Δαπάνη Εισερχόμενων Τουριστών	€11.739	€13.187
Δαπάνη Τουριστών Κρουαζιέρας	445	468
Δαπάνη Εταιρειών Κρουαζιέρας	216	227
Αερομεταφορές	1.077	1.177
Θαλάσσιες Μεταφορές	132	133
Εγχώριος Τουρισμός	1.434	1.578
Επενδύσεις	200	200
Άμεση Επίπτωση Τουρισμού	€15.243	€16.971
ως % ΑΕΠ	8,4%	9,5%
πολλαπλασιαστής ΙΟΒΕ	2,2	2,2
Έμμεσο και Άμεσο Αποτέλεσμα	€33.534	€37.337
ως % ΑΕΠ	18,4%	20,9%
πολλαπλασιαστής ΚΕΠΕ	2,65	2,65
Έμμεσο και Άμεσο Αποτέλεσμα	€40.393	€44.974
ως % ΑΕΠ	22,1%	25,1%
ΑΕΠ	€182.400	€178.900

1 Συμβολή τουρισμού στο ΑΕΠ.

[1] Ακκαθάριστο Εγχώριο προϊόν

Περιφέρεια	% κατανομή διανυκτε-ρεύσεων ξενοδοχείων 2013	αναλογία άμεσης τουριστικής δαπάνης 2013 - σε € εκ.	ΑΕΠ Περιφέ-ρειας 2012 - σε εκ.	συμβολή τουρισμού στο ΑΕΠ Περιφέ-ρειας με στοιχεία 2012	κατά κεφαλήν ΑΕΠ - σε €
Κρήτη	28,7%	4.372	9.067	48%	14.398
Ν. Αιγαίο	24,7%	3.767	6.240	60%	18.064
Ιόνια Νησιά	11,0%	1.680	3.402	49%	16.100
Κεντ. Μακεδονία	10,7%	1.626	26.109	6%	13.645
Αττική	9,2%	1.403	94.964	1%	24.099
Πελοπόννησος	3,2%	481	8.241	6%	13.870
Θεσσαλία	2,6%	394	9.505	4%	12.757
& Θράκη	2,4%	369	7.653	5%	12.270
Β. Αιγαίο	2,1%	325	2.784	12%	13.394
Δυτ. Ελλάδα	2,1%	317	9.150	3%	13.431
Στερεά Ελλάδα	1,7%	257	8.543	3%	15.075
Ήπειρος	1,2%	185	4.242	4%	12.207
Δυτ. Μακεδονία	0,5%	69	4.304	2%	15.050
Σύνολο Χώρας	100,0%	15.242	194.204	8%	17.507

2 Συμβολή τουρισμού ανά περιοχή.

Έπειτα, ο λόγος που επιλέξαμε έναν σύντομο οδηγό, σε αντίθεση με έναν ο οποιός προσφέρει αναλυτικές πληροφορίες για τα διάφορα μέρη και καταστήματα του νομού, είναι το οτι οι πληροφορίες που προσπαθούν να προσφέρουν οι οδηγοί αυτοί δν είναι ποτέ αρκετές ώστε να ανταγωνιστούν τις επίσημες ιστοσελίδες των επιμέρους περιεχομένων.Για το λόγο αυτό αναπτύξαμε έναν σύντομο οδηγό ο οποίος έχει ως σκοπό να δώσει αρκετες πληροφορίες σε κάποιο επισκέπτη του νομού, ώστε να αποφασίσει εάν το Ηράκλειο είναι ο προορισμός, στο οποίο θα ήθελε να περάσει τις μέρες των διακοπών του ή την μέρα/τις μέρες μεταξύ κάποιας υποχρέωσης.

2. ΕΡΓΑΛΕΙΑ

1. NETBEANS

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το εργαλείο NetBeans. Το NetBeans είναι ένα εργαλείο ανάπτυξης λογισμικού, γραμμένο στη γλώσσα Java. Κύρια χρήση του είναι, επίσης, η ανάπτυξη λογισμικού σε γλώσσα Java, ωστόσο υποστηρίζει και άλλες γλώσσες προγραμματισμού, όπως C/C++, HTML/HTML5, Javascript,PHP και CSS.

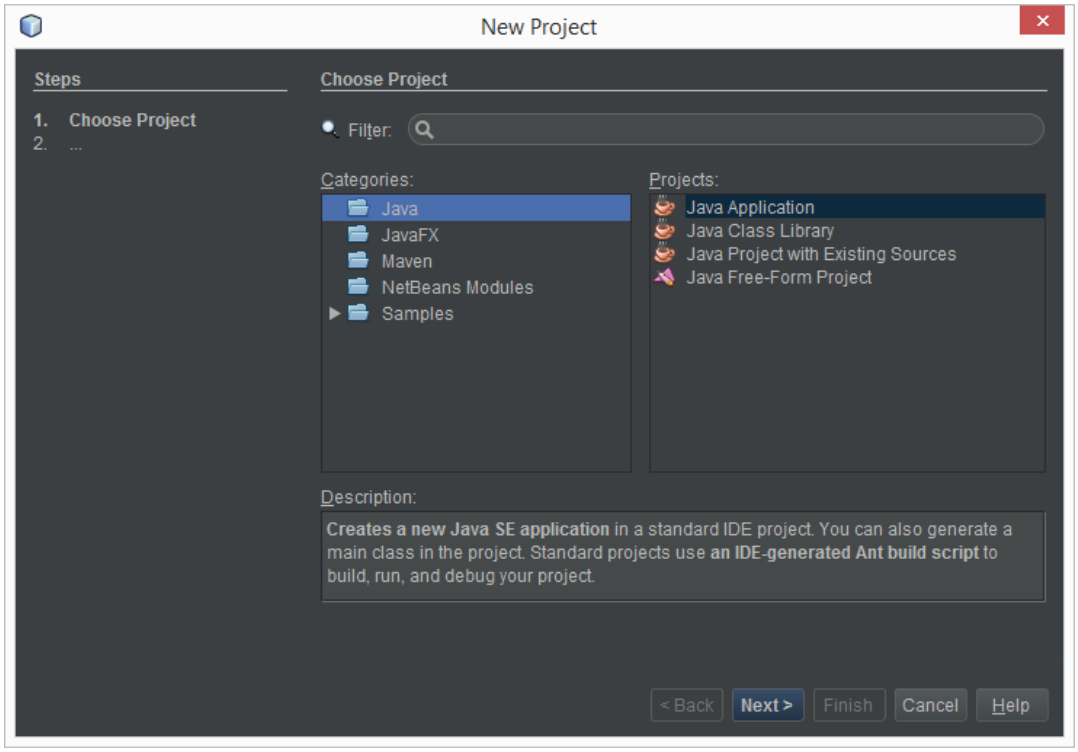
Η δημιουργία του NetBeans, ξεκίνησε ώς εργασία μαθήματος, το 1996, στο πανεπιστήμιο "Charles University" της Πράγας. Το 1997 ο Roman Stanek, ξεκίνησε την εμπορική διάθεση του εργαλείου έως όπου εξαγοράστηκε από την "Sun Microsystems" και αργότερα από την "Oracle Corporation", η οποία έκανε το NetBeans εργαλείο ανοιχτού κώδικα.

Για την εγκατάσταση του NetBeans, χρειάστηκαν να γίνουν κάποια βήματα, τα οποία θα αναφέρουμε παρακάτω για διευκόλυνση.

- Εγκατάσταση του JDK[2] από την ακόλουθη ιστοσελίδα.
- Λήψη του NetBeans από την επίσημη ιστοσελίδα.
- Εγκατάσταση του NetBeans ακολουθώντας τις οδηγίες που δίνονται από την εφαρμογή κατα την διάρκεια της εγκατάστασης.

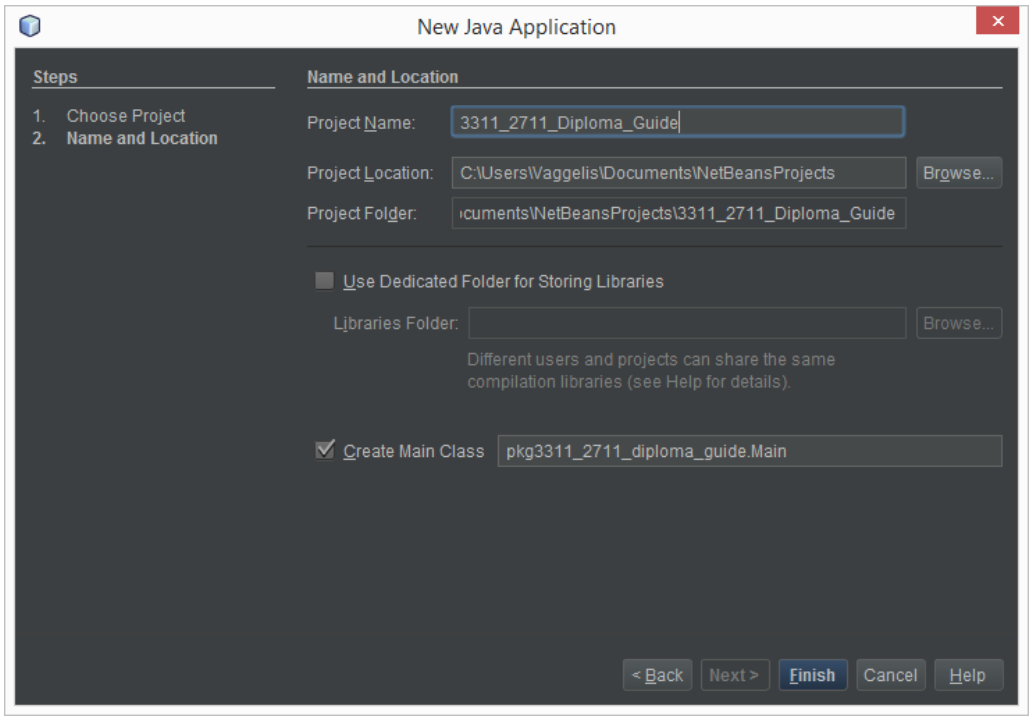
Έπειτα από την εγκατάσταση του εργαλείου, δημιουργήσαμε μία νέα Java εφαρμογή από το μένου "File → New Project", το οποίο θα μας φέρει στο παράθυρο που φαίνεται παρακάτω και θα πρέπει να διαλέξουμε τις επιλογές που φαίνονται στην εικόνα 3.

[2] Java Development Kit



3 Δημιουργία εφαρμογής Java σε Netbeans.

Στην συνέχεια, επιλέξαμε το όνομα της εφαρμογής μας και είμαστε πλέον έτοιμοι για την συγγραφή του κώδικα σε γλώσσα Java.



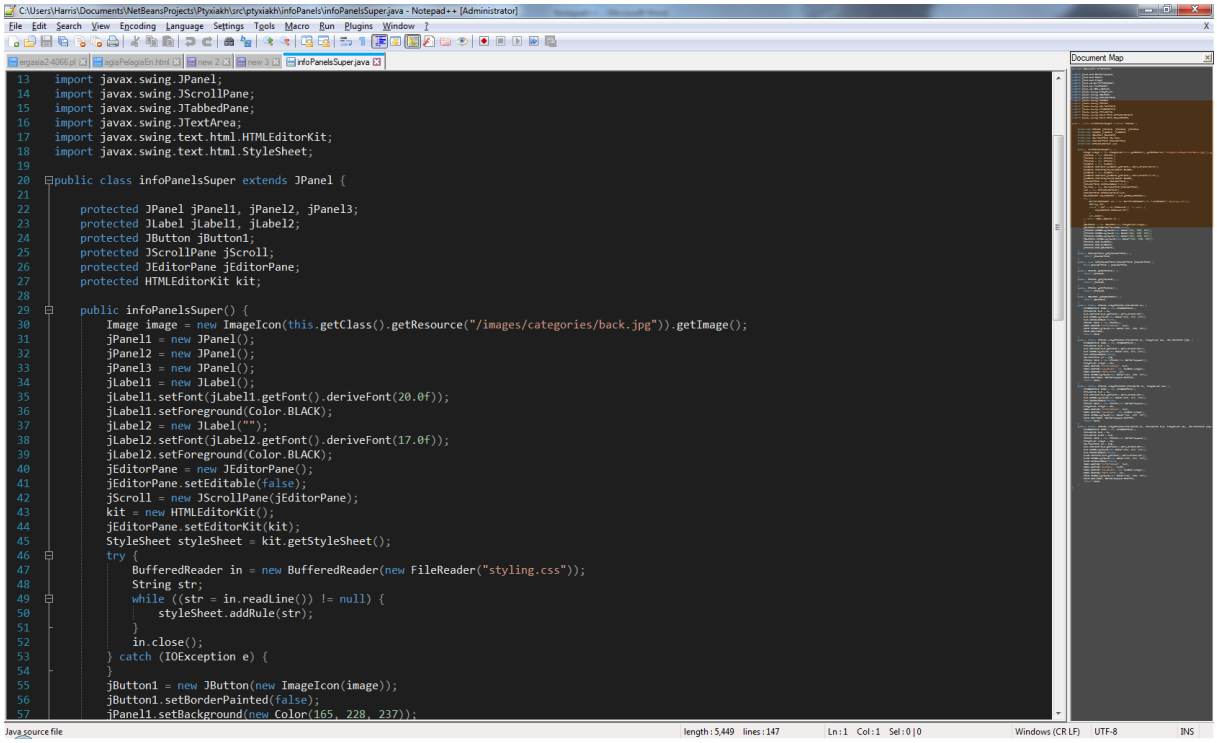
4 Ονομασία και Τοποθεσία εφαρμογής.

2. NOTEPAD ++

Το Notepad++ είναι ένα πρόγραμμα ανοιχτού λογισμικού και πιο συγκεκριμένα ένας text editor, δηλαδή ένα πρόγραμμα σύνταξης κειμένου. Το συγκεκριμένο πρόγραμμα χρησιμοποιείται ευρέως για το λόγο του ότι, είναι αρκετά ελαφρύ και ένας προγραμματιστής μπορεί να αποθηκύσει πολλά μεγάλα κομμάτια κωδικά σε αυτό, χωρίς να χρειάζεται να καταναλώνει πολλούς πόρους. Χρησιμοποιεί και παρέχει βιβλιοθήκες απο πληθώρα γλωσσών προγραμματισμού και αυτό το καθιστά αρκετά χρήσιμο και ευέλικτο.

Το βασικό του μειονέκτημα είναι, ότι όπως περιγράφει και η ονομασία του, είναι ένα απλο notepad, δηλαδή ένα απλό σημειωματάριο. Αυτό σημαίνει ότι δε δίνει την ευχαίρια στο χρήστη, όπως κάνουν άλλοι editors, να "τρέξει" και να ελέγξει τον κώδικα ή το πρόγραμμα του, γι αυτό και χρησιμοποιείται κυρίως ως πρόχειρο ή αποθηκευτικός χώρος (ή back up) για το αρχικό τμήμα κώδικα που χρησιμοποιεί ο προγραμματιστής.

Στο παρακάτω παράδειγμα, υπάρχει κώδικας ανεπτυγμένος στη γλώσσα προγραμματισμού java, στο περιβάλλον του notepad++



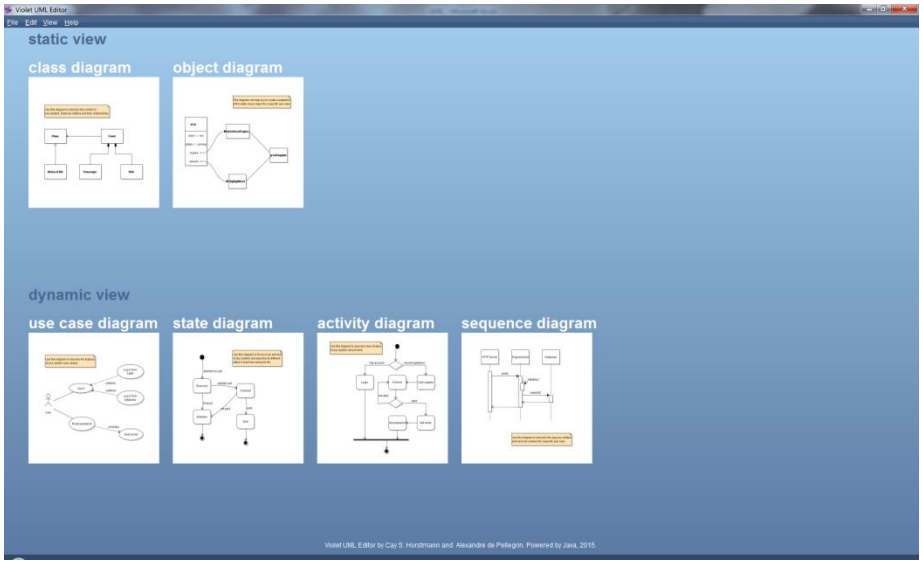
5 Βασικό περιβάλλον του Notepad++.

3. Violet UML

Το Violet UML Editor, είναι ένα πρόγραμμα ανοιχτού λογισμικού και πιο συγκεκριμένα ένας editor γλώσσας UML. Χρησιμοποιείται για τη σωστή απεικόνιση διαγραμμάτων, τα οποία αντιπροσωπεύουν τη δομή και την όλη λειτουργικότητα ενός προγράμματος / έργου ανεπτυγμένο σε λογισμικό υπολογιστή. Το violet απεύθυνεται σε χρήστες τόσο σε επαγγελματίες προγραμματιστές όσο και σε καθηγητές, μαθητές κ.α. , που επιθυμούν να εφαρμόσουν την τεχνική της γλώσσας UML.

Είναι ένα ευκολόχρηστο πρόγραμμα, τόσο όσον αφορά τη χρήση του αλλά και τόσο όσον αφορά την εύρεση και εγκατάστασή του. Το violet UML editor, παρέχεται δωρέαν από την ιστοσελίδα του, από όπου μπορεί ο χρήστης να το κατεβάσει και να το εκτελέσει. Είναι ένα έτοιμο εκτελέσιμο αρχείο, το οποίο δεν απαιτεί εγκατάσταση.

Το violet UML editor παρέχει όλα τα γνωστά, όσον αφορά τη UML, για το χρήστη διαγράμματα, όπως, use-case diagram, class diagram, activity diagram κ.α. Αυτά βρίσκονται στο αρχικό μενού του προγράμματος, δηλαδή ο χρήστης μπορεί να επιλέξει τι θέλει να δημιουργήσει με την εκκίνηση του προγράμματος:



6 Μενού επιλογής διαγράμματος του Violet UML.

Επιλέγοντας το διάγραμμα που επιθυμεί να αναπτύξει, ο χρήστης, έχει στον εργασιακό χώρο που εν συνεχεία του παρέχεται κάθε λογής εργαλείο που επιθυμεί να χρησιμοποιήσει, προκειμένου να καταλήξει στο επιθυμητό αποτέλεσμα, δηλαδή τη σωστή απεικόνιση της δομής του προγράμματός του.

7 Παράδειγμα ένωσης μεταξύ αντικειμένων.

Εν κατακλείδι, πρόκειται για ένα χρήσιμο πρόγραμμα, ικανό να ανταπεξέλθει στις απαιτήσεις των χρηστών του, αλλά και στο να βοηθήσει στην εκμάθηση και εξάσκηση αυτών.

3. ΓΛΩΣΣΕΣ

1. Γλώσσες Προγραμματισμού.

Οι γλώσσες προγραμματισμού, είναι τεχνητά φτιαγμένες, από τον άνθρωπο γλώσσες, οι οποίες χρησιμοποιούνται σε μηχανήματα, κυρίως υπολογιστές. Όπως και οι ανθρώπινες γώσσες, οι γλώσσες προγραμματισμού συντάσσονται και αποτελούνται απο πληθώρα γραμματικών, συντακτικών και πολλών άλλων κανόνων, σύμφωνα με τους οποίους λειτουργούν και παράγουν το αποτέλεσμα για το οποίο έχουν δημιουργηθεί.

Οι γλώσσες αυτές, δημιουργήθηκαν με σκοπό τη διευκόλυνση και τη βελτιστοποίηση της επεξεργασίας των πληροφοριών, που χρησιμοποιούνται από τα μηχανήματα (υπολογιστές) , αλλά και για τη σωστή εκμετάλλευση και επεξεργασία των αλγορίθμων.

Η κάθε γλώσσα ορίζεται ξεχωριστά απο τους δικούς της κανόνες που την καθορίζουν. Έτσι λοιπόν χωρίζονται και στις ανάλογες κατηγορίες μεταξύ τους, όπου κάθε κατηγορία γλώσσων προγραμματισμού, αντιπροσωπεύει το σκοπό για τον οποίο έχουν δημιουργηθεί οι γλώσσες που ανήκουν σε αυτήν και τι αποτελέσματα μπορούν να επιφέρουν. Οι κατηγορίες αυτές, είναι οι εξης:

- **Διαδικαστικές**: Σε αυτή την κατηγορία, η γλώσσα έχει την ικανότητα να μοιράζει και να οργανώνει το πρόγραμμα σε πολλές διαδικασίες, με αποτέλεσμα την καλύτερη και πιο οργανωμένη δομή του.
- **Αντικειμενοστραφεις**: Σε αυτή την κατηγορία, η γλώσσα προγραμματισμού οργανώνει το πρόγραμμα της με το να το διασπά σε περαιτέρω αντικείμενα, τα οποία συνδέονται και επεξεργάζονται τα δεδομένα μεταξύ τους.
- **Συναρτησιακές**: Σε αυτή την κατηγορία, η γλώσσα προγραμματισμού οργανώνει το πρόγραμμα της με ένα αντίστοιχο τρόπο με αυτόν της αντικειμενοστραφούς, με τη διαφορά ότι, αντί τη διαίρεση του προγράμματος σε περαιτέρω αντικείμενα, το διασπά σε πολλές συναρτήσεις, όπου η κάθε μία συνάρτηση αντιπροσωπεύει και μία διαφορετική ενέργεια ή κατάσταση. Έτσι, με τον τρόπο αυτό, όταν το πρόγραμμα θέλει να χρησιμοποιήσει ή να καλέσει μια ενέργεια, χρησιμοποιεί την αντίστοιχη συνάρτηση.
- **Γενικών εφαρμογών**: Πρόκειται για τις γλώσσες προγραμματισμού, που αναπτύχθηκαν με στόχο την ικανοποίηση προγραμμάτων γενικού περιεχομένου, όπως για παράδειγμα γλώσσες που χρησιμοποιούνται ως εκπαιδευτικές.

- **Προγραμματισμού Συστημάτων**: Πρόκεται για τις γλώσσες προγραμματισμού που χρησιμοποιούνται για την ανάπτυξη λογισμικού συστημάτων, όπως λειτουργικών συστημάτων ή υλικών μηχανής.
- **Σεναρίου**: Σε αυτή την κατηγορία, οι γλώσσες έχουν κυρίως ως στόχο τη δημιουργία μικρών και απλών εφαρμογών, μικρών απαιτήσεων και απλών λειτουργιών. Αυτό έχει ως αποτέλεσμα τη μεγάλη ταχύτητα εκτέλεσης και υλοποίησης των προγραμμάτων αυτών.
- **Ειδικών εφαρμογών**: Στην περίπτωση αυτή, οι γλώσσες προγραμματισμού δημιουργήθηκαν με σκοπό την ικανοποίηση και ανάπτυξη συγκεκριμένων προγραμμάτων για συγκεκριμένους σκοπούς.

Μερικές από τις βασικές γλώσσες προγραμματισμού που χρησιμοποιούνται μέχρι και σήμερα, είναι οι εξής: C, C++, C#, Java, Javascript, Matlab, PHP, Python, SQL, Visual Basic.

2. Αντικειμενοστραφής Προγραμματισμός.

Ο αντικειμενοστραφής προγραμματισμός ειναι ένας τρόπος προγραμματισμού ή οπως αποκαλείται, ένα "προγραμματιστικό παράδειγμα", κατά το οποίο οι προγραμματιστές καλούνται να δηλώσουν τον τύπο των δεδομένων καθώς και την δομή αυτών των δεδομένων. Έτσι, το αποτελέσμα καλείται ως "αντικείμενο", το οποίο περιλαμβάνει όλη την αναγκαία πληροφορία που χρειαζόμαστε για αυτο, όπως τα δεδομένα του και τις συναρτήσεις με τις οποίες διαχειρίζεται αυτά τα δεδομένα.

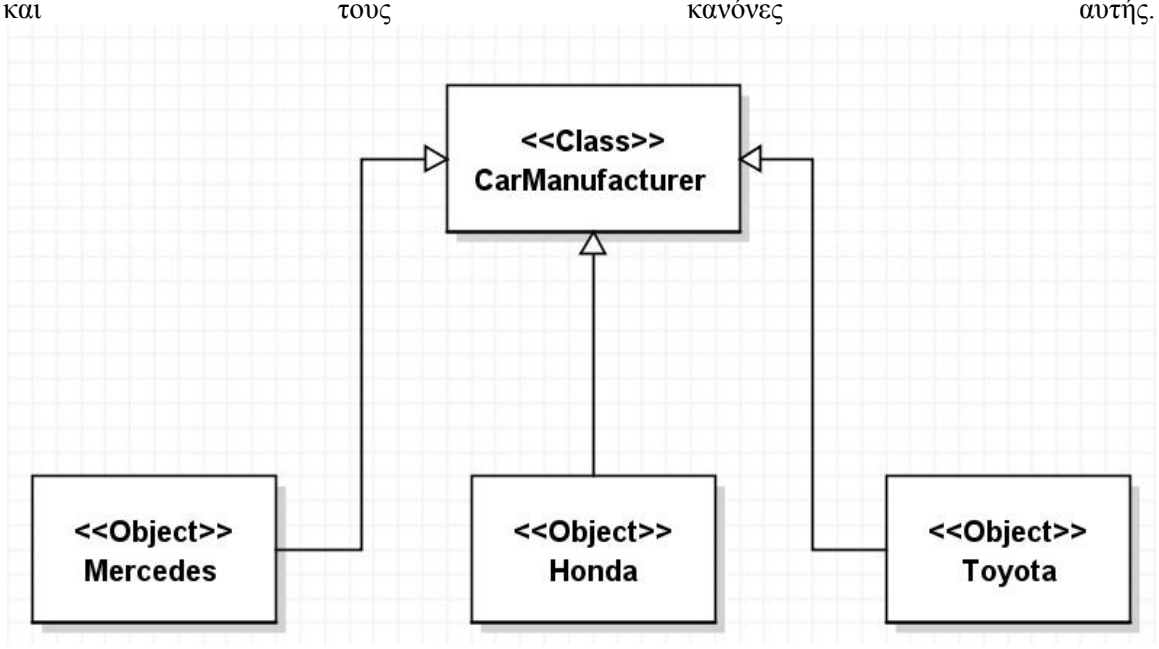
Η ιδέα του αντικειμενοστραφή προγραμματισμού έκανε την πρώτη του εμφάνιση στο MIT[3] μεταξύ του 1950 και 1960 ως μέρος του περιβάλλοντος μίας ομάδας που ασχολήθηκε με την τεχνητή νοημοσύνη. Η επίσημη μορφή του ήρθε με την γλώσσα "Simula 67", η οποία κατασκευάστηκε από τους Ole-Johan Dahl και Kristen Nygaard στο Νορβηγικό υπολογιστικό κέντρο.

Τα βασικά στοιχεία που ορίζονται στον αντικειμενοστραφή προγραμματισμό και τα οποία καλείται ο χρήστης-προγραμματιστής να κατανοήσει πλήρως ώστε κάθε πρόγραμμα να ειναι κατανοήσιμο, αξιόπιστο και κυρίως εύκολα χρησιμοποιήσιμο απο άλλους χρήστες, είναι τα εξής :

- Κλάσεις(Classes) - Μία κατηγορία αντικειμένων η οποία ορίζεται ώς τη διάταξη των δεδομένων και των διαθέσιμων συναρτήσεων -γνωστές καί ως μέθοδοι- που μπορεί να χρησιμοποιήσει αυτό το αντικείμενο.

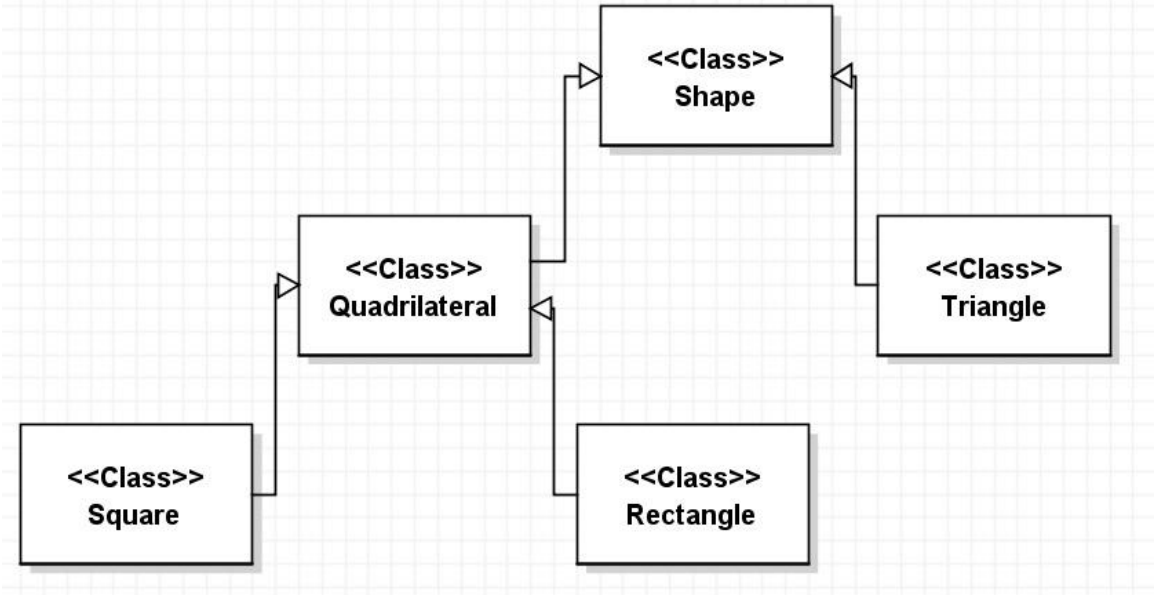
[3] Το "Massachusetts Institute of Technology" είναι ιδιοτικό,ερευνητικό πανεπιστήμιο στο Cambridge, Massachusetts.

- Αντικείμενα(Objects) - Ένα στιγμιότυπο κάποιας κλάσης που υπακούει πλήρως στα δεδομένα και τους κανόνες αυτής.



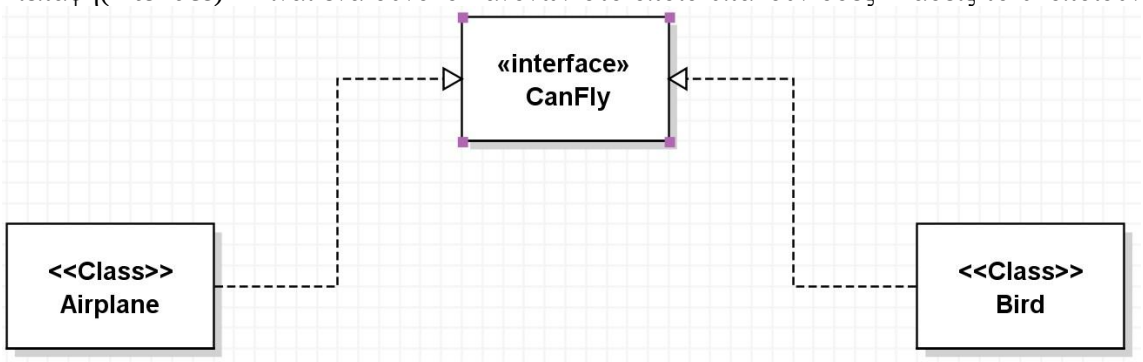
8 Παράδειγμα αντικειμένων.

- Κληρονομικότητα(Inheritance) - Την σχέση μεταξύ κλάσεων. Μία κλάση μπορεί να κληρονομεί στοιχεία κάποιας άλλης κλάσης, το οποίο σημαίνει ότι μπορεί να εκτελέσει μεθόδους και να διαχειριστεί πληροφορίες αυτής της κλάσης χωρίς να χρειάζεται να δηλωθούν στην ίδια.



9 Παράδειγμα κλάσεων.

- Διέπαφή(Interface) - Είναι ένα σύνολο κανόνων στο οποίο υπακούν όσες κλάσεις το υλοποιούν.



10 Παράδειγμα διεπαφών.

- Ενθυλάκωση δεδομένων(Encapsulation) - Είναι ο τρόπος με τον οποίο συνδέονται όλα τα στοιχεία μέσα σε μία κλάση.
- Απόκρυψη πληροφοριών(Information Hiding) - Είναι ο τρόπος σύνταξης των στοιχείων κάθε κλάσης έτσι ώστε να μην είναι προσβάσιμα και διαχειρήσιμα από άλλα, εξωτερικά, κομμάτια κώδικα ενός προγράμματος.

3. Java

Η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού, η οποία βασίζεται στις κλάσεις καί στα αντικείμενα. Είναι σχεδιασμένη για ευκολότερη μάθηση σε σχέση με άλλες αντικειμενοστραφής γλώσσες,Είναι σχεδιασμένη,ώστε να παρέχει πλεονεκτήματα σε όσους την χρησιμοποιούν και σε όσους την αναπτύσουν. Μερικά πλεονεκτήματα της είναι τα εξής:

- **Φορητότητα**: Ο πηγαίος κώδικας δεν χρειάζεται να συνταχθεί(compile) κάθε φορά που χρησιμοποιείται αλλά μία φορά σε μορφή Bytecode το οποίο κατανοείται από όλα τα συστήματα JVM, το οποίο έχει σαν αποτέλεσμα να τρέχει σχεδόν σε όλα τα συστήματα, εφ'όσων αυτά την υποστηρίζουν, δηλαδή έχουν κάποιο σύστημα στο δίκτυο (Server ή Client) το οποίο έχει Java Virtual Machine(JVM).
- **Επεκτατικότητα**: Η Java είναι αντικειμενοστραφής γλώσσα το οποίο σημαίνει ότι οποιοςδήποτε χρειαστεί να προσθέσει ή να διορθώσει κάποιο κομμάτι κώδικα δεν χρειάζεται να τον κατανοήσει ολόκληρο αλλά να κατευθυνθεί αμέσως στο αντικείμενο που χρειάζεται καί και να ενεργήσει ανάλογα.
- **Java Applet:** Τα Java Applets είναι μικρές εφαρμογές οι οποίες είναι ανεπτυγμένες σε Java ή καί άλλες γλώσσες προγραμματισμού όπου στο τέλος συντάσσονται σε Java Bytecode. Αυτές υλοποιούνται μέσα από αλλες εφαρμογές και ο λόγος ύπαρξης τους είναι η πολύ μεγάλη ταχύτητα εκτέλεσης για μία συγκεκριμμένη διαδικασία. (Παράδειγμα:)

```
import java.applet.*;
import java.awt.*;


//Applet code for the "Hello, World!" example.
//This should be saved in a file named as "HelloWorld.class".
```

```
public class HelloWorld extends applet {


   public void paint(Graphics g){
      //Print a message on the screen (x=20, y=10).
      g.drawString("Hello World", 20, 10);

      //Draws a circle on the screen (x=40, y=30).
      g.drawArc(40, 30, 20, 20, 0, 360);

      //Draws a rectangle on the screen (x1=100, y1=100, x2=300, y2=300).
      g.drawRect(100, 100, 300, 300);

      //Draws a square on the screen (x1=100, y1=100, x2=200, y2=200).
      g.drawRect(100, 100 ,200, 200);
   }

}
```

```
<!DOCTYPE html>
<html>
<head>
   <title>HelloWorld_Example.html</title>
</head>
<body>
   <h1>A Java applet example</h1>
   <p>Here it is: <applet code="HelloWorld.class" height="40" width="200">
      this is where HelloWorld.class runs.
      </applet></p>
</body>
</html>
```

Το 1991 ο James Gosling, ο Patrick Naughton και ο Mike Sheridan ξεκίνησαν να γράφουν μία καινούρια γλώσσα προγραμματισμού διότι εκείνο τον καιρό, ήταν απασχολημένοι με την δημιουργία μίας διαδραστικής τηλεόρασης καί είχαν απογοητευτεί από τις γλώσσες που είχαν στην διάθεση τους τότε.

Στην αρχή η Java είχε το όνομα "Oak" το οποίο πήρε από μία βελανιδία που βρισκόταν έξω από το σπίτι του Gosling. Αργότερα μετονομάστηκε σε "Green" διότι το "Oak" ήταν κατοχυρωμένο πνευματικά από μία άλλη εταιρεία, την "Oak Technologies". Τελικά ονομάστηκε "Java" από το νησί της ινδονησίας όπου πρωτοεμφανίστηκε ο καφές καί ήθελαν να θυμήσει την πρωτοτυπία ανάμεσα στις υπόλοιπες γλώσσες.

Η Java αναπτύχθηκε από τους προγραμματιστές που αναφέραμε προηγουμένως, ενώ ήταν μέλη της "Sun Microsystems". Αργότερα η εταιρεία εξαγοράστηκε από την "Oracle Corporation", μία πολυεθνική εταιρεία τεχνολογίας.

Παρακάτω, θα παρουσιάσουμε παραδείγματα για τα βασικά κομμάτια της Java έτσι ώστε να την κατανοήσουμε καί τελικά να κατανοήσουμε την εργασία που παρουσιάζεται.

Ξεκινώντας, θα κοιτάξουμε την σύνταξη των βασικών εννοιών της Java :

- **Κλάσεις** καί τρόπος δήλωσης μεταβλητών κλάσης(Class variables) και μεθόδων.

```
package javatutorial;

public class Car {

   String name;
   int gear;
   double speed;

   void Accelerate() {
   }

   void SlowDown() {
   }

}
```

- **Constructor** : Είναι μία ιδιαίτερη μέθοδος η οποία χρησιμοποιείται για την αρχικοποίηση του αντικειμένου, καθώς και για την αρχικοποίηση άλλων παραμέτρων μέσα στο αντικείμενο.

```
public class Car {

   String name;
   int gear;
   double speed;


   public void car(String Cname) {
      name = Cname;
   }
```

Παρατήρηση! Ένα αντικείμενο μπορεί να έχει περισσότερους από έναν constructors, με διαφορετική παραμετροποίηση ο καθένας από αυτούς.

- **Κληρονομηκότητα :** Παρακάτω βλέπουμε τον τρόπο με το οποίο δηλώνεται στην Java η κληρονομηκότητα, δηλαδή έχει ήδη τα χαρακτηριστικά μίας άλλης κλάσης.

```
public class Toyota extends Car {

   public void Toyota(){
      this.name = "Corolla";
   }

}
```

Βλέπουμε επίσης ότι η νέα κλάση μπορεί να διαχειριστεί μεταβλητές κλάσης(Class Variables) χωρίς να χρειάζεται να τις δηλώσει εφ'όσων τις κληρονομεί από την άλλη κλάση.

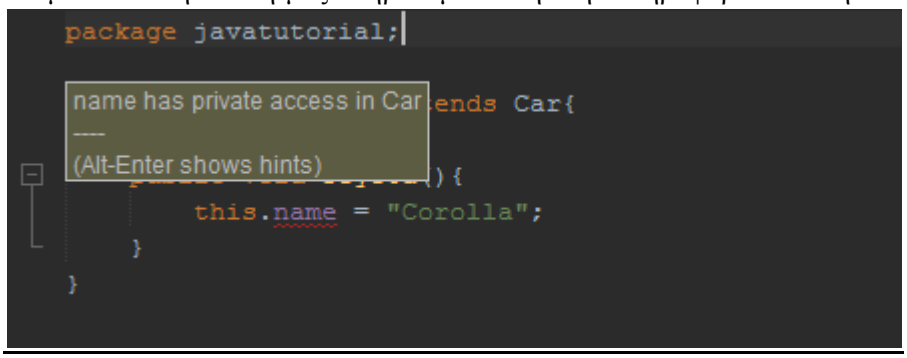
- **Απόκρυψη πληροφορίας (Data Hiding):** Χρησιμοποιώντας την λέξη "private" δηλώνουμε μία μεταβλητή ως ιδιοτική, το οποίο δίνει την δυνατότητα στο αντικείμενο μας να προστατευει τις πληροφορίες από να υπόκεινται σε αλλαγές από μεθόδους εκτός του ίδιου του αντικειμένου.

```
public class Car {

   private String name;
   private int gear;
   private double speed;


   public void car(String Cname) {
      name = Cname;
   }
```

- Όπως φαίνεται λοιπόν παρακάτω δεν έχουμε πλέον πρόσβαση στην μεταβλητή "name" ακόμα και άν η κλάση μας κληρονομεί αυτήν την πληροφορία από την κλάση "γονέα".



11 **Μήνυμα λάθους του Netbeans.**

4. HTML

Η γλώσσα προγραμματισμού html αναπτύχθηκε το 1990, βασισμένη στην τεχνική της SGML, η οποία έχει ως στόχο τη σωστή διαχείριση διαδικτυακών εγγράφων. Η html δημιουργήθηκε με αρχικό σκοπό, να παρέχει στους χρήστες, που δεν είχαν τις απαιτούμενες γνώσεις για να χρησιμοποιήσουν την τεχνική SGML, την ευκαιρία να μπορούν να διαχειριστούν και να δημοσιεύσουν διαδικτυακά έγγραφα που επιθυμούν ή θέλουν να αναπτύξουν.

Λόγω του ότι η γλώσσα ήταν αρκετά εύκολη στο να διδαχθεί και να χρησιμοποιηθεί, αλλά και λόγω της ικανότητας της, να μπορεί να αυτοσυντηρείται, γρήγορα άρχισε να χρησιμοποιείται για την ανάπτυξη πολλών εφαρμογών.

Η html, είναι μία γλώσσα προγραμματισμού με κύριο τομέα, αυτόν του διαδικτύου. Τα αρχικά της λέξης αυτής προέρχονται από τα αγγλικά "hyper text markup language" και στα ελληνικά "γλώσσα σήμανσης υπερκειμένου". Είναι η πλέον βασική και κύρια γλώσσα προγραμματισμού, για την ανάπτυξη περιβάλλοντος μίας ιστοσελίδας.

Η μορφή σύμφωνα με την οποία η html αναπτύσσεται και γράφεται, αποτελείται από πολλά στοιχεία όπως: ετικέτες, οι οποίες αντιπροσωπεύουν διάφορους τύπους στοιχείων (components), τα οποία

προσδιορίζουν και αποτελούν κομμάτια του σκελετού της ιστοσελίδας. Οι ετικέτες απαρτίζονται από τα σύμβολα (μεγαλύτερο και μικρότερο από (< >)) όπως για παράδειγμα
. Συνήθως οι ετικέτες αυτές λειτουργούν με την εξής λογική: Ξεκινώντας ένα τύπο στοιχείου που θέλουμε να κάνει κάτι, ανοίγουμε την πρώτη ετικέτα η οποία αντιπροσωπεύει την έναρξη, γράφουμε το κομμάτι του κώδικα που επιθυμούμε και κλείνουμε την ετικέτα. Η έναρξη και η λήξη δια χωρίζονται ως εξής : < > και </ > αντίστοιχα.

Τα έγγραφα, τα οποία παράγονται, μέσω της HTML, διαβάζονται από τους περιηγητές διαδικτύου (web browsers).

Η βασική δομή που αποτελεί το σκελετό από την αρχή έως το τέλος ενός html εγγράφου γίνεται με τον εξής τρόπο: Ως αρχή του εγγράφου δηλώνουμε ένα tag ως <html> για να συμβολίσουμε τον τύπο του εγγράφου, στη συνέχεια το δεύτερο tag που δηλώνουμε είναι το <body> , το οποίο αντιπροσωπεύει ότι και η λέξη, δηλαδή το σώμα του εγγράφου. Μέσα στο "body" δηλώνουμε και αναπτύσουμε όλο το περιεχόμενο του εγγράφου html, δηλαδή κομμάτια text παραγράφων, τίτλων, γραφικού περιβάλλοντος και πολλά άλλα. Όταν θέλουμε να " κλείσουμε " το "body" και να φτάσουμε το έγγραφο στο τέλος, κλείνουμε το body ως εξής </body> . Ένα παράδειγμα κώδικα html εγγράφου, το οποίο περιέχει ένα τίτλο και μία παραγραφο, είναι το εξής:

```
<html>
<body>
<h1> Title </h1>
   <p>
      Paragraph Info
   </p>
</body>
</html>
```

Επίσης, μέσα στο έγγραφο της html, μπορούν να δηλωθούν και πολλά άλλα στοιχεία, προέρχομενα από άλλους τύπους εγγράφων και κατα συνέπεια γραμμένα και ανεπτυγμένα από άλλες γλώσσες προγραμματισμού. Σύνηθες παράδειγμα αυτού, είναι η δήλωση javascript εγγράφων μέσα στο έγγραφο html που αναπτύσσουμε, αλλά και εγγράφων css. Η javascript βοηθάει στην υλοποίηση της λειτουργικότητας της Html και η Css στη διαμόρφωση του γραφικού περιβάλλοντος, γνωστό ως τεχνική styling, της html.

Υπάρχουν τρεις τρόποι, σύμφωνα με τους οποίους η html μπορεί να χρησιμοποιήσει έγγραφα άλλων γλωσσών. Ο ένας είναι, να δημιουργηθεί τοπικά το έγγραφο της άλλης γλώσσας που θέλουμε να δανειστούμε, μέσα στο ίδιο το έγγραφο της html. Ο άλλος τρόπος είναι, να δημιουργήσουμε ξεχωριστά το έγγραφο της άλλης γλώσσας και απλώς να το δηλώσουμε μέσα στο έγγραφο της html, προκειμένου να το αναγνωρίσει. Στον τρίτο τρόπο δεν δημιουργούμε κάποια ξεχωριστό έγγραφο όπως στις δύο προηγούμενες περιπτώσει, αλλά δηλώνουμε για το κάθε στοιχείο ξεχωριστά, τοπικά μέσα σε αυτό, αυτό που θέλουμε να προσδιορίσουμε. Για παράδειγμα θέλουμε να δώσουμε χρώμα σε έναν τίτλο στο έγγραφο html, μέσα στην αρχή του tag, γράφουμε style = "" και μέσα στα αυτάκια αυτό που θέλουμε να δώσουμε, στην προκειμένη χρώμα, έστω άσπρο. Άρα θα δηλωθεί ως εξής: <h1 style="color: white;"> Title </h1>.

Στην πρώτη περίπτωση, δημιουργούμε το έγγραφο μέσα στην html, ακριβως πριν το body. Δημιουργούμε πρώτα ένα νέο tag σε αυτη την περίπτωση, το <head> , το οποίο αντιπροσωπεύει ότι και η λέξη, δηλαδή το κεφάλι του εγγράφου της html. Είναι δηλαδή το τι θα δηλώσουμε αρχικά, προκειμένου να χρησιμοποιήσουμε για το χτίσιμο του σώματο "body" και στο τέλος (πριν αρχίσει το body) κλείνουμε το head με το tag </head>. Στο επόμενο παράδειγμα θα χρησιμοποιήσουμε τη γλώσσα προγραμματισμού CSS για να εφαρμόσουμε "styling" στον τίτλο του εγγράφου. Αυτό γίνεται ως εξής. Ακριβώς μετα το tag

που ανοίγει το κεφάλι (<head>) δηλώνουμε το tag <style>, το οποίο αντιπροσωπεύει την έναρξη του εγγράφου CSS. Μέσα σε αυτό το πεδίο δηλώνουμε τα στοιχεία που θέλουμε κάνουμε styling, στην προκειμένη τον τίτλο που συμβολίζεται με το tag <h1>. Μέσα στο css το δηλώνουμε με το ίδιο το όνομα του tag, δηλαδή h1 (προκειμένου το body της Html να αναγνωρίσει το αντίστοιχο στοιχείο) ως εξής: h1 {..} , όπου μέσα στις αγκύλες δηλώνουμε από τι αυτό θα εμπεριέχεται (στην προκειμένη, το χρώμα, έστω άσπρο) και τέλος κλείνουμε το styling με το tag </style>. Όλη η δομή του html εγγράφου λοιπόν αυτού, θα έχει ως εξής:

```
<html>

<head>
   <style> h1 { Color: white; } </style>
</head>
```

Στη δεύτερη περίπτωση, δημιουργούμε ξεχωριστά το έγγραφο που θέλουμε να δανείσουμε σε αυτό της html, έστω ενα CSS και δηλώνουμε την ύπαρξη του μέσα στην html προκειμένου να αναγνωρίσει τα στοιχεία τα οποία θέλουμε να της δανείσουμε. Για να το επιτύχουμε αυτό, όπως προαναφέρεται και στην παραπάνω περίπτωση, δηλώνουμε μέσα στο script της CSS το όνομα του στοιχείου που θέλουμε να επεξεργαστούμε, ακριβώς με το ίδιο όνομα του tag της html που το χαρακτηρίζει. Δηλαδή όπως θα δούμε στο επόμενο παράδειγμα, θέλουμε ο τίτλος του html εγγράφου να είναι λευκός και χρησιμοποιούμε για αυτόν, το tag <h1>, άρα στο css, όπως και στο παραπάνω παράδειγμα θα το δηλώσουμε ως h1 { color : white; }. Για να γίνει η δήλωση του εγγράφου μέσα στην html, δηλώνουμε μέσα στο head την εξής γραμμη <link rel="stylesheet" type = "text/css" href="teststyle.css">. Αναλυτικά για αυτή την εντολή: Δηλώνουμε ότι το έγγραφο που δανειζόμαστε είναι τύπου "stylesheet" με το rel="stylesheet", δηλώνουμε ότι είναι τύπου κειμένου (text) css με το type = "text/css" και τέλος δηλώνουμε ποιο είναι αυτό το έγγραφο που θα δανειστούμε δίνοντας τον τύπο και το όνομα του, που σε αυτή την περίπτωση είναι το teststyle τύπου css, άρα href = " teststyle.css ". Αναλυτικά ο κώδικας στο παράδειγμα:

```
<html>
<head>
   <link rel="stylesheet" type="text/css" href="teststyle.css">
</head>
```

Στην τρίτη περίπτωση, δε δημιουργούμε κανένα ξεχωριστό έγγραφο, ούτε τοπικά μέσα στην html αλλά ούτε ως ξεχωριστό, όπως στις 2 προηγούμενες περιπτώσεις. Εδώ δηλώνουμε ξεχωριστά για το κάθε στοιχείο αυτά που θέλουμε να του προσδιορίσουμε, μέσα στο ίδιο το tag του. Δηλαδή, στο παράδειγμα το οποίο αναλύουμε, θέλουμε να δώσουμε ένα τίτλο του εγγράφου και να του δώσουμε λευκό χρώμα. Σε αυτή την περίπτωση, το επιτυγχάνουμε αυτό δηλώνοντας μέσα στο tag το ίδιο το style, δηλαδή ως εξής : <h1 style="color: white;"> Title </h1>.

Μέσα στον κώδικα το βλέπουμε αυτό στην παρακάτω εικόνα, ολοκληρωμένο:

```
<body>
<h1 style="color:white;"> Title </h1>
   <p>
      Paragraph Info
```

```
   </p>
</body>
```

Η τελευταία αυτή περίπτωση όμως, δεν ενθαρύνεται γενικά, λόγω του ότι χρησιμοποιεί και καταναλώνει πολλούς πόρους χωρίς λόγο, για το λόγω της συχνής αναφοράς του ίδιου στοιχείου.

Άρα, εν κατακλείδι, η γλώσσα html μπορεί να χρησιμοποιηθεί πολύ εύκολα για τη δημιουργία της όλης εικόνας και του γραφικού περιβάλλοντος μιας εφαρμογής, αλλά παράλληλα να βοηθηθεί από έγγραφα άλλων γλωσσών για περαιτέρω λειτουργίες.

Ένα από τα αρκετά εύκολα και "γρήγορα", ανοιχτού λογισμικού, περιβάλλοντα, που μπορεί η γλώσσα (αλλά και πολλές άλλες γλώσσες) να αναπτυχθεί, είναι το notepad++ .

Ένα ολοκληρωμένο παράδειγμα μίας απλής σελίδας αναπτυσσόμενης σε html, με τη βοήθεια των στοιχείων απο άλλες γλώσσες, είναι το παρακάτω:

```
<html>
<head>
   <style>
      h1{
         color: black;
         font-style: oblique;
         font-size: 185%;
         }

      button1 {
         background-color: green; /*Green*/
         color: white;
         text-align: center;
         font-size: 16px;
      }
   </style>
</head>
<body>

<h1> EDW EXOUME TON TITLO</h1>
   <p> Edw </br> vazoume mia aplh </br> paragrafo, me allages </br> grammhs </p>

<button type="button1">Button</button>
</body>
</html>
```

Αναλυτικότερα για το παραπάνω παραδειγμα, έχουμε δηλώσει ένα header μέσα στην html, το οποίο και χρησιμοποιούμε ως τίτλο του εγγράφου μας, μία απλή παράγραφο με αλλαγές γραμμής και ένα απλό κουμπι. Επίσης έχουμε δηλώσει τοπικά το stylesheet μας, όπου καθορίζουμε το πως θα φαίνεται το κουμπί και το header που χρησιμοποιούμε στο body της html.

5. CSS

Η γλώσσα προγραμματισμού CSS, είναι τα αρχικά της αγγλικής πρότασης "Cascading Style Sheets", στα ελληνικά "Αλληλουχία φύλλων στυλ". Αναπτύχθηκε το 1994, με σκοπό την υποστήριξη άλλων προγραμματιστικών γλωσσών, στο θέμα του γραφικού τους περιβάλλοντος και στη βελτιστοποίηση της μορφοποίησης του τελικού αποτελέσματος της εφαρμογής, που οι γλώσσες αυτές αναπτύσσουν.

Δημιουργήθηκε πιο συγκεκριμένα για να υποστηρίξει τη γλώσσα προγραμματισμού HTML στο γραφικό της περιβάλλον, αλλά γενικά για να βελτιώσει και να κάνει ευκολότερο, για τους προγραμματιστές, το περιβάλλον ανάπτυξης λογισμικού για το διαδίκτυο, που αφορά το γραφικό του κομμάτι.

Η γλώσσα προγραμματισμού CSS, είναι μία " βοηθητική", προς άλλες γλώσσες προγραμματισμού, γλώσσα, η οποία περιγράφεται κυρίως με μικρά έγγραφα κειμένου. Χρησιμοποιείται κυρίως για να καθορίσει την περιγραφή εγγράφων κειμένου άλλων, όσον αφορά το γραφικό τους περιβάλλον.

Είναι μία γλώσσα, που δε μπορεί από μόνη της να περιγραφεί ή να χρησιμοποιηθεί κάπου, για το λόγο του ότι αποτελείται από μικρά κομμάτια κώδικα, τα οποία περιγράφουν αντικείμενα άλλων γλωσσών προγραμματισμού και πιο συγκεκριμένα τη μορφοποίηση αυτών. Έτσι λοιπόν, οι γλώσσες αυτές υιοθετούν το αντίστοιχο CSS έγγραφο κειμένου, που θέλουν να χρησιμοποιήσουν, με αποτέλεσμα να βελτιστοποιούν τη μορφοποίηση των αντικειμένων που έχουν δημιουργήθει και η CSS περιγράφει.

Η σύνταξη της γλώσσας είναι αρκετά απλή και κοντά στο χρήστη (user-friendly). Χρησιμοποιεί ως αντικείμενα και συναρτήσεις περιγραφής της, αντίστοιχες αγγλικές λέξεις, οι οποίες αντιπροσωπεύουν τις ιδιότητες που περιγράφουν μέσα στη γλώσσα.

Πιο συγκεκριμένα, η δομή ενός εγγράφου CSS, αποτελείται απο πολλά μικρά τμήματα κώδικα, όπου το καθένα ξεχωριστά αποτελεί το σύνολο των ιδιοτήτων, που καθορίζουν το γραφικό τμήμα του αντικειμένου, που το κομμάτι κώδικα αυτού περιγράφει. Η σύνταξης της περιγραφής αυτής, έχει ως εξής: Δίνουμε πρώτα το όνομα του αντικειμένου που θέλουμε να περιγράψουμε, συγκεκριμένα όμως, το όνομα αυτό που θα δηλώσουμε μεσα στο CSS έγγραφο, πρέπει να είναι ακριβώς το ίδιο με αυτό του αντικειμένου, της γλώσσας που θα το χρησιμοποιήσει. Αν για παράδειγμα θέλουμε να περιγράψουμε ένα header στη γλώσσα προγραμματισμού html, το οποίο περιγράφεται με το tag <h1>, τότε και στο CSS το όνομα του αντικειμένου που θα δηλώσουμε, θα είναι το h1. Στη συνέχεια, αφού δηλώσουμε το όνομα, ανοίγουμε και κλείνουμε αγκύλες. Αν δηλαδή το όνομα είναι το h1, τότε ο κώδικας μετά το δεύτερο βήμα, θα έχει ως εξης : h1 {...}. Το επόμενο μας βήμα είναι να " γεμίσουμε " τις αγκύλες, δηλαδή μέσα σε αυτές τοποθετούμε ένα προς ένα, όλες τις ιδιότητες του αντικειμένου. Μερικές από αυτές μπορούν να είναι, το χρώμα του αντικειμένου, το χρώμα του κειμένου του, η στοίχιση του κειμένου, η στοίχιση του μέσα στη σελίδα και πολλά άλλα. Η σύνταξη των ιδιοτήτων των αντικειμένων στη CSS έχει ως εξής: Πρώτα δηλώνουμε τον τύπο μεταβήτης που θέλουμε να του δώσουμε ιδιότητα, παράδειγμα χρώμα (color), στη συνέχεια πληκτρολογούμε το σύμβολο της άνω κάτω τελειας ":", το οποίο αντιπροσωπεύει ότι μετά από αυτό ακολουθεί η ιδιότητα της μεταβλητής. Στο επόμενο βήμα γράφουμε την ιδιότητα που θα δώσουμε στη μεταβλητή μας και τέλος κλείνουμε την πρόταση με το σύμβολο του ελληνικού ερωτηματικού ";". Για παράδειγμα, θέλουμε να δώσουμε χρώμα κίτρινο σε ένα header ενός html εγγράφου, με όνομα (του header) h1. Θα γράψουμε μέσα στο έγγραφο CSS το εξής: h1 { color: yellow; }

Στο παρακάτων παράδειγμα που ακολουθεί, αναπτύσσουμε κώδικα ενός εγγράφου css (stylesheet), το οποίο μπορεί να χρησιμοποιηθεί από ένα έγγραφο html για τη μορφοποίηση των αντικειμένων του.

```
h1 {
```

```
  color: yellow;
  font-style: oblique;
  font-size: 185%;
  text-align: center;
}

button1 {
  background-color: green;
  color: white;
  text-align: center;
  font-size: 16px;
}

p1 {
  background-color:red;
  font-size: 14px;
}
```

Πιο αναλυτικά, στο παραπάνω παράδειγμα, έχουμε δημιουργήσει 3 αντικείμενα. Ένα για το header h1, ένα για το κουμπί button1 κι ένα για τις παραγράφους p1, ενός εγγράφου html. Ως ιδιότητες των αντικειμένων αυτών, δίνουμε το χρώμα τους "color, background-color", το στυλ της "οικογένειας" (font-style), το μέγεθος (font-size) και τη στοίχιση (text-align) του κειμένου τους.

Το έγγραφο html, για του οποίου τα αντικείμενα δημιουργήθηκε το έγγραφο CSS αυτό, θα υιοθετήσει τις ιδιότητες αυτές και θα τις αποδόσει με τη σειρά του στα δικά του αντικείμενα, προκειμένου να αποδοθεί το επιθυμητό αποτέλεσμα, στην τελική μορφοποίησή τους.

6. UML

Οι γλώσσες μοντελοποίησης, είναι γλώσσες που έχουν δημιουργηθεί και αναπτυχθεί, με σκοπό και στόχο, την έκφραση ενός προγράμματος ή γενικά ενός έργου στην κατασκευαστική του μορφή.

Αναλυτικότερα, μια γλώσσα μοντελοποίησης έχει την ικανότητα να εκφράζει το έργο (για το οποίο χρησιμοποιείται) με τέτοιο τρόπο, ώστε να καταλαβαίνει ο κάθε χρήστης την όλη δομή του (έργου) από την αρχή μέχρι το τέλος. Οι γλώσσες μοντελοποίησης μπορεί να είναι είτε γραφικές ή κειμενικές.

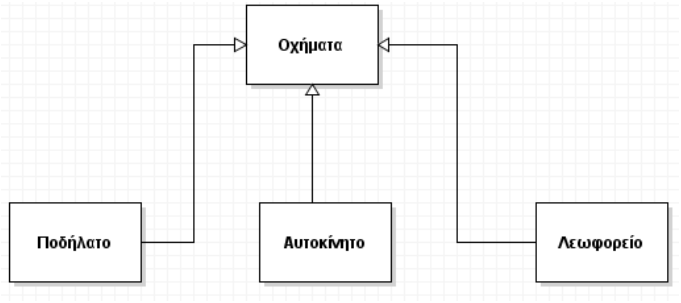
Η δομή των γραφικών γλωσσών έχει ως εξης: Αναλύεται ο σκελετός του πρόγραμμα ως "δέντρο", όπου ουσιαστικά ο κορμός, είναι η βασική δομή του όλου προγράμματος και το κάθε κλαδί οι περαιτέρω λειτουργίες του βάση του κορμού. Άρα, όσο πιο πολύ αναλύεται ένα κλαδί σε περαιτέρω κλαδιά, τόσες περισσότερες λειτουργίες έχει το πρόγραμμα, οι οποίες αναλύονται και εκφράζονται βάση του προηγούμενου κλαδιού στο οποίο ανήκουν. Είναι λειτουργίες δηλαδή, που αποτελούν επεκτάσεις μίας βασικής λειτουργίας. Ένα παράδειγμα γραφικής γλώσσας μοντελοποίησης είναι η UML.

Η UML, αναπτύχθηκε τη δεκαετία του 1990 βασιζόμενη πάνω στη λογική του αντικειμενοστραφούς προγραμματισμού. Προέρχεται από την αγγλική πρόταση Unified Modeling Language και αποτελεί μία γλώσσα μοντελοποίησης. Χρησιμοποιείται στον τομεά μηχανικής λογισμικού με σκοπό τη σωστή απεικόνιση της όλης σχεδίασης και δομής ενός έργου.

Η δομή της UML αναπτύσσεται ως εξής:

Σε πρώτη φάση, το έργο αναλύεται με βάση το περιεχόμενο του, δηλαδή τον κώδικα του προγράμματος, τις κλάσσεις του (αν πρόκειται για αντικειμενοστραφή προγραμματισμό) και τις συνδέσεις μεταξύ τους. Δηλαδή αν έχουμε ένα πρόγραμμα το οποίο αναλύεται σε κλάσεις θα το εκφράσουμε στη UML ως εξής: Στην κορυφή του προγράμματος θα βρίσκεται η αρχική και βασική κλάση (σύμφωνα με την οποία οι υπόλοιπες αναλύονται) και από κάτω της, οι ακριβώς επόμενες κλάσεις οι οποίες αποτελούν επεκτάσεις αυτής και ούτω καθεξής. Οι κλάσεις εκφράζονται ως τετράγωνα (blocks) μέσα στο πρόγραμμα και αυτό ονομάζεται "class diagram". Οι συνδέσεις μεταξύ τους γίνονται με βέλη, τα οποία ως κατάληξη έχουν την κλάση η οποία αποτελεί προηγούμενο κλαδί, ή αλλιώς βάση αυτής που το βέλος ξεκινάει. Επίσης τα βέλη μπορούν να συνδέσουν και μία ιδιότητα μιας κλάσης με την ιδιότητα μίας άλλης, με σκοπό να δείξουν ότι η ιδιότητα αυτή έχει παρθεί και χρησιμοποιείται και από τη δεύτερη κλάση.

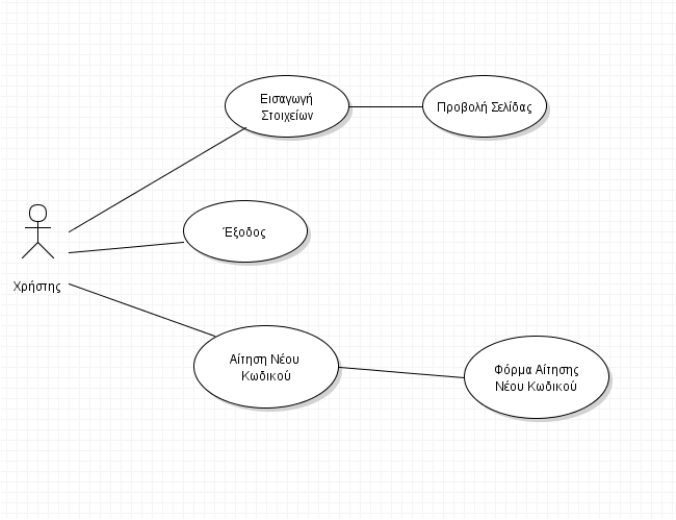
Στο παρακάτω παράδειγμα έχουμε ένα πρόγραμμα το οποίο εκφράζεται ως εξής: Έχουμε μία βασική κλάση την οποία ονομάζουμε " όχημα" και έχει μέσα της κάποιες ιδιότητες. Τις ιδιότητες αυτές κληρονομούν οι κλάσεις "παιδιά" της, οι οποίες βρίσκονται από κάτω της και ονομάζονται, "ποδήλατο", "αυτοκίνητο", "λεωφορείο" αντίστοιχα. Η έκφραση του προγράμματος αυτού, όσον αφορά το class diagram θα έχει ως εξής:



12 Παράδειγμα σύνδεσης αντικειμένων.

Σε δεύτερη φάση, το έργο αναλύεται από την πλευρά του χρήστη και πως αυτός αλληλεπιδρά με το πρόγραμμα. Σε αυτή τη φάση η uml εκφράζεται ως εξής: Ως αφετηρία τοποθετούμε το μοντέλο που αντιπροσωπεύει το χρήστη (οποιουδήποτε ρόλου) και στη συνέχεια συνδέουμε τις λειτουργικότητες που του παρέχει το πρόγραμμα και όπου αυτός αλληλεπιδρά. Είναι δηλαδή η λειτουργικότητα του προγράμματος από τα "μάτια" του χρήστη. Η έκφραση του προγράμματος αυτού, ονομάζεται "use-case diagram". Οι συνδέσεις μεταξύ των αλληλεπιδράσεων γίνονται με μία γραμμή, η οποία ξεκινάει από την αρχή και καταλήγει στο τέλος της αλληλεπίδρασης.

Στο παρακάτω παράδειγμα, έχουμε ένα απλό πρόγραμμα το οποίο αποτελεί την πρώτη εικόνα εισόδου σε μία σελίδα, γνωστό ως login screen. Εδώ ο χρήστης έχει 3 επιλογές, να εισαγάγει τα στοιχεία του προκειμένου να συνδεθεί στη σελίδα, να εξέλθει από το πρόγραμμα ή να ζητήσει νέο κωδικό. Στην πρώτη περίπτωση, αν τα στοιχεία είναι σωστά, ο χρήστης εισέρχεται στη σελίδα. Στη δεύτερη περίπτωση εξέρχεται από το πρόγραμμα και στην τρίτη παραπέμπεται στη σελίδα για αίτηση νέου κωδικού. Η έκφραση του προγραμματος αυτού σε UML, όσον αφορά το use-case diagram θα έχει ως εξής:

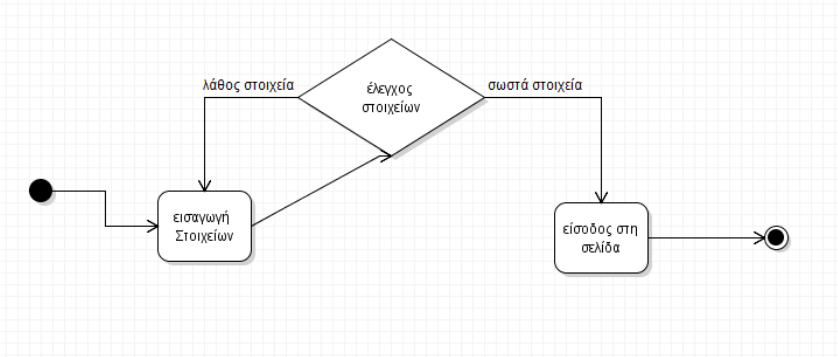


13 Απεικόνιση επιλογών του χρήστη.

Σε τρίτη φάση, η UML αναλύει την κάθε λειτουργία / διαδικασία του έργου από την αρχή της μέχρι το τέλος της, παραθέτοντας όλες τις ενδιάμεσες καταστάσεις και περιπτώσεις που υπάρχουν ή μπορούν να προκύψουν. Η έκφραση του προγράμματος αυτού, ονομάζεται "activity diagram".

Το "activity diagram" εκφράζεται ως εξής: Έναρξη της διαδικασίας αποτελεί μια μαύρη κουκίδα. Στη συνέχεια συνδέεται με ένα βέλος με την επόμενη κατάσταση που προκύπτει, το οποίο καταλήγει στην επόμενη κατάσταση η οποία εκφράζεται ως ένα τετράγωνο (block). Σε περίπτωση που χρειαστεί να παρθεί κάποια απόφαση σε κάποια κατάσταση, το αντίστοιχο βέλος "δείχνει" στην απόφαση αυτή όπου από την οποία, με τον ίδιο τρόπο καταλήγουμε στις επόμενες καταστάσεις. Τέλος για να καθορίσουμε το τέλος της διαδικασίας χρησιμοποιούμε μια κουκίδα αντίστοιχη αυτής της έναρξης.

Κοιτώντας το παραπάνω παράδειγμα (σε use-case diagram) παρατηρούμε ότι μία διαδικασία είναι αυτή της εισαγωγής στοιχείων. Μία διαδικασία όπως αυτή, έχει ενδιάμεσες καταστάσεις και περιπτώσεις μέχρι να φτάσει στο τέλος της. Ξεκινάει από τη στιγμή που δίνει τα στοιχεία ο χρήστης, το πρόγραμμα ελέγχει αν αυτά είναι σωστά (απόφαση) και καταλήγει στο να του επιτρέψει την είσοδο στη σελίδα ή να τον επιστρέψει στην αρχική του κατάσταση. Η έκφραση του προγραμματος αυτού σε UML, όσον αφορά το "activity diagram" έχει ως εξής:



14 Παράδειγμα ενός Activity Diagram.

4. ΚΥΡΙΟ ΜΈΡΟΣ.

Προκειμένου να ξεκινήσει μία κλάση να αναπτύσσεται και να εκφράζεται μέσω κώδικα, θα πρέπει πρώτα να δηλωθούν στην αρχή και να "φορτωθούν" οι βιβλιοθήκες που θα χρησιμοποιηθουν. Δηλαδή, κάθε κλάση χρησιμοποιεί βιβλιοθήκες, που είναι ήδη προεγκατεστημένες στη γλώσσα προγραμματισμού που χρησιμοποιούμε (στην προκειμένη είναι η java) και μας παρέχουν τις ιδιότητες που υπάρχουν μέσα σε αυτές.

Ο τρόπος που δηλώνουμε μια βιβλιοθήκη σε μία κλάση είναι με τη λέξη κλειδί "import" και δίπλα το όνομα της βιβλιοθήκης. Αν θέλουμε να δηλώσουμε ένα ολόκληρο τμήμα βιβλιοθηκών που ανήκουν στην ίδια κλάση, αντί να τις δηλώνουμε μία μία, δηλώνουμε το όνομα της κλάσης, στη συνέχεια ακολουθεί το σύμβολο της τελείας "." και μετά το σύμβολο "*". Έτσι με αυτόν τον τρόπο, έχουμε δηλώσει ένα ολόκληρο τμήμα βιβλιοθηκών που συμπεριλαμβάνει όλες τις βιβλιοθήκες που επιθυμούμε να χρησιμοποιήσουμε.

Ξεκινώντας τη main, βασική κλάση μας λοιπόν, δηλώνουμε τις βιβλιοθήκες, με τον τρόπο που αναφέρουμε παρακάτω, ακριβώς κάτω από τη γραμμή που καθορίζει το πακέτο, στο οποίο η κλάση αυτή, ανήκει. :

```
package ptyxiakh;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import ptyxiakh.infoPanels.*;
```

Στο παραπάνω τμήμα κώδικα λοιπόν, έχουμε τη δήλωση των βιβλιοθηκών της main κλάσης μας. Κατα σειρά οι βιβλιοθήκες έχουν ως εξής: Στην αρχή είναι οι βιβλιοθήκες που ανήκουν στη, default προεγκατηστημένη, κλάση awt, της java.

- Η βιβλιοθήκη BorderLayout δίνει τη δυνατότητα στην κλάση μας, στο γραφιστικό της κομμάτι, να μπορεί να τοποθετήσει τα αντικείμενά της δυναμικά μέσα στο χώρο που τα χρησιμοποιεί.
- Η βιβλιοθήκη Color, παρέχει τη δυνατότητα χρησιμοποίησης χρώματος πάνω στα αντικείμενα της κλάσης.
- Η βιβλιοθήκη Image, μας επιτρέπει να χρησιμοποιήσουμε εικόνες με όποιον τρόπο θέλουμε στα αντικείμενα που χρησιμοποιούμε.
- Η ActionEvent και η ActionListener δημιουργούν "γεγονότα", δηλαδή στιγμιότυπα αλληλεπιδράσεων μεταξύ των αντικειμένων της κλάσης, που ο χρήστης επιθυμεί να χρησιμοποιήσει.
- Στη συνέχεια είναι οι βιβλιοθήκες που ανήκουν στη , default προεγκατηστημένη, κλάση io, της java.

- Η βιβλιοθήκη IOException παρέχει ιδιότητες που το πρόγραμμα χρησιμοποιεί για την πρόληψη τυχών εξαιρεσέων που μπορεί να προκύψουν, στην εκτέλεση του προγράμματος.
- Στη συνέχεια είναι οι βιβλιοθήκες που ανήκουν στη , default προεγκατηστημένη, κλάση swing, της java.
- Η βιβλιοθήκη ImageIcon μας επιτρέπει να χρησιμοποιήσουμε ως εικόνες διάφορα αντικείμενα.
- Η JButton παρέχει τη δημιουργία αντικειμένων τυπου button, δηλαδή κουμπί.
- Η JFrame, όπου είναι και η βασική βιβλιοθήκη που χρησιμοποιεί το πρόγραμμα μας, επιτρέπει στο πρόγραμμα τη δημιουργία ενός frame, δηλαδή παραθύρου, όπου μέσα σε αυτό μπορούμε να δημιουργήσουμε τα αντικείμενά μας.
- Και τέλος η infoPanels, είναι μία κλάση όπου εμείς δημιουργήσαμε και όπου η main χρησιμοποιεί όλα της αντικείμενα.

Ξεκινώντας τη main κλάση, δίνουμε το όνομα της, MainFrame στην προκειμένη και τον τύπο που αντιπροσωπεύει, εδώ JFrame. Δηλαδή η ίδια η κλάση αντιπροσωπεύει ένα frame (παράθυρο). Στη συνέχεια, πριν αρχίσουμε να χτίζουμε το σώμα της κλάσης, δηλώνουμε τις μεταβλητές μας (declaration). Αυτό βλέπουμε στο παρακάτω τμήμα κώδικα:

```
public class MainFrame extends JFrame {

   private HomePage home = new HomePage();
   private Categories cat = new Categories();
   private Kathgories kat = new Kathgories();
   private JButton jButton1, jButton2;
   private Family family = new Family();
   private Sports sports = new Sports();
   private History history = new History();
   private Beaches beaches = new Beaches();
   private FoodCafe food = new FoodCafe();
   private Oikogeneia oikogeneia = new Oikogeneia();
   private Aquarium aqua = new Aquarium();
   private Athlitismos athl = new Athlitismos();
   private Istoria istoria = new Istoria();
   private Paralies paralies = new Paralies();
   private FaghtoPota faghto = new FaghtoPota();
   private events event = new events();
   private Ekdilwseis ekdilwseis = new Ekdilwseis();
   private WaterPark water = new WaterPark();
   private LunaEn lunaEn = new LunaEn();
   private DinosauriaEn dinoEn = new DinosauriaEn();
   private WaterSports waterSports = new WaterSports();
   private TennisEn tennisEn = new TennisEn();
   private divingEn divingEn = new divingEn();
   private GolfEn golfEn = new GolfEn();
   private KnossosEn knossosEn = new KnossosEn();
```

Στο παραπάνω τμήμα κώδικα λοιπόν, δηλώνουμε αλλά και αρχικοποιούμε παράλληλα τις μεταβλητές που θα χρησιμοποιήσει η main κλάση. Οι συγκεκριμένες μεταβλητές, είναι κλάσεις (αντικείμενα) που έχουμε δημιουργήσει και τις δηλώνουμε στη main κλάση για να τις τοποθετήσουμε μέσα στο βασικό μας frame.

Ο τρόπος με τον οποίο δηλώνουμε τη main Κλάση αλλα και τι τύπος είναι (JFrame), γίνεται με την πρώτη γραμμή: Δίνουμε το όνομα της κλασης, public class MainFrame και λέμε ότι αποτελεί επέκταση της JFrame κλάσης, που υπάρχει ήδη προεγκατεστημένη στη java. Άρα με την εντολή "extends JFrame" δηλώνουμε την κλάση μας ως ένα frame (πράθυρο).

Στη συνέχεια του προγράμματος, δηλώνουμε ότι ξεκινάει το χτίσιμο του "σώματος" της κλάσης, δηλαδή δηλώνουμε το " constructor " της κλάσης.

Ο constructor μίας κλάσης λοιπόν, αποτελεί το βασικό κορμό της και μέσα σε αυτόν γίνονται όλες οι λειτουργίες, αλλαγές αλλά και ότι άλλο μπορεί να αποτελέσει κομμάτι που να καθορίζει την κλάση.

Στο παρακάτω τμήμα κώδικα δημιουργούμε το constructor της " MainFrame " και ξεκινάμε με το να δίνουμε τα "setters" της, δηλαδή συναρτήσεις που καθορίζουν διάφορες ιδιότητες της.

```
public MainFrame() throws IOException {
    // set this
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setExtendedState(JFrame.MAXIMIZED_BOTH);
```

Στη πρώτη γραμμή που δηλώνουμε το constructor, φροντίζουμε να αποφύγουμε τυχών προβλήματα που μπορούν να προκύψουν απο εξαιρέσεις που δεν προβλέψαμε. Αυτό λοιπόν το φροντίζει η εντολή " throws IOException". Στη συνέχεια, ξεκινώντας με τις πρώτες ιδιότητες του σώματος της κλάσης, δηλώνουμε τα setters.

- Στο πρώτο setter, (setDefaultCloseOperation), λέμε στο πρόγραμμα ότι κάθε φορα που θα κλείνει ή θα σταματήσει να εκτελείται με οποιονδήποτε τρόπο ή για οποιονδήποτε λόγο, τότε θα κλείνει αυτόματα μόνο του. Ο λόγος που το κάνουμε αυτό είναι για να αποφευχθούν προβλήματα όπως, να εκτελείται το πρόγραμμα ενώ ο χρήστης το έχει κλείσει με κάποιο παράδοξο τρόπο.
- Στο δεύτερο setter (setExtendedState), δίνουμε την ιδιότητα στο παράθυρό μας κάθε φορά που θα ανοίγει να γεμίζει την οθόνη (maximized). Ο λόγος που γίνεται αυτό, είναι προκειμένου η μορφολογία του προγράμματος να είναι δυναμικά ανεπτυγμένη, δηλαδή να μπορεί να ανοίγει σωστά σε οποιαδήποτε οθόνη.

Στη συνέχεια, δηλώνουμε και αρχικοποιούμε μεταβλητές που θα χρησιμοποιεί η MainFrame, πιο συγκεκριμένα είναι αντικείμενα που τοποθετούνται μέσα στο παράθυρο και ορίζουμε σε αυτά κάποιες από τις ιδιότητές τους (με τα setters), που εμείς θα χρησιμοποιήσουμε.

```
  //init comp
    Image image1 = new
ImageIcon(this.getClass().getResource("/images/categories/home.jpg")).getImage();
    Image image2 = new
ImageIcon(this.getClass().getResource("/images/categories/exit.jpg")).getImage();
    jButton1 = new JButton(new ImageIcon(image1));
    jButton2 = new JButton(new ImageIcon(image2));
    jButton1.setBackground(new Color(165, 228, 237));
    jButton2.setBackground(new Color(165, 228, 237));
```

Αναλυτικά για την κάθε μεταβλητή, του παραπάνω τμήματος κώδικα:

- Χρησιμοποιούμε τις μεταβλητές τύπου ImageIcon, προκειμένου να δημιουργήσουμε ένα αντικείμενο το οποίο ουσιαστικά αποτελεί μία εικόνα.
  Ο τρόπος, με τον οποίο "παίρνουμε" την εικόνα που θέλουμε είναι με την εντολή ".getResource", μέσα στην οποία δίνουμε το μονοπάτι όπου βρίσκεται η εικόνα και στη συνέχεια με την εντολή ".getImage()" , τοποθετούμε την εικόνα αυτή μέσα στο αντικείμενο.
- Με τις μεταβλητές τύπου JButton, δημιουργούμε αντικείμενα Buttons, δηλαδή κουμπιά, τα οποία και τοποθετούμε μέσα στο παράθυρο
  Στο πρώτο κουμπί δίνουμε ως "εξώφυλλο" την εικόνα που δημιουργήσαμε παραπάνω με την ImageIcon, συγκεκριμένα την "image1"

15 Κουμπι Home.

και στο δεύτερο κουμπί την " image2"

16 Κουμπί Exit.

Και τέλος δίνουμε χρώμα στο κουμπί. Ο λόγος που το κάνουμε αυτό, είναι για να καλύψουμε τυχών κομμάτια που έμειναν στο εξώφυλλο του κουμπιού που δεν κάλυψε η εικόνα. Έτσι λοιπόν δίνουμε το ίδιο χρώμα με αυτό της εικόνα. Αυτό το επιτυγχάνουμε δίνοντας τον αντίστοιχο κωδικό rgb που επιθυμούμε να χρησιμοποιήσουμε, μέσα από τη συνάρτηση "new Color".

Στη συνέχεια του προγράμματος, τοποθετούμε μέσα στο παράθυρο, δηλαδή στο αντικείμενο της MainFrame, αντικείμενα που έχουμε δημιουργήσει σε άλλη κλάση, η οποία αποτελεί JPanel. Ουσιαστικά παίρνουμε "κομμάτια" που έχουμε δημιουργήσει αλλού και τα τοποθετούμε μέσα στο παράθυρο για να δημιουργήσουμε, την αρχική μας σελίδα / εικόνα.

```
add(home.getjPanel1(), BorderLayout.NORTH);
    add(home.getjPanel2(), BorderLayout.CENTER);
        add(home.getjPanel3(), BorderLayout.SOUTH);
```

Αναλυτικότερα για παραπάνω τμήμα κώδικα:

Χρησιμοποιούμε τη συνάρτηση add() της java, η οποία αυτόματα, τοποθετεί αυτό που της δίνουμε μέσα στην κλάση που βρισκόμαστε, στο σημείο που της δίνουμε.

Δηλαδή εδώ κάνουμε add τρία αντικείμενα τύπου panels από την κλάση " homePage ", τα οποία καλούμε με την εντολή " home.getAntikeimeno", όπου home ένα αντικείμενο τύπου homePage.

Η εντολή αυτή δηλαδή καλεί το αντικείμενο από την κλάση homePage.Το τοποθετούμε μέσα στο παράθυρο, με την εντολή "BorderLayout" όπου καθορίζει το σήμειο που θα τοποθετηθεί. Στο πρώτο αντικείμενο δηλαδή έχουμε BorderLayout.North. Αυτό σημαίνει ότι θα πάρει το αντικείμενο και θα το τοποθετήσει στο πάνω (βόρεια πλευρά) τμήμα του παραθύρου.

Τα αντικείμενα που καλούμε εδώ, είναι ουσιαστικά η αρχική σελίδα του προγράμματος μας, τα οποία έχουμε δημιουργήσει και αναπτύξει στην κλάσση home αλλά τα "ενεργοποιούμε" εδώ μέσα στη main. Η αρχική σελίδα λοιπόν είναι η εξής: (**mainFrame1.1**)

17 Αρχικό μενού της εφαρμογής.

Εξετάζοντας αναλυτικότερα την κλάση "homePage".

Βλέπουμε κι εδώ (όπως και παραπάνω), ότι για να ξεκινήσει η δομή της κλάσης, δηλώνουμε πρώτα το πακέτο στο οποίο ανήκει η κλάση (package ptyxiakh). Στη συνέχεια δηλώνουμε τις βιβλιοθήκες , τις οποίες χρησιμοποιεί για την ανάπτυξη και την υλοποίηση των δεδομένων της. Οι βιβλιοθήκες αυτές είναι οι εξής:

```
import java.awt.Color;
import java.awt.Image;
import java.io.IOException;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
```

Η homepage, είναι ουσιαστικά η κλάση στην οποία αποθηκεύουμε τα δεδομένα και την πληροφορία γενικά, που χρησιμοποιούμε στην αρχική σελίδα. Στην κλάση mainframe ουσιαστικά δίνουμε σύσταση στα δεδομένα αυτά, αφού τα καλέσουμε από την κλάση homepage.

Βλέποντας τον κώδικα της homepage λοιπόν, βλέπουμε τα δεδομένα αυτά. Δηλώνουμε σε πρώτη φάση τις μεταβλητές των τύπων που θα χρησιμοποιήσουμε και στη συνέχεια τα αρχικοποιούμε μέσα στον constructor της κλάσης(η διαδικασία που ακολουθούμε σε κάθε κλάση δηλαδή). Τν αρχικοποίηση και απόδοση τιμών των δεδομένων κάνουμε ως εξής:

```
Image image = new
ImageIcon(this.getClass().getResource("/images/homePage/arxikh_logotupo.jpg")).getImage();
      Image image2 = new
ImageIcon(this.getClass().getResource("/images/homePage/untitled.jpg")).getImage();
      Image image3 = new
ImageIcon(this.getClass().getResource("/images/homePage/untitled2.jpg")).getImage();
      Image image4 = new
ImageIcon(this.getClass().getResource("/images/homePage/untitled3.jpg")).getImage();
      jLabel1 = new JLabel(new ImageIcon(image));
      jLabel2 = new JLabel();
      jLabel3 = new JLabel();
      jPanel1 = new JPanel();
      jPanel2 = new JPanel();
      jPanel3 = new JPanel();
      jButton1 = new JButton(new ImageIcon(image2));
      jButton2 = new JButton(new ImageIcon(image3));
      jButton3 = new JButton(new ImageIcon(image4));
      cat = new Categories();
```

Αναλυτικότερα τα παραπάνω δεδομένα, έχουν ως εξής:

- Για τα αντικείμενα τύπου Image χρησιμοποιούμε τη συνάρτηση ImageIcon προκειμένου να δώσουμε στο αντικείμενο ως τιμή μία εικόνα. Αυτές που εμείς χρησιμοποιούμε λοιπόν είναι οι εξής:

- Στο αντικείμενο jLabel1 περνάμε την πρώτη εικόνα, δηλαδή το εξώφυλλο της εφαρμογής, προκειμένου να το περάσουμε μετά μέσα στο Panel
- Στα αντικείμενα jButton1, jButton2 και jButton3 δίνουμε για τιμές τις παραπάνω τρεις εικόνες. Παίρνουν ως background δηλαδή τις εικόνες αυτές.

Στη συνέχεια τοποθετούμε τα παραπάνω αντικείμενα μέσα στα panels που θέλουμε να χρησιμοποιήσουμε μέσα στο mainFrame , κάτω, πάνω και στο κέντρο, ως εξής:

```
jPanel1.add(jButton1);
      jPanel1.add(jButton2);
      jPanel2.add(jLabel1);
      jPanel3.add(jButton3);
```

Αφού τα έχουν τοποθετηθεί τα αντικείμνεα στα panels που θέλουμε, απενεργοποιούμε τα "όρια" των αντικειμένων τύπου buttons ως εξής:

```
jButton1.setBorderPainted(false);
        jButton2.setBorderPainted(false);
        jButton3.setBorderPainted(false);
```

προκειμένου να να μην υπάρχει διαφορά μεταξύ του background του Panel και του Button.

Στη συνέχεια κλείνουμε τον constructor της κλάσης και δημιουργούμε τις συναρτήσεις, γνωστές ως Getters, οι οποίες παίρνουν ως γνώρισμα ένα από τα αντικείμενα της κλάσης και ουσιαστικά το αντιπροσωπεύουν. Ο λόγος που υπάρχουν οι συναρτήσεις αυτές, είναι για να τις χρησιμοποιήσουμε σε μία άλλη κλάση (όπως κάνουμε εδώ στην κλάση MainFrame) για να καλέσουμε τα αντικείμενα που θέλουμε να χρησιμοποιήσουμε από την κλάση αυτή. Ο τρόπος που υλοποιούνται οι συναρτήσεις αυτές είναι ο εξής:

```
public JPanel getjPanel1() {
    return jPanel1;
}

public JPanel getjPanel2() {
    return jPanel2;
}

public JPanel getjPanel3() {
    return jPanel3;
}

public JButton getjButton3() {
    return jButton3;
}

public JButton getjButton1() {
    return jButton1;
}

public JButton getjButton2() {
    return jButton2;
}
```

Μία συνάρτηση getter λειτουργεί ως εξής: Δηλώνεται εκτός του constructor της κλάσης και η κάθε μία αντιπροσωπεύει ένα αντικείμενο της κλάσης αυτής. Ξεκινάμε δηλώνοντας τη συνάρτησης ως “public” , προκειμένου να τη χρησιμοποιήσουμε σε μία άλλη κλάση με σκοπό να καλέσουμε και να χρησιμοποιήσουμε το αντικείμενο, που αυτή αντιπροσωπεύει. Η δήλωση της λοιπόν ξεκινάει με το “public”, στη συνένεια ακολουθεί ο τύπος του αντικειμένου της συνάρτησης – αν π.χ. είναι Jbutton, τότε θα ξεκινάει “public JButton” - , στη συνέχεια ακολουθεί ή λέξη “get” μαζί με το όνομα του αντικειμένου (getjButton1). Τέλος η συνάρτηση, παίρνει μέσα σαν τιμή επιστροφής το ίδιο το αντικείμενο που αντιπροσωπεύει.

Παράδειγμα: Έστω ότι θέλουμε να δημιουργήσουμε μια συνάρτηση getter για ένα αντικείμενο τύπου JButton με όνομα jButton1 . Τότε η getter θα έχει ως εξής: pubic JButton getjButton1 () { return jButton1;}

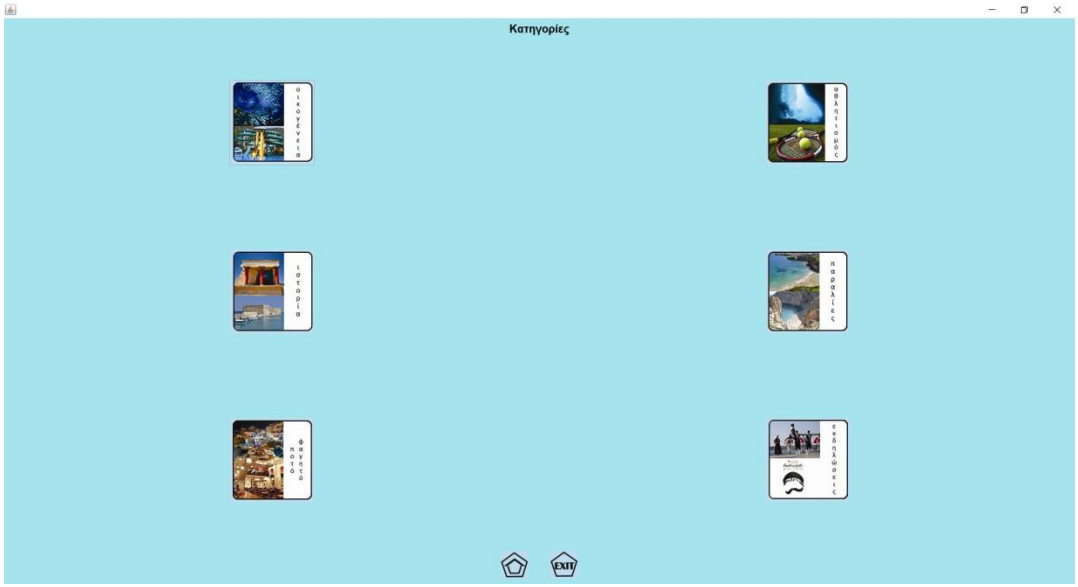
Συνεχίζοντας στη κλάση mainFrame (**mainFrame1.1**), αφού λοιπόν έχουμε τοποθετήσει στο παράθυρο, τα αντικείμενα της κλάσης homePage, ξεκινάμε να τους δίνουμε λειτουργία. Το αντικείμενο jButton3 αντιπροσωπεύει το “exit” του προγράμματος. Οπότε πατώντας το, θέλουμε να κλείνουμε το παράθυρο αλλά και το πρόγραμμα. Ο τρόπος με τον οποίο γίνεται αυτό, είναι ο εξής:

```
home.getjButton3().addActionListener(new ActionListener() {
        @Override
```

```
        public void actionPerformed(ActionEvent e) {
            dispose();
        }
    });
```

Στο παραπάνω τμήμα κώδικα,επιλέγουμε το αντικείμενο jButton3 της κλάσης home που είναι τύπου homePage (home.getjButton3) και της δίνουμε ένα actionListener. Η actionListener είναι μία συνάρτηση της java η οποία αναλαμβάνει να εκτελέσει ένα γεγονός (event) πάνω στο αντικείμενο όπου τοποθετείται. Στην προκειμένη περίπτωση αναλαμβάνει να εκτελέσει το γεγονός που θα υλοποιείται, κάθε φορά που ενεργοποιείται το αντικείμενο home.getjButton3, ή αλλιώς, κάθε φορά που επιλέγεται το exit button. Ο τρόπος με τον οποίο το επιτυγχάνουμε αυτό είναι με το να δώσουμε μέσα στην actionListener, ως γεγονός, τη συνάρτηση της Java, "**dispose()**". Η συνάρητηση dispose όταν καλείται, κλείνει και σταματάει τη λειτουργία της κλάσης στην οποία βρίσκεται. Έτσι εδώ βρισκόμαστε στο βασικό παράθυρο του προγράμματος, όποτε με το να καλούμε τη dispose, σταματάμε τη λειτουργία του παραθύρου αλλά και του όλου προγράμματος, συνεπώς κάνουμε exit.

Στη συνέχεια, δίνουμε μία actionListener στο αντικείμενο home.getjButton2. Το αντικείμενο αυτό, αντιπροσωπεύει το κουμπί για το ελληνικό μενού του πρόγραμματος. Οπότε πατώντας το, κλείνουμε το προηγούμενο παράθυρο (αυτό της homePage), δηλαδή κλείνουμε όλα τα αντικείμενα (panels) που είχαμε πριν για να ανοίξουμε τα νέα, δηλαδή αυτά του ελληνικού μενού, τα οποία βρίσκονται στην κλάση "kathgories". Το μενού των κατηγοριών στα ελληνικά είναι το εξής:



18 Μενού κατηγοριών.

Ο κώδικας για το event του home.getjButton2 είναι ο εξής:

```
home.getjButton2().addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            home.getjPanel1().setVisible(false);
            home.getjPanel2().setVisible(false);
            home.getjPanel3().setVisible(false);
```

```
            kat.getjPanel3().add(jButton1);
            kat.getjPanel3().add(jButton2);
            kat.getjPanel1().setVisible(true);
            kat.getjPanel2().setVisible(true);
            kat.getjPanel3().setVisible(true);
            add(kat.getjPanel1(), BorderLayout.NORTH);
            add(kat.getjPanel2(), BorderLayout.CENTER);
            add(kat.getjPanel3(), BorderLayout.SOUTH);
        }
    });
```

Στο παραπάνω τμήμα κώδικα, όπως προαναφέρθηκε, κλείνουμε τα προηγούμενα Panels και ανοίγουμε τα νέα, αυτά των κατηγοριών. Παράδειγμα στο home.getjPanel1().setVisible(false) κλείνουμε το πρώτο panel του homePage, στο kat.getjPanel1().setVisible(true), ανοίγουμε το jpanel1 της κλασης "kathgories" και με το add(kat.getjPanel1(), BorderLayout.North) τοποθετούμε στο βασικό παράθυρο του προγράμματος (frame) το jPanel1 των κατηγοριών, στο πάνω τμήμα του. (**mainFrame1.2**)

Εξετάζοντας αναλυτικότερα την κλάση "**Kathgories**". Όπως και στις υπόλοιπες κλάσεις δίνουμε τις βιβλιοθήκες, τις μεταβλητές που χρησιμοποιούμε και στη συνέχεια τον constructor. Εδώ χρησιμοποιούμε ως μεταβλητές αντικείμενα τις εικόνες που θα χρησιμοποιήσουμε, τα buttons που θα βάλουμε τις εικόνες αυτές, το αρχικό label και τα panels που όλα αυτά τοποθετούνται. Για τον κώδικα των εικονών που χρησιμοποιεί η κλάση "**Kathgories**" έχουμε:

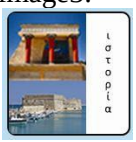
```
 Image image = new ImageIcon(this.getClass().getResource("/images/kathgories/s1g.jpg")).getImage();
      Image image2 = new
ImageIcon(this.getClass().getResource("/images/kathgories/s2g.jpg")).getImage();
      Image image3 = new
ImageIcon(this.getClass().getResource("/images/kathgories/s3g.jpg")).getImage();
      Image image4 = new
ImageIcon(this.getClass().getResource("/images/kathgories/s4g.jpg")).getImage();
      Image image5 = new
ImageIcon(this.getClass().getResource("/images/kathgories/s5g.jpg")).getImage();
      Image image6 = new
ImageIcon(this.getClass().getResource("/images/kathgories/s6g.jpg")).getImage();
```

Για το κάθε image κατά σειρά έχουμε:

- image:

- image2:

- image3:



- image4:

- image5:

- image6:

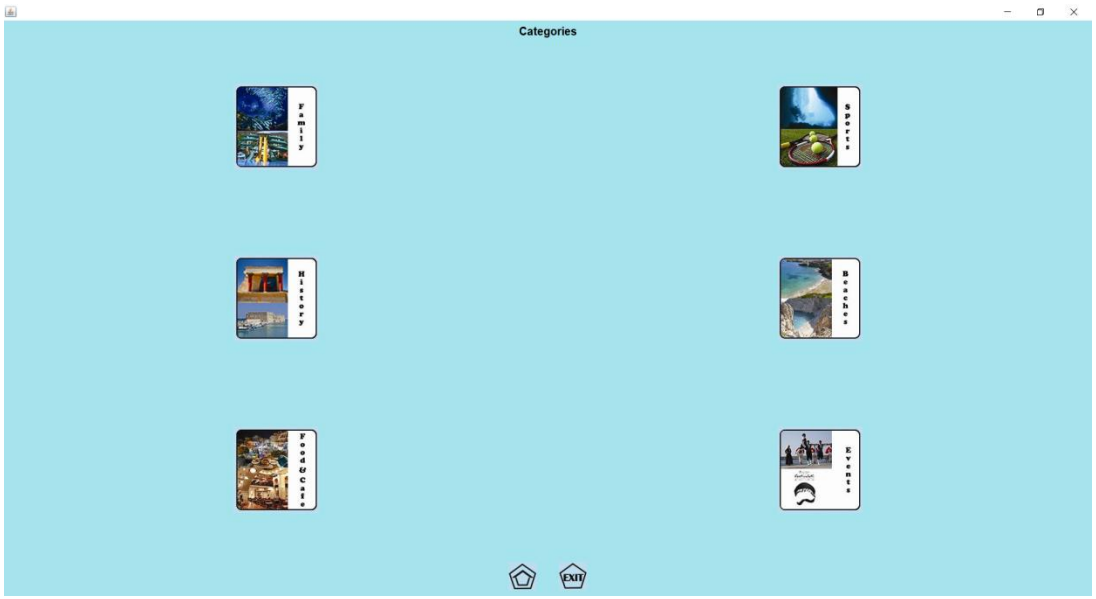
Στη συνέχεια, αρχικοποιούμε τα panels της κλάσης και τα υπόλοιπα αντικείμενα της. Τα τοποθετούμε όλα μέσα στα panels. Στο τέλος του constructor ορίζουμε τη μορφολογία του μεσαίου panel, δηλαδή του κεντρικού, του jPanel2 με τον εξής τρόπο:

```
GridLayout layout  = new GridLayout(3, 2);
      jPanel2.setLayout(layout);
```

Με τη συνάρτηση gridLayout δίνουμε στο Panel τη μορφολογία των 3 γραμμών και 2 στηλών. Ο λόγος που το κάνουμε αυτό, είναι για λόγους καλαισθησίας.

Τέλος, αφού έχει κλείσει ο constructor, τοποθετούμε και ορίζομουμε τις συναρτήσεις getter των αντικειμένων της κλάσης "**Kathgories**", για να τα χρησιμοποιήσουμε στη mainFrame.

Συνεχίζοντας στην κλάση mainFrame λοιπόν (**mainFrame1.2**), εκτός από το home.getjButton2 έχουμε και το home.getjButton1 που αντιπροσωπεύει το αγγλικό μενού των κατηγοριών. Στο home.getjButton1 κάνουμε ακριβώς ότι και στο home.getjButton12, δηλαδή τοποθετούμε μία actionListener, κλείνουμε τα προηγούμενα panels (setVisible(false)) και ανοίγουμε τα καινούρια, δηλαδή αυτά του αγγλικού μενού. Το αγγλικό μενού της εφαρμογής είναι το εξής:



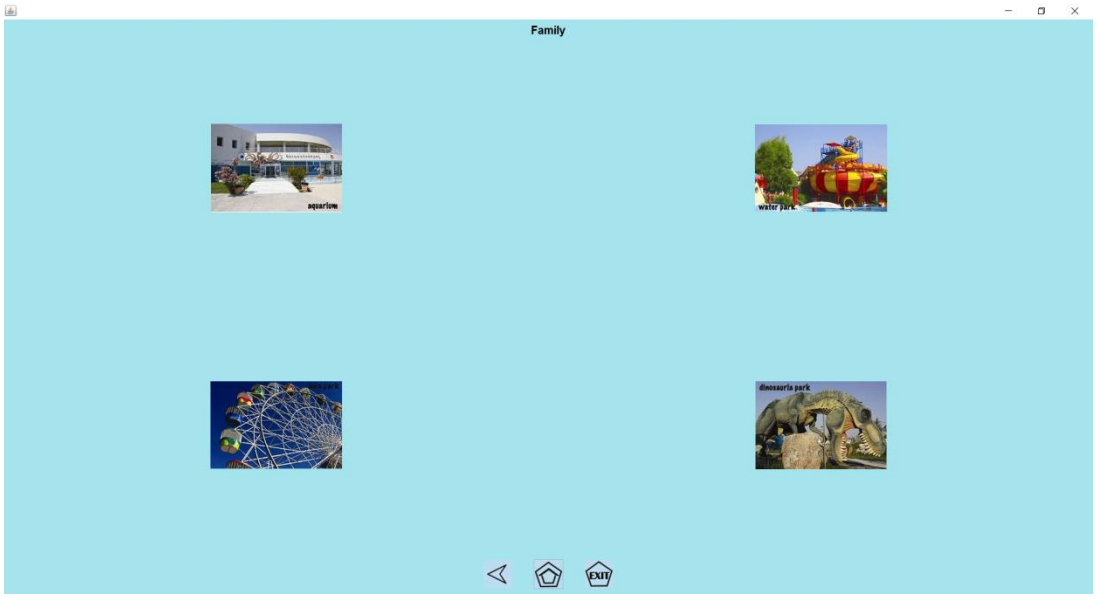
19 Μενου κατηγοριών (Αγγλικα).

Η κλάση "**Categories**" λειτουργεί ακριβώς όπως και η κλάση "**Kathgories**" με τη μόνη διαφορά ότι χρησιμοποιεί διαφορετικά Images, τα οποία είναι τα ίδια αλλά στ' Άγγλικά.

Συνεχίζοντας στη mainframe (**mainFrame1.2**), έχουμε πλέον μπει στο μενού των κατηγοριών (έστω ότι είμαστε στο αγγλικό), οπότε ξεκινάμε και βάζουμε Listeners στα κουμπιά του μενού αυτού, προκειμένου να μας παραπέμψει στην κατηγορία που επιθυμούμε να επισκεφθούμε. Έστω λοιπόν ότι θέλουμε να επισκεφθούμε το τμήμα "**family**" των αγγλικών κατηγοριών. Το αντικείμενο κουμπί της κλάσης "**Categories**" που αντιπροσωπεύει το τμήμα "**family**", είναι το cat.getjButton1. Στο listener του button αυτού λοιπόν, λειτουργούμε όπως και πριν, δηλαδή κλείνουμε όλα τα προηγούμενα panels Και ανοίγουμε τα νέα, δηλαδή το συνολικό παράθυρο της κατηγορίας "**family**". Ο listener είναι ο εξής: (**mainFrame1.3**)

```
cat.getjButton1().addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
           cat.getjPanel1().setVisible(false);
           cat.getjPanel2().setVisible(false);
           cat.getjPanel3().setVisible(false);
           home.getjPanel1().setVisible(false);
           home.getjPanel2().setVisible(false);
           home.getjPanel3().setVisible(false);
           family.getjPanel3().add(jButton1);
           family.getjPanel3().add(jButton2);
           family.getjPanel1().setVisible(true);
           family.getjPanel2().setVisible(true);
           family.getjPanel3().setVisible(true);
           add(family.getjPanel1(), BorderLayout.NORTH);
           add(family.getjPanel2(), BorderLayout.CENTER);
           add(family.getjPanel3(), BorderLayout.SOUTH);
        }
    });
```

Το τμήμα "**family**" στην εφαρμογή είναι το εξής:



20 Μενού επιλογών που αφορούν κυρίως οικογένειες.

Εξετάζοντας την κλάση "**family**" αναλυτικότερα. Είναι μία κλάση τύπου "**CategoriesSuper**". Αυτό το βλέπουμε από τη δήλωση της.

```
public class Family extends CategoriesSuper{
```

Δηλαδή εδώ βλέπουμε ότι η κλάση **family** κληρονομεί την **CategoriesSuper**. Η **CategoriesSuper** όμως είναι μία κλάση που δημιουργήσαμε εμείς, με σκοπό τη δημιουργία και την αποθήκευση των δεδομένων και μεταβλητών που χρησιμοποιούν όλες οι κλάσεις κατηγοριών (αγγλικές και ελληνικές). Έτσι με αυτό τον τρόπο δεν θα χρειάζεται να τα δημιουργούμε σε κάθε κλάση κατηγορίας όλα από την αρχή, θα τα χρησιμοποιούμε απ ευθείας από την **CategoriesSuper** για να τους δώσουμε την ανάλογη τιμή από την κλάση αυτή.

Βλέποντας αναλυτικότερα, την κλάση "**CategoriesSuper**". Όπως και στις υπόλοιπες κλάσεις, δίνουμε τις βιβλιοθήκες και τις μεταβλητές, στην αρχή, που θα χρησιμοποιήσει η κλάση. Στη συνέχεια ανοίγουμε τον constructor της κλάσης και αρχίζουμε να τη χτίζουμε. Έχουμε μία μεταβλητή τύπου ImageIcon, την οποία όχι απλώς δηλώνουμε εδώ, αλλά την αρχικοποιούμε επίσης, γιατί θα την χρησιμοποιούν ίδια όλες οι κλάσεις που θα κληροδοτεί η **CategoriesSuper.** Για την ImageIcon λοιπόν:

```
Image image = new ImageIcon(this.getClass().getResource("/images/categories/back.jpg")).getImage();
```

Η εικόνα που περναμε στην image:

Το αντικείμενο image περνάμε μετά στο αντικείμενο, jButton5 που ουσιαστικά αντιπροσωπεύει το backButton. Όποτε χρησιμοποιείται δηλαδή επιστρέφει στην προηγούμενη κατάσταση.

Επιστρέφοντας στην κλάση family λοιπόν, βλέπουμε ότι, εφόσον κληρονομεί την CategoriesSuper , δεν έχει δικές τις αρχικές τιμές, παρά μόνο τιμές στις οποίες δίνει σύσταση από την κλάση CategoriesSuper. Ο constructor της family είναι ο εξής:

```
   public Family() {
       Image image = new
ImageIcon(this.getClass().getResource("/images/categories/family/aquarium.jpg")).getImage();
       Image image2 = new ImageIcon(this.getClass().getResource("/images/categories/family/water
park.jpg")).getImage();
       Image image3 = new
ImageIcon(this.getClass().getResource("/images/categories/family/luna.jpg")).getImage();
       Image image4 = new
ImageIcon(this.getClass().getResource("/images/categories/family/dinosauria.jpg")).getImage();

       jLabel1.setText("Family");
       jButton1.setIcon(new ImageIcon(image));
       jButton2.setIcon(new ImageIcon(image2));
       jButton3.setIcon(new ImageIcon(image3));
       jButton4.setIcon(new ImageIcon(image4));

   }
```

Οι εικόνες που χρησιμοποιούμε για το παραπάνω τμήμα κώδικα, στο κάθε Button είναι οι εξής:

- jButton1:
  

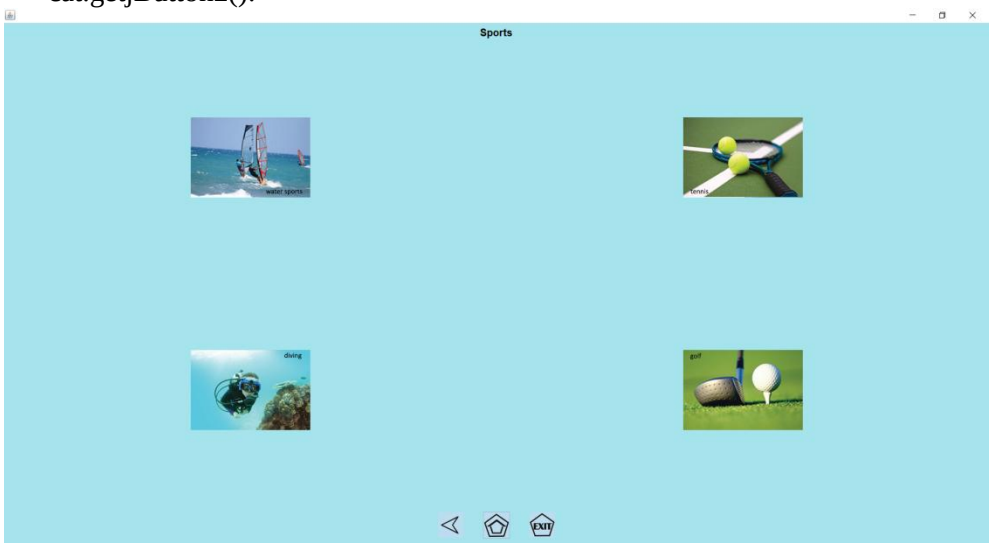
- jButton2:
  

- jButton3:
  

- jButton4:

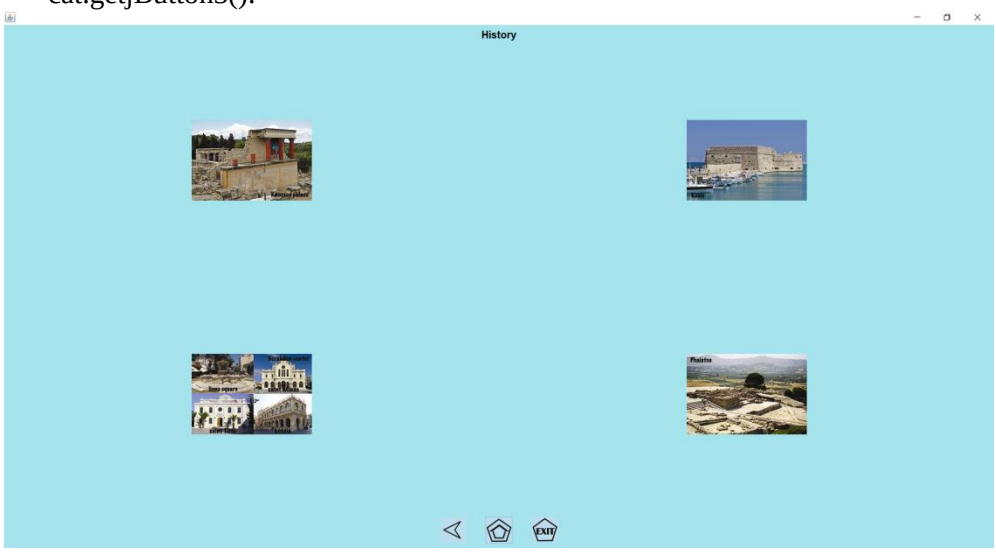
Συνεχίζοντας στη mainframe (**mainFrame1.3**), έχουμε actionListeners για το κάθε button των κατηγοριών. Όλοι οι listener δουλεύουν ακριβώς το ίδιο (όπως και αυτό του Family). Οπότε για τα παράθυρα της κάθε κατηγορίας έχουμε:

cat.getjButton2():



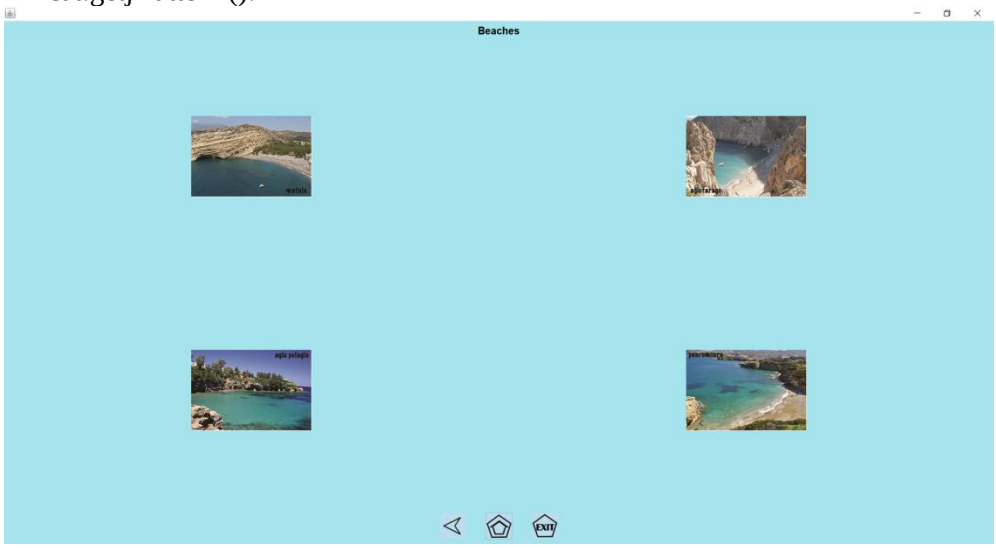
21 Μενού επιλογών για αθλήματα.

cat.getjButton3():



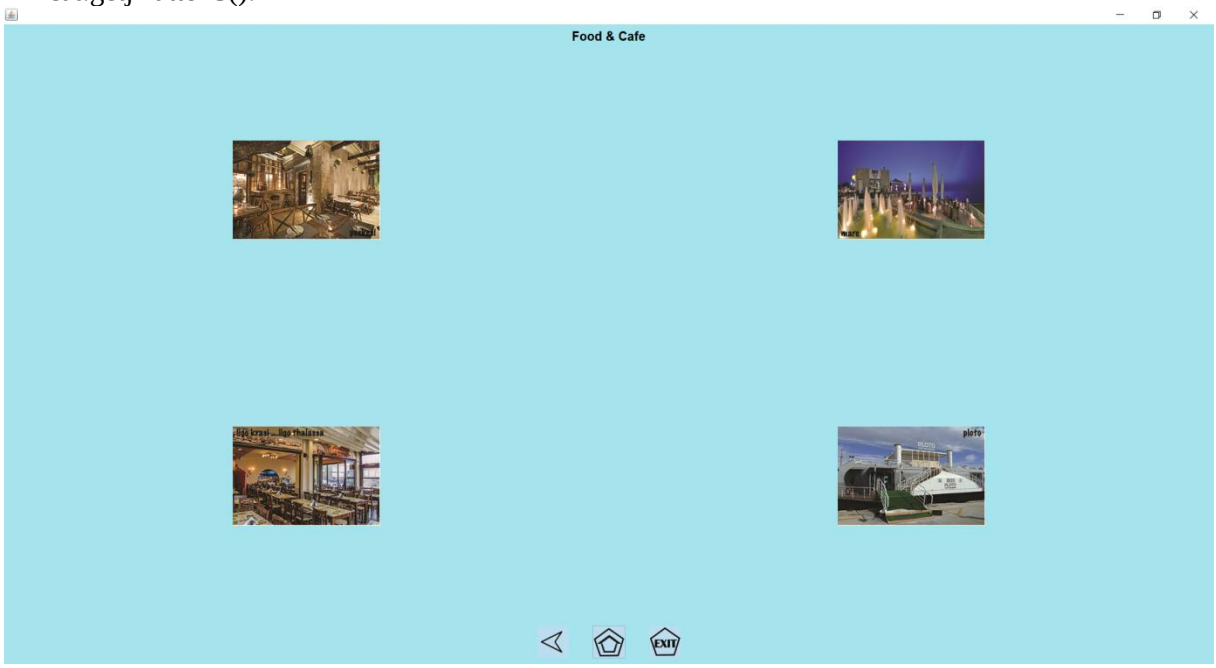
22 Μενού επιλογών ιστορικών τοποθεσιών.

cat.getjButton4():



23 Μενού επιλογών παραλιών.

cat.getjButton5():



24 Μενού επιλογών χώρων εστίασης.

Για τις κατηγορίες στα ελληνικά ισχύει ακριβώς το ίδιο, με τη διαφορά ότι οι εικόνες είναι στα ελληνικά. Επίσης όλες οι κλάσεις των κατηγοριών(αγγλικά και ελληνικά) λειτουργούν ακριβώς το ίδιο με το παράδειγμα της κλάσης "**family**" παραπάνω. Κληρονομούν την **CategoriesSuper** και υλοποιούν τις ίδιες μεταβλητές.

Συνεχίζοντας στη mainframe λοιπόν, θεωρούμε ότι έχουμε μπει στο παράθυρο κάποιας κατηγορίας(έστω της history). Υλοποιούμε με ActionListener τη λειτουργικότητα του **jButton5**, που όπως είδαμε παραπάνω, αντιπροσωπεύει το back button. Δηλαδή εδώ το πρόγραμμα, κάθε φορά και σε κάθε κατηγορία, όταν επιλέγεται και ενεργοποιείται, θα κλείνει τα panel στα οποία βρίσκεται και θα

ανοίγει αυτά των αντίστοιχων κατηγοριών (ελληνικά ή αγγλικά), που βρισκόταν πριν. Π.χ. εδώ είμαστε στην κατηγορία **history**, όταν πατάμε το back button (**jButton5**) κλείνουμε τα panels του **history** και ανοίγουμε ξανα αυτά των **Categories**.

Η κάθε κατηγορία, όπως μπορούμε να παρατηρήσουμε, αποτελείται από 4 υποκατηγορίες. Η κάθε υποκατηγορία, αντιπροσωπεύει ένα κουμπί που μας παραπέμπει σε ένα νέο παράθυρο. Το νέο παράθυρο αυτό αποτελεί τις πληροφορίες που ο χρήστης επιθυμεί να δει για την κατηγορία αυτή.

```
// HISTORY
      history.getjButton5().addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            history.getjPanel1().setVisible(false);
            history.getjPanel2().setVisible(false);
            history.getjPanel3().setVisible(false);
            cat.getjPanel3().add(jButton1);
            cat.getjPanel3().add(jButton2);
            cat.getjPanel1().setVisible(true);
            cat.getjPanel2().setVisible(true);
            cat.getjPanel3().setVisible(true);
         }
      });
```

Όλες οι κατηγορίες λειτουργούν με τον ίδιο τρόπο, με τη διαφορά του αν ανήκουν στο ελληνικό μενού, ανοίγουν τα Panels της κλάσης **Kathgories**.

Στον κώδικα της MainFrame λοιπόν, παρατηρούμε ότι για το κάθε button της κάθε κατηγορίας(έστω ότι τώρα βρισκόμαστε πάλι στην κατηγορία history), έχουμε ένα ActionListener, που μας παραπέμπει στο νέο παράθυρο, τις πληροφορίες των υποκατηγοριών δηλαδή. Εδώ βλέπουμε τον κώδικα για το ActionListener του button που μας παραπέμπει στο παράθυρο του "**Knossos Palace**" (**mainFrame1.4**)

```
//    HISTORY
      //    KNOSSOS
      history.getjButton1().addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            history.getjPanel1().setVisible(false);
            history.getjPanel2().setVisible(false);
            history.getjPanel3().setVisible(false);
            knossosEn.getjPanel3().add(jButton1);
            knossosEn.getjPanel3().add(jButton2);
            knossosEn.getjPanel1().setVisible(true);
            knossosEn.getjPanel2().setVisible(true);
            knossosEn.getjPanel3().setVisible(true);
            add(knossosEn.getjPanel1(), BorderLayout.NORTH);
            add(knossosEn.getjPanel2(), BorderLayout.CENTER);
            add(knossosEn.getjPanel3(), BorderLayout.SOUTH);
         }
      });
```

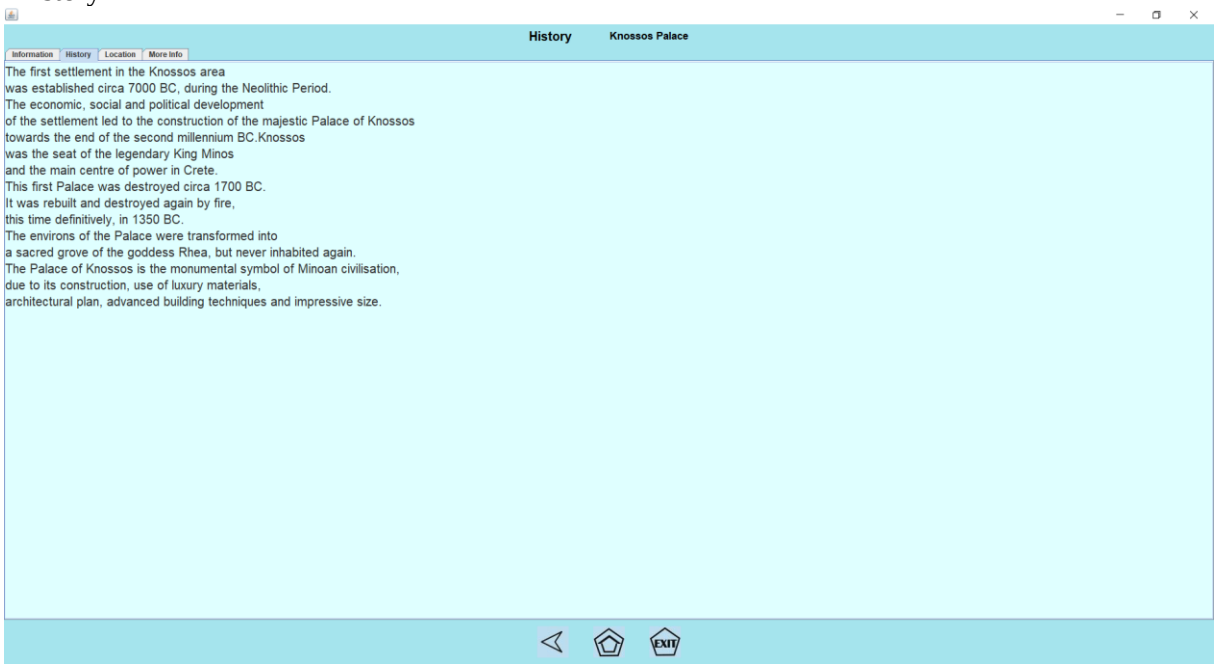
Τα παράθυρα αυτά, αποτελούνται από 4 tabs. Κατά σειρά έχουν ως εξής:

- Ένα για τις πληροφορίες της υποκατηγορίας αυτής
- Ένα για την ιστορία (αν υπάρχει όπως εδώ) της της υποκατηγορίας
- Ένα για την τοποθεσία (εικόνα από χάρτη)
- Και τέλος ένα για περαιτέρω πληροφορίες της υποκατηγορίας (για όποιον ενδιαφέρεται)

Το παράθυρο αυτό λοιπόν (για την υποκατηγορία history) έχει ως εξής, για το κάθε tab, κατά σειρά. "information"

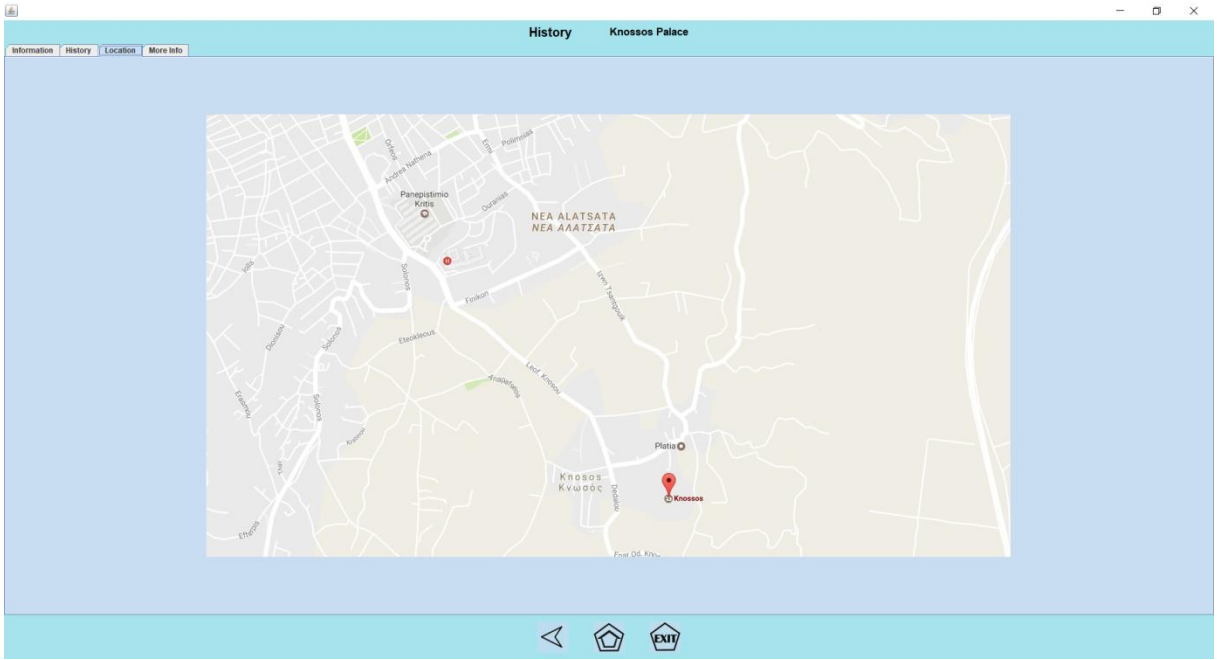
25 Παράθυρο εμφάνισης γενικών πληροφοριών.

"History"



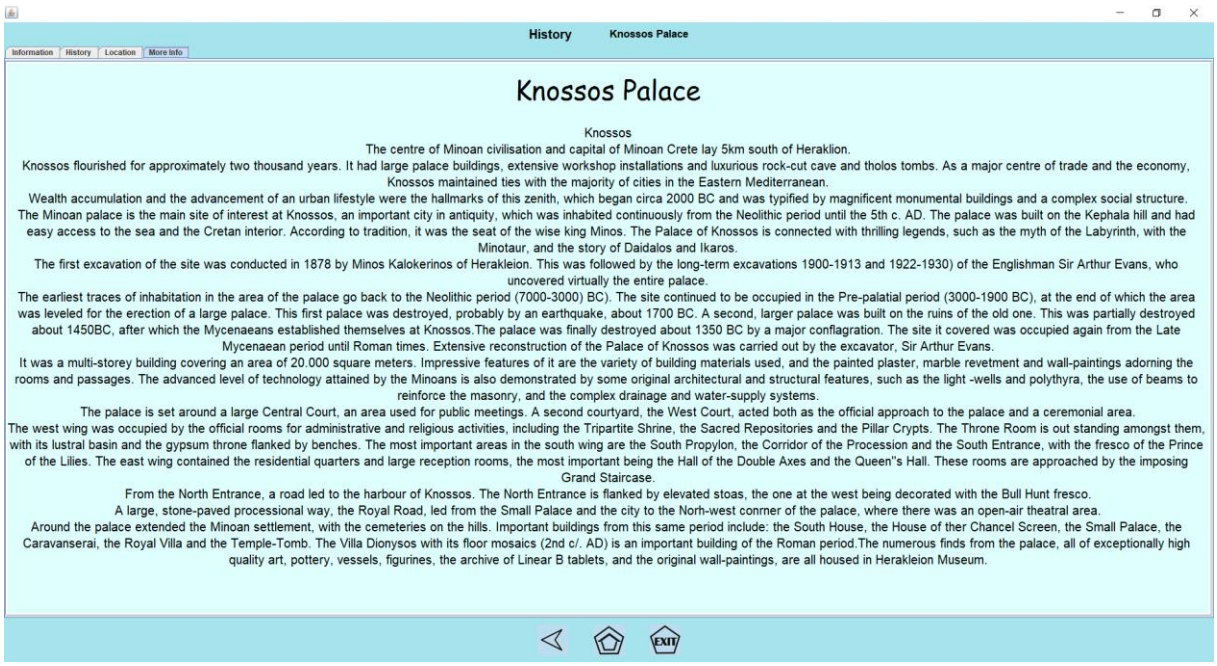
26 Παράθυρο ενημέρωσης σχετικά με την ιστορία.

"Location"



27 Εικόνα με την γενική τοποθεσία.

“More Info”



28 Παράθυρο με αναλυτικότερες πληροφορίες απο HTML.

Βλέποντας αναλυτικότερα την κλάση “**KnossoEn**”, η οποία και αντιπροσωπεύει το παραπάνω παράθυρο, παρατηρούμε ότι κληρονομεί την κλάση “**infoPanelsSuper**”. Είναι δηλαδή μία κλάση τύπου “ **infoPanelsSuper** ”. Αυτό το βλέπουμε από τη δήλωση της.

```
public class Family extends infoPanelsSuper {
```

Δηλαδή εδώ βλέπουμε ότι η κλάση **KnossoEn** κληρονομεί την **infoPanelsSuper**. Η **infoPanelsSuper** όμως είναι μία κλάση που δημιουργήσαμε εμείς, με σκοπό τη δημιουργία και την αποθήκευση των δεδομένων και μεταβλητών που χρησιμοποιούν όλες οι κλάσεις υποκατηγοριών (αγγλικές και ελληνικές). Έτσι με αυτό τον τρόπο δεν θα χρειάζεται να τα δημιουργούμε σε κάθε κλάση υποκατηγορίας όλα από την αρχή, θα τα χρησιμοποιούμε απ ευθείας από την **infoPanelsSuper** για να τους δώσουμε την ανάλογη τιμή από την κλάση αυτή.

Βλέποντας αναλυτικότερα, την κλάση ” **infoPanelsSuper**”. Όπως και στις υπόλοιπες κλάσεις, δίνουμε τις βιβλιοθήκες και τις μεταβλητές, στην αρχή, που θα χρησιμοποιήσει η κλάση. Στη συνέχεια ανοίγουμε τον constructor της κλάσης και αρχίζουμε να τη χτίζουμε.

Πέρα από τις μεταβλητές και τα αντικείμενα της java που χρησιμοποιούμε και στις υπόλοιπες κλάσεις, εδώ χρησιμοποιούμε και 2 συναρτήσεις, για να μεταφράσουμε τμήματα κώδικα από τη γλώσσα προγραμματισμού **html** και τη γλώσσα προγραμματισμού **css.** Ο λόγος που το κάνουμε αυτό είναι για να

“περάσουμε” μέσα στο πρόγραμμα μας άλλα τμήματα γλώσσων προγραμματισμού που θέλουμε να δανειστούμε, για να τα χρησιμοποιήσουμε μέσα στο πρόγραμμα μας.

Για τις συναρτήσεις αυτές λοιπόν έχουμε:

HTML

Έπειτα θα εξηγήσουμε την διαδικασία με την οποία η εφαρμογή μας επεξεργάζεται τα αρχεία HTML και CSS ώστε να τα εμφανίσει στο παράθυρό μας και να τα παρουσιάσει όπως έχει οριστεί.

```
String HTML = "";
 try {
    BufferedReader in = new BufferedReader(new FileReader("aquaEn.html"));
    String str;
    while ((str = in.readLine()) != null) {
       HTML += str;
    }
    in.close();
 } catch (IOException e) {
 }

 Document doc = kit.createDefaultDocument();
 jEditorPane.setDocument(doc);
 jEditorPane.setText(HTML);
 jScroll = new JScrollPane(jEditorPane);
```

Με τον παραπάνω κώδικα η εφαρμογή μας πέρνει ως είσοδο ένα αρχείο της μορφής HTML και το μετατρέπει σε jScrollPane ώστε να μπορέσουμε να το εισάγουμε σε ένα jPanel και τελικά να εμφανιστεί στο τελικό μας jFrame. Αναλυτικότερα, η σειρά με την οποία γίνονται οι διαδικασίες είναι οι εξής:

- Ξεκινάμε ένα BufferedReader, το οποίο δέχεται ως είσοδο, χαρακτήρες τους οποίους στην συνέχεια τους παρέχουμε από το αρχείο που διαβάζουμε μέσω ενός FileReader. Το αρχείο που χρησιμοποιήσαμε σε αυτό το παράδειγμα έχει το όνομα “aquaEn.html”.
- Στο επόμενο βήμα έχουμε ένα βρόγχο While με τον οποίο διαβάζουμε όλες τις σειρές του αρχείου που δηλώσαμε και τις εισάγουμε σε μία μεταβλητή String.
- Έπειτα δημιουργούμε μία μεταβλητή Document χρησιμοποιώντας την μέθοδο createDefaultDocument η οποία επιστρέφει ένα κείμενο στην κατάλληλη μορφή για να χρησιμοποιήσουμε στον HTML editor.
- Δίνουμε στον editor το αρχείο που δημιουργήσαμε προηγουμένως με την μέθοδο setDocument(doc).
- Έπειτα δηλώνεται το κείμενο μέσα σε αυτο το component σύμφωνα με το είδος και την μορφοποίηση που έχει δηλωθεί προηγουμένως(HTML στην περίπτωση μας).
- Τέλος,εισάγουμε το τελικό αποτέλεσμα, με το κείμενο μας σε μορφή HTML, σε ένα jScrollPane το οποίο αργότερα θα εισαχθεί σε ένα jPanel και τελικα το jPanel θα εισαχθεί στο τελικό μας jFrame.

Η διαδικασία που προαναφέραμε επιτρέπει στην εφαρμογή μας να παρουσιάζει αρχεία HTML στον χρήστη, τα οποία θα ήταν βέλτιστο να βρίσκονται σε ένα κεντρικό εξυπηρετητή. Όταν ένας χρήστης επιθυμεί να μάθει περισσότερες πληροφορίες για ένα μέρος ή μία επιχείρηση που τον ενδιαφέρει, θα μπορούσε η εφαρμογή να συνδέεται στο διαδίκτυο (εάν αυτό είναι διαθέσιμο εκείνη την στιγμή) και να «τραβάει» τα αρχεία και να τα παρουσιαζει στον χρήστη όπως αυτά θα φαίνονταν στην ιστοσελίδα σε ένα περιηγητή(Mozilla, Google Chrome, Safari κτλ) χωρίς να χάνεται η διαρύθμηση της ιστοσελίδας.

CSS

Παρακάτω θα εξηγήσουμε την διαδικασία που υλοποιήσαμε ώστε τα αρχεία HTML που εμφανίσαμε στο jPanel, χρησιμοποιώντας τον κώδικα που αναφέραμε στο προηγούμενο βήμα, να υπακούουν σε κανόνες CSS.

```
        try {

      BufferedReader in = new BufferedReader(new FileReader("styling.css"));
      String str;
      while ((str = in.readLine()) != null) {
         styleSheet.addRule(str);
      }
      in.close();
    } catch (IOException e) {
    }
```

Η διαδικασία που χρησιμοποιήσαμε έχει ως εξής:

- Αρχικά ξεκινάμε διαβάζοντας το αρχείο CSS όπου περιέχονται οι κανόνες που θα χρησιμοποιήσουμε.
- Δημιουργούμε μία μεταβλητή τύπου String όπου θα αποθηκεύσουμε τους κανόνες.
- Τέλος προσθέτουμε στον Editor τους κανόνες μέσω της μεθόδου "addRule()" τους οποίους έχουμε διαβάσει με την χρήση ενός βρόγχου While ανά σειρά του αρχείου CSS.

Η **infoPanelsSuper** υλοποιεί περαιτέρω και 4 συναρτήσεις, που η κάθε μία αντιπροσωπεύει τα παράθυρα των υποκατηγοριών αυτών, ανάλογα με τα πόσα tabs έχουν. Για την πρώτη συνάρτηση:

```
  public static JPanel imagePanel1(JTextArea tx) {
    JTabbedPane tabs = new JTabbedPane();
    JTextArea txt = tx;
    txt.setFont(txt.getFont().deriveFont(20f));
    txt.setBackground(new Color(223, 255, 255));
    txt.setEditable(false);
    JPanel home = new JPanel();
    tabs.addTab("Information", txt);
    home.setBackground(new Color(165, 228, 237));
    home.add(tabs);
    return home;
  }
```

Δημιουργούμε το παράθυρο το οποίο έχει ένα tab μόνο, το "information". Δίνουμε τιμές στις μεταβλητές που αντιπροσωπεύουν τη μορφολογία του και τέλος το περνάμε στο panel.

Αντίστοιχα έχουμε την επόμενη συνάρτηση που αντιπροσωπεύει τα παράθυρα τα οποία αποτελούνται από 3 tabs :

```
public static JPanel imagePanel2(JTextArea tx, ImageIcon loc, JScrollPane jsp) {
 JTabbedPane tabs = new JTabbedPane();
 JTextArea txt = tx;
 txt.setFont(txt.getFont().deriveFont(20f));
 txt.setBackground(new Color(223, 255, 255));
 txt.setEditable(false);
 JScrollPane js = jsp;
 JPanel home = new JPanel(new BorderLayout());
 ImageIcon image = loc;
 tabs.addTab("Information", txt);
 tabs.addTab("Location", new JLabel(image));
 tabs.addTab("More Info", js);
 home.setBackground(new Color(165, 228, 237));
 home.add(tabs, BorderLayout.CENTER);
 return home;
}
```

Εδώ δημιουργούμε το παράθυρο το οποίο έχει τρία tabs, το "information", το "location" και το "More info"

Στη συνέχεια έχουμε την επόμενη συνάρτηση που αντιπροσωπεύει τα παράθυρα τα οποία αποτελούνται από 2 tabs :

```
public static JPanel imagePanel21(JTextArea tx, ImageIcon loc) {
 JTabbedPane tabs = new JTabbedPane();
 JTextArea txt = tx;
 txt.setFont(txt.getFont().deriveFont(20f));
 txt.setBackground(new Color(223, 255, 255));
 txt.setEditable(false);
 JPanel home = new JPanel(new BorderLayout());
 ImageIcon image = loc;
 tabs.addTab("Information", txt);
 tabs.addTab("Location", new JLabel(image));
 home.setBackground(new Color(165, 228, 237));
 home.add(tabs, BorderLayout.CENTER);
 return home;
```

Εδώ δημιουργούμε το παράθυρο το οποίο έχει δύο tabs, το "information"και το "location".

Και τέλος έχουμε την επόμενη συνάρτηση που αντιπροσωπεύει τα παράθυρα τα οποία αποτελούνται από 4 tabs :

```
public static JPanel imagePanel3(JTextArea tx, JTextArea tx2, ImageIcon loc, JScrollPane jsp) {
 JTabbedPane tabs = new JTabbedPane();
 JTextArea txt = tx;
```

```
        JTextArea txt2 = tx2;
        JPanel home = new JPanel(new BorderLayout());
        ImageIcon image = loc;
        JScrollPane js = jsp;
        txt.setFont(txt.getFont().deriveFont(20f));
        txt.setBackground(new Color(223, 255, 255));
        txt.setEditable(false);
        txt2.setFont(txt.getFont().deriveFont(20f));
        txt2.setBackground(new Color(223, 255, 255));
        txt2.setEditable(false);
        tabs.addTab("Information", txt);
        tabs.addTab("History", txt2);
        tabs.addTab("Location", new JLabel(image));
        tabs.addTab("More Info", js);
        home.setBackground(new Color(165, 228, 237));
        home.add(tabs, BorderLayout.CENTER);
        return home;
    }
```

Εδώ δημιουργούμε το παράθυρο το οποίο έχει τρία tabs, το “information”,το “History”, το “location” και το “More info”.

Τώρα που είδαμε τη λειτουργικότητα της ” **infoPanelsSuper**”, επιστρέφουμε για να δούμε αυτή της κλάσης “ **KnossoEn** ”.

Η “ **KnossoEn** ” λοιπόν, ξεκινάει και υλοποιεί τις απλές μεταβλητές που κληρονομεί από την **infoPanelsSuper** , όπως να δίνει τιμή στα labels που αντιπροσωπεύουν τον τίτλο του παραθύρου. Στη συνέχεια υλοποιεί τις συναρτήσεις για τον html και css κώδικα που είδαμε πριν. Τέλος τα περνάει όλα μέσα στο τελικό Panel. Η κλάση υλοποιημένη, έχει ως εξής:

```
package ptyxiakh.infoPanels;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import javax.swing.ImageIcon;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.text.Document;

public class KnossosEn extends infoPanelsSuper {

    public KnossosEn() {
        jLabel1.setText("History");
        jLabel2.setText("          Knossos Palace");

         String HTML = "";
        try {
            BufferedReader in = new BufferedReader(new FileReader("knossosEn.html"));
            String str;
```

```
            while ((str = in.readLine()) != null) {
                HTML += str;
            }
            in.close();
        } catch (IOException e) {
        }

        Document doc = kit.createDefaultDocument();
        jEditorPane.setDocument(doc);
        jEditorPane.setText(HTML);
        jScroll = new JScrollPane(jEditorPane);

         //Knossos
        JTextArea KnossosTxt = new JTextArea();
        JTextArea KnossosTxt2 = new JTextArea();
        KnossosTxt.setText("Knossos, the famous Minoan Palace lies 5 kilometres" + "\n" + "southeast of
Heraklion, in the valley of the river Kairatos. " + "\n" + "The river rises in Archanes, runs through
Knossos " + "\n" + "and reaches the sea at Katsabas, the Minoan harbour of Knossos." + "\n" + "In
Minoan times the river flowed all year round and the surrounding " + "\n" + "hills were covered in oak
and cypress trees, " + "\n" + "where today we see vines and olives. " + "\n" + "The pine trees inside the
archaeological site were planted by Evans." + "\n" + "Constant habitation for 9,000 years has brought " +
"\n" + "about great changes to the natural environment, " + "\n" + "so it is hard to imagine what the
Minoan landscape was like." + "\n" + "\n");
        KnossosTxt2.setText("The first settlement in the Knossos area" + "\n" + "was established circa 7000
BC, during the Neolithic Period. " + "\n" + "The economic, social and political development " + "\n" +
"of the settlement led to the construction of the majestic Palace of Knossos " + "\n" + "towards the end of
the second millennium BC.Knossos " + "\n" + "was the seat of the legendary King Minos " + "\n" + "and
the main centre of power in Crete." + "\n" + "This first Palace was destroyed circa 1700 BC. " + "\n" + "It
was rebuilt and destroyed again by fire, " + "\n" + "this time definitively, in 1350 BC. " + "\n" + "The
environs of the Palace were transformed into " + "\n" + "a sacred grove of the goddess Rhea, but never
inhabited again." + "\n" + "The Palace of Knossos is the monumental symbol of Minoan civilisation, " +
"\n" + "due to its construction, use of luxury materials, " + "\n" + "architectural plan, advanced building
techniques and impressive size.");
        JPanel KnossosHome = imagePanel3(KnossosTxt, KnossosTxt2, new ImageIcon("Knossos.jpg"),
jScroll);
        jPanel2 = KnossosHome;
    }

}
```

Όλες οι κλάσεις τύπου **infoPanelsSuper (**είτε αγγλικές, είτε ελληνικές**),** λειτουργούν με ακριβώς τον ίδιο τρόπο, όπως αυτή της **KnossoEn** με τη μόνη διαφορά, να υλοποιούν την ανάλογη συνάρτηση για τον αριθμό των tabs που χρησιμοποιούν.

Επιστρέφοντας στο τελικό κομμάτι της MainFrame λοιπόν (**mainFrame1.4**), χρησιμοποιούμε ένα ActionListener για το home button που χρησιμοποιούν όλα τα παράθυρα μετά από αυτό του homepage. Το button αυτό κάθε φορά που εκτελείται παραπέμπει στην αρχική σελίδα (homepage). Οπότε, φτιάχνουμε το Listener μία φορά μέσα στον κώδικα, και όχι για κάθε παράθυρο ξεχωριστά, όπως κάναμε πριν για άλλα buttons, με τη λογική ότι όταν θα εκτελείται θα κλείνουν όλα τα παράθυρα του

προγράμματος (ανεξάρτητα του που βρισκόμαστε δηλαδή) και θα ανοίγουν τα panels του homepage. Ένα κομμάτι του τμήματος κώδικα αυτού είναι το εξής:

```
// HOME - EXIT BUTTONS
      jButton1.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {

            plotoGr.getjPanel1().setVisible(false);
            plotoGr.getjPanel2().setVisible(false);
            plotoGr.getjPanel3().setVisible(false);
            peskesiGr.getjPanel1().setVisible(false);
            peskesiGr.getjPanel2().setVisible(false);
            peskesiGr.getjPanel3().setVisible(false);
            mareGr.getjPanel1().setVisible(false);
            mareGr.getjPanel2().setVisible(false);
            mareGr.getjPanel3().setVisible(false);
            home.getjPanel1().setVisible(true);
            home.getjPanel2().setVisible(true);
            home.getjPanel3().setVisible(true);
            add(home.getjPanel1(), BorderLayout.NORTH);
            add(home.getjPanel2(), BorderLayout.CENTER);
            add(home.getjPanel3(), BorderLayout.SOUTH);
          }
      });
```

Τέλος, στο τελευταίο κομμάτι του constructor της **mainFrame**, κάνουμε το παράθυρο μας ορατό, προκειμένου όταν τρέχει το πρόγραμμα, να μπορεί το παράθυρο να ανοίξει.

```
setVisible(true);
```

Στη συνέχεια, κλείνουμε τον constructor και τη **mainframe.**

5. ΤΡΌΠΟΙ ΒΕΛΤΊΩΣΗΣ

Ο αρχικός στόχος, όπου ήταν η ανάπτυξη αυτής της εφαρμογής, έχει επιτευχθεί. Παρ'όλα αυτά όπως κάθε εφαρμογή υπάρχουν τρόποι βελτίωσης τους οποίους θα αναφέρουμε αρχικά και έπειτα θα αναπτύξουμε περαιτέρω.

- Δυνατότητα συνδεσιμότητας με το διαδίκτυο.
- Πρόσθεση επιπλέον νομών εκτός από τον νομο Ηρακλείου.
- Επέκταση της εφαρμογής σε φορητές συσκευές τηλεφώνων.

Δυνατότητα συνδεσιμότητας με το διαδίκτυο :

Η εφαρμογή στο στάδιο που βρίσκεται τώρα παρέχει όλες τις πληροφορίες σε κατάσταση εκτός σύνδεσης. Στο κομμάτι πληροφοριών τοποθεσίας θα μπορούσε να ενσωματωθεί δυνατότητα GPS[4] το οποίο θα είναι πλήρως διαδραστικό (μεγένθυση, μετακίνηση, πληροφορίες κοντινών σημείων ενδιαφέροντος). Η διαφορά με αυτήν την αλλαγή είναι οτι αυτή η στιγμή η εφαρμογή έχει την δυνατότητα λειτουργίας χωρίς να χρειάζεται σύνδεση διαδικτύου.

Πρόσθεση επιπλέον νομών εκτός από τον νομο Ηρακλείου :

Υπάρχει, επίσης η δυνατότητα πρόσθεσης και άλλων νομών της Κρήτης, καθώς και οποιασδήποτε αλλης περιοχής της Ελλάδας. Για να προστεθούν οι επιπλέον νομοί χρειάζεται να γίνει ανάπτυξη χρησιμοποιώντας πληροφορίες απο γνωστούς τόπους και καταστήματα στην κοινωνία της περιοχής που χρειάζεται.

Επέκταση της εφαρμογής σε φορητές συσκευές τηλεφώνων :

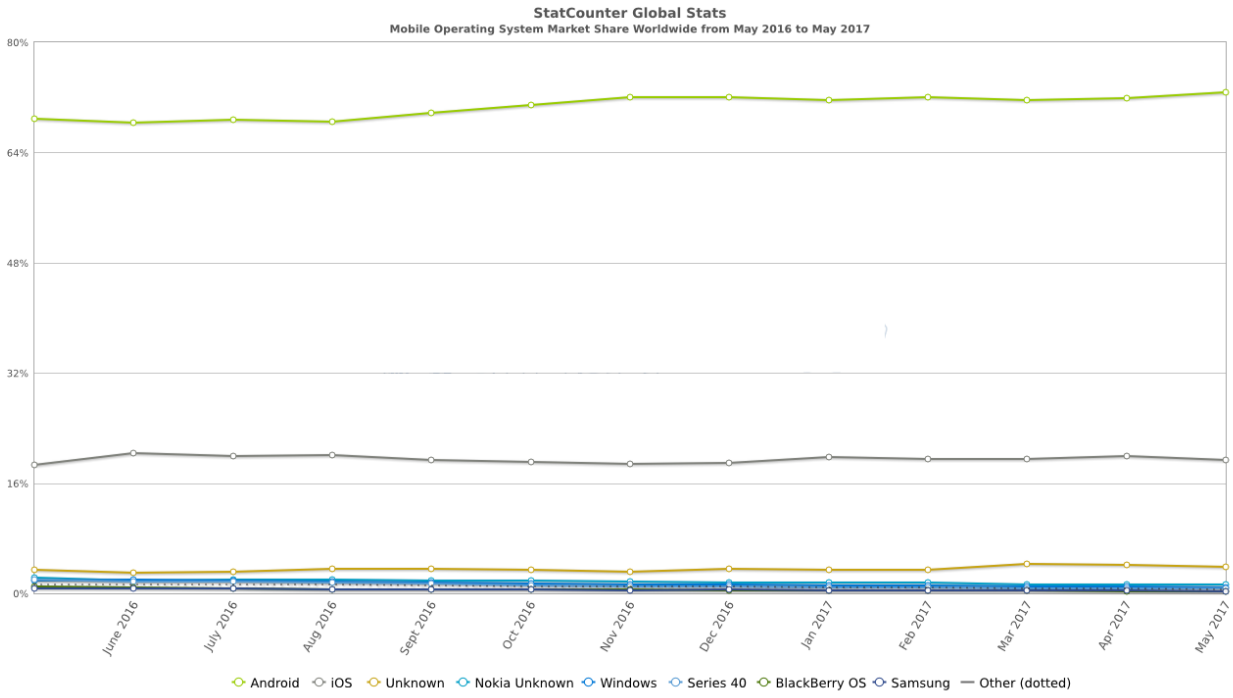
Η σημαντικότερη βελτίωση που θα μπορούσε να έχει ο οδηγός είναι η μεταφορά του σε λογισμικό φορητών όπως το "Android" λειτουργικό σύστημα.

[4] Global Positioning System: Δορυφορικό σύστημα εντόπισμου τοποθεσίας.

Το λειτουργικό σύστημα "Android" αναπτύχθηκε απο την "Google", είναι βασισμένο στο kernel[5] ενός αλλου λειτουργικού συστήματος, του "Linux" και είναι σχεδιασμένο για να τρέχει κυρίως σε φορητές συσκευές τηλεφώνων με οθόνες αφής.

Το "Android" αναπτύχθηκε αρχικά από την εταιρεία "Android Inc" που ιδρύθηκε το 2003 από τους Andy Rubin, Rich Miner, Nick Sears και Chris White. Ο αρχικός στόχος του λογισμικού ήταν να λειτουργεί σε ψηφιακές φωτογραφικές κάμερες αλλά αργότερα διαπίστωσαν τις μεγάλες δυνατότητες τις οποίες είχε στο να γνωρίζει πληροφορίες όπως τις προτιμήσεις και την τοποθεσία των χρηστών.Το 2005, η εταρεία εξαγοράστηκε από την "Google" η οποία από εκείνη την περίοδο είναι υπεύθυνη για την διαχείρηση, συντήρηση καί την αναβάθμιση του λειτουργικού συστήματος.

Ο λόγος για το οποίο η εφαρμογή μας θα ήταν καλό να μεταφερθεί και σε "Android" είναι λόγο του ποσοστού χρήσης του λειτουργικού συστήματος αυτού σε σχέση με τα υπόλοιπα λειτουργικά συστήματα που υπαρχουν στην αγορά.



29 Γράφημα απεικόνισης χρήσης λογισμικών φορητών συσκευών τηλεφωνίας.

Βλέπουμε στην παραπάνω εικόνα ότι το ποσοστό χρηστών με λειτουργικό "Android" είναι εμφανώς μεγαλύτερο σε σχέση με τα υπόλοιπα, επομένως το μέρος της αγοράς στόχου που θα έχει την δυνατότητα να κάνει χρήση της εφαρμογής και να επωφεληθεί από αυτά που προσφέρει, είναι το μεγαλύτερο δυνατό για την συγκεκριμμένη χρονική περίοδο.

[5] Το Kernel ενός λειτουργικού συστήματος είναι το είναι το βασικό του πρόγραμμα που έχει πλήρη έλεγχο του συστήματος.

Η αλλαγή αυτή θα μπορούσε επίσης να γίνει σε συνδυασμό με την πρώτη αλλαγή που προτίναμε, δηλαδή να έχει την δυνατότητα σύνδεσης στο διαδίκτυο και έτσι να κάνει ακόμα μεγαλύτερη αξιοποίηση κάποιων χαρακτηριστικών του φορητών συσκευών, όπως την μετακίνηση με την βοήθεια του GPS.

6. ΠΙΝΑΚΕΣ

1. Εικόνων.

7. Βιβλιογραφία.

https://www.google.gr

https://wikipedia.org

http://www.ironspider.ca

https://www.w3schools.com//

https://notepad-plus-plus.org

http://alexdp.free.fr/violetumleditor/page.php

https://www.javatpoint.com/

http://sete.gr

http://www.statistics.gr