Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

Τίτλος: Σχεδίαση και ανάπτυξη εφαρμογής υπολογιστικής και συγκριτικής ικανότητας για αυτόματη διόρθωση γραπτών πολλαπλής επιλογής.

Καλαμουδάκος Κωνσταντίνος (ΑΜ: 1880)

Μανωλαράκης Γεώργιος (ΑΜ: 2049)

Επιβλέπων καθηγητής: Παπαδάκης Νικόλαος

Επιτροπή Αξιολόγησης: Κορνάρος Γεώργος, Μανιφάβας Χαράλαμπος & Παπαδάκης Νικόλαος

Ημερομηνία Παρουσίασης:

Ευχαριστίες

Ευχαριστούμε πολύ όσους συνέλαβαν στην ολοκλήρωση της πτυχιακής μας εργασίας. Τους γονείς μας που με την υποστήριξή τους, μας βοήθησαν να ολοκληρώσουμε τις σπουδές μας. Είμαστε ευγνώμονες στον εισηγητή μας κ. Παπαδάκη Ν. για την ιδέα της πτυχιακής εργασίας και για την βοήθεια που προσέφερε. Την Πέπη Μούντανου για την πολύτιμη βοήθεια που μας προσέφερε κατά τη συγγραφή του Word *και του* PowerPoint *της πτυχιακής μας εργασίας. Τέλος, ευχαριστούμε τον* designer *Ηλία Καλαμουδάκο για την συμβολή του στο* logo *της εφαρμογής μας.*

Abstract

The purpose of this project is to create a reliable application that effectively, easily and quickly is able to correct automatically a large number of multiple choice exams, extract the grades and correspond them with the Number Register of the examinees.

The basic technology that used, is Platform NetBeans. The application developed and materialized with the use of Object Oriented Programming (OPP).

Σύνοψη

Σκοπός της παρούσας Πτυχιακής είναι η δημιουργία μιας αξιόπιστης εφαρμογής η οποία με αποτελεσματικότητα, ευκολία και ταχύτητα είναι σε θέση να διορθώνει αυτόματα ένα μεγάλο αριθμό γραπτών πολλαπλής επιλογής, να εξάγει τους βαθμούς και να τους αντιστοιχεί με τους Αριθμούς Μητρώου των εξεταζόμενων.

Η βασική τεχνολογία που χρησιμοποιήθηκε, είναι η πλατφόρμα NetBeans. Η εφαρμογή αναπτύχθηκε και υλοποιήθηκε με τη χρήση Αντικειμενοστραφούς Προγραμματισμού.

Contents

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

1 ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία, όσον αφορά τον εκπαιδευτικό της χαρακτήρα, είναι ένα σημαντικό κομμάτι των σπουδών, διότι παρέχει τη δυνατότητα στον σπουδαστή να μελετήσει, να αναλύσει και να αναπτύξει διάφορα θέματα υπό την καθοδήγηση του εισηγητή της πτυχιακής. Ο σπουδαστής από τη διαδικασία αυτήν αποκτά εις βάθος γνώσεις για το αντικείμενο, και αποκομίζει εφόδια που θα του είναι χρήσιμα στην περαιτέρω επιστημονική και επαγγελματική του πορεία.

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάλυση, σχεδίαση και υλοποίηση μιας εφαρμογής αυτόματης διόρθωσης πολλαπλής επιλογής διαγωνισμάτων. Η συγκεκριμένη εφαρμογή αντλεί τα αρχεία που έχουν μετατραπεί από εικόνα σε κείμενο (txt), μέσω της OCR, και τα συγκρίνει με ένα πρωτότυπο αρχείο για να βρει διαφορές. Στη συνέχεια, αφού έχει εντοπίσει τα λάθη εξάγει τη βαθμολογία των γραπτών και τα αντιστοιχεί με τους Αριθμούς Μητρώων (ΑΜ) των εξεταζόμενων. Για την υλοποίηση όλων των παραπάνω χρησιμοποιήθηκε η πλατφόρμα NetBeans. Για την ανάπτυξή τους χρησιμοποιήθηκε ο Αντικειμενοστραφής Προγραμματισμός. Η εκτέλεση της εφαρμογής απαιτεί την ύπαρξη περιβάλλοντος Java.

Η εφαρμογή θα δίνει τη δυνατότητα στο χρήστη (εκπαιδευτικός κυρίως) να εξάγει βαθμολογίες με λιγότερη προσπάθεια και σε λιγότερο χρόνο, έχοντας σαν αποτέλεσμα να είναι πιο αποδοτικός και αποτελεσματικός στην εργασία του.

1.1 ΚΙΝΗΤΡΑ ΓΙΑ ΤΗΝ ΔΙΕΞΑΓΩΓΗ ΤΗΣ ΕΡΓΑΣΙΑΣ

Με την αύξηση του αριθμού εισακτέων φοιτητών σε Τεχνολογικά Ιδρύματα, αλλά και με τη μείωση του εκπαιδευτικού προσωπικού, οι εκπαιδευτικοί τείνουν να χρησιμοποιούν όλο και περισσότερο διαγωνίσματα τύπου πολλαπλής επιλογής, γιατί απαιτούν μικρό χρόνο για τη διόρθωσή τους. Η παρούσα Πτυχιακή Εργασία στοχεύει στην ακόμη πιο γρήγορη και εύκολη διόρθωση γραπτών πολλαπλής επιλογής. Αυτό θα επιτευχθεί με την κατασκευή ενός εργαλείου αυτόματης διόρθωσης και εξαγωγής βαθμών πολλαπλών γραπτών.

1.2 ΣΚΟΠΟΣ ΚΑΙ ΣΤΟΧΟΙ ΕΡΓΑΣΙΑΣ

Σκοπός της Πτυχιακής εργασίας είναι η ανάπτυξη λογισμικού, που θα μπορεί να χρησιμοποιηθεί για την ταχύτερη και ευκολότερη διόρθωση πολλαπλών διαγωνισμάτων (πολλαπλής επιλογής).

Στόχος της Πτυχιακής Εργασίας αποτελεί η χρησιμοποίηση της εφαρμογής από το ΑΤΕΙ Ηρακλείου Κρήτης και συνεπώς από το εκπαιδευτικό προσωπικό του. Επιθυμούμε μακροπρόθεσμα να γίνει ένα καθημερινό λειτουργικό εργαλείο στα χέρια του μειωμένου και ελλιπές προσωπικού του ΑΤΕΙ, ώστε να επιτύχουμε τελικά τη μείωση του επί του παρόντος υπέρογκου φόρτου εργασίας τους και την πιο γρήγορη πληροφόρηση των αποτελεσμάτων των φοιτητών.

1.3 ΔΟΜΗ ΕΡΓΑΣΙΑΣ

Η δομή της εργασίας περιλαμβάνει τα παρακάτω κεφάλαια:

Στο πρώτο Κεφάλαιο, γίνεται μια εισαγωγική περιγραφή για το σκοπό και το κίνητρο της πτυχιακής εργασίας.

Στο δεύτερο Κεφάλαιο, περιγράφονται η μεθοδολογία υλοποίησης και το σχέδιο δράσης για την εκπόνηση της εργασίας, καθώς και οι θεωρίες που χρειάζονται.

Στο τρίτο Κεφάλαιο, διαδραματίζεται το κύριο μέρος της εργασίας μας όπου αναλύουμε τις κινήσεις που έγιναν για να δημιουργηθεί η εφαρμογή διεξοδικά και παρατίθεται η πλήρης μορφή του κώδικά μας.

Στο τέταρτο Κεφάλαιο, παρατίθεται ένα εγχειρίδιο χρήσης της εφαρμογής, το οποίο έχει ως σκοπό τη διευκόλυνση των χρηστών.

Στο πέμπτο Κεφάλαιο, εξάγονται τα αποτελέσματα και τα συμπεράσματα από την υλοποίηση αυτής της πτυχιακής, καθώς και οι επεκτάσεις της.

2 ΜΕΘΟΔΟΛΟΓΙΑ & ΣΧΕΔΙΟ ΔΡΑΣΗΣ ΕΚΠΟΝΗΣΗΣ ΕΡΓΑΣΙΑΣ

Για να μπορέσουμε να αναπτύξουμε μία εφαρμογή αυτόματης διόρθωσης πολλαπλής επιλογής διαγωνισμάτων, θα πρέπει να εκμεταλλευτούμε τις πληροφορίες που θα μπορέσουμε να εξάγουμε από αυτά. Αρχικά, με τη βοήθεια ενός σκάνερ θα πάρουμε σε μορφή εικόνας τα γραπτά διαγωνίσματα των εξεταζόμενων, στη συνέχεια με την χρήση μιας εφαρμογής OCR, θα μετατρέψουμε τις εικόνες αυτές σε αρχεία κειμένου(.txt) και τέλος, θα εισάγουμε στο πρόγραμμα μας τα αρχεία (.txt) και θα τα συγκρίνουμε με το πρωτότυπο αρχείο σωστών αποτελεσμάτων.

Πιο συγκεκριμένα, όσον αφορά τη μορφοποίηση των γραπτών, το scanner παρέχουν τη δυνατότητα μετατροπής των χειρόγραφων εγγράφων σε εικόνα. Υπάρχει δε, η επιλογή μετατροπής πολλαπλών γραπτών στη σειρά, έτσι μπορούμε να εισάγουμε όλα τα γραπτά σε αυτόν και να εξάγουμε απευθείας όλες τις εικόνες σε έναν φάκελο. Άρα αυτή η διαδικασία δεν είναι χρονοβόρα για τον χρήστη. Όμως, για να είναι σε θέση ο χρήστης να εκμεταλλευτεί όλες τις πληροφορίες, είναι απαραίτητη η χρήση ενός προγράμματος που μετατρέπει την εικόνα σε κείμενο(txt). Τέτοια εφαρμογή είναι η OCR που μας επιτρέπει να εξάγουμε σε έναν φάκελο, όλες τις εικόνες τροποποιημένες σε .txt αρχεία. Η OCR δίνει την δυνατότητα στο χρήστη να εισάγει τον φάκελο και να μετατρέψει τα αρχεία του ταυτόχρονα. Δυστυχώς πρόκειται για μία εφαρμογή που δεν έχει φτάσει ακόμη στην βέλτιστη μορφή της. Υπάρχουν OCR ελεύθερες στην αγορά που δεν είναι όμως αρκετά λειτουργικές, καθώς δεν μορφοποιούν σωστά όλους τους γραφικούς χαρακτήρες. Υπάρχουν επίσης και επί πληρωμή OCR, οι οποίες δεν μπορούν να αποκτηθούν από τον κοινό χρήστη(υψηλό αντίτιμο). Με την εξέλιξη της OCR, η εφαρμογή μας θα είναι ακόμη πιο λειτουργική.

Στη συνέχεια, θα χρησιμοποιήσουμε την πλατφόρμα NetBeans, που μας επιτρέπει να αναπτύξουμε εύκολα και σχετικά γρήγορα εφαρμογές που βασίζονται στον αντικειμενοστραφή προγραμματισμό. Ο αντικειμενοστραφής προγραμματισμός είναι η πιο κατάλληλη γλώσσα προγραμματισμού γιατί μπορεί να εκτελεστεί από πολλούς χρήστες χωρίς να υπάρχουν ανεπιθύμητες παρενέργειες, είναι απλή, υψηλής απόδοσης, ανεξάρτητη αρχιτεκτονικής, κατάλληλη για τον παγκόσμιο ιστό, μαθαίνετε εύκολα και διανέμεται δωρεάν. Θα δημιουργήσουμε λοιπόν μία εφαρμογή που θα δίνει την δυνατότητα εκμετάλλευσης των πληροφοριών, που έχουν συλλεχθεί απ' τα γραπτά. Με την ευελιξία που μας προσφέρει ο αντικειμενοστραφής προγραμματισμός, θα είναι εφικτή η σύγκριση των γραπτών με ένα πρωτότυπο αρχείο και η αυτόματη διόρθωση τους, καθώς και ο υπολογισμός της βαθμολογίας τους. Επίσης, το πρόγραμμα θα δίνει την δυνατότητα επιστροφής της βαθμολογίας όλων των γραπτών και των Αριθμών Μητρώων των φοιτητών αντίστοιχα, σε ένα txt αρχείο. Με τις δυνατότητες που μας δίνει το NetBeans, θα μπορούμε να εισάγουμε ταυτόχρονα πολλά αρχεία από το φάκελο και ο χρήστης δε θα χρειάζεται να επαναλαμβάνει τη διαδικασία για κάθε αρχείο ξεχωριστά. Όλα τα παραπάνω καθιστούν την εφαρμογή μας πολύ χρήσιμη και λειτουργική για το χρήστη.

2.1 ΟΙ ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Υπάρχουν 5 γενιές γλωσσών προγραμματισμού:

1) *Η πρώτη γενιά (1940).* Οι γλώσσες 1ης γενιάς είναι γλώσσες προγραμματισμού σε επίπεδο μηχανής. Τα προγράμματα εκτελούνται με μεγάλη ταχύτητα και αποδοτικότητα καθώς εκτελούνται από την CPU. Η γλώσσα μηχανής είναι δύσκολη στην μάθηση. Πολύ δύσκολα εντοπίζονται και διορθώνονται σφάλματα. Αν προσθέσουμε εντολές τότε από το σημείο προσθήκης και κάτω θα πρέπει να μετακινηθούν όλες οι διευθύνσεις μνήμης. Η μεταφορά (portability) προγραμμάτων είναι πρακτικά αδύνατη. Θέματα αρχιτεκτονικής δυσκολεύουν την μεταφορά (π.χ. registers στην αρχιτεκτονική μιας CPU συνήθως διαφέρουν από αυτούς άλλης CPU).

Program 8-3. 64 Search BASIC Loader.

```
799 X=PEEK(55):POKE55,X-1:REM PROTECT ML
800 FOR ADRES=40800TO40913:READ DATTA:
    POKE ADRES,DATTA:NEXT ADRES
900 PRINT"SYS40800 TO ACTIVATE"
4096 DATA 162, 0, 173, 1, 8, 133
4102 DATA 165, 173, 2, 8, 133, 166
4108 DATA 160, 0, 177, 165, 208, 6
4114 DATA 200, 177, 165, 208, 1, 96
4120 DATA 160, 0, 177, 165, 141, 167
4126 DATA 0, 200, 177, 165, 141, 168
4132 DATA 0, 200, 177, 165, 133, 57
4138 DATA 200, 177, 165, 133, 58, 165
4144 DATA 165, 24, 105, 4, 133, 165
4150 DATA 165, 166, 105, 0, 133, 166
4156 DATA 160, 0, 177, 165, 240, 28
4162 DATA 205, 6, 8, 240, 4, 200
4168 DATA 76, 158, 159, 162, 0, 232
4174 DATA 200, 189, 6, 8, 240, 7
4180 DATA 209, 165, 240, 245, 76, 158
4186 DATA 159, 32, 201, 159, 165, 167
4192 DATA 133, 165, 165, 168, 133, 166
4198 DATA 76, 108, 159, 32, 201, 189
4204 DATA 169, 32, 32, 210, 255, 96
READY.
```

Εικόνα 1: 1ης γενιάς Γλώσσα Προγραμματισμού

2) *Η δεύτερη γενιά (1950).* Δεύτερης γενιάς προγραμματιστική γλώσσα θεωρούμε τις γλώσσες assembly. Ο κώδικας μπορεί να γραφεί και να διαβαστεί από προγραμματιστές. Για να τρέξει

σε έναν Η/Υ θα πρέπει να μεταφρασθεί σε φόρμα αναγνώσιμη από την CPU. Η διαδικασία μετατροπής ονομάζεται assembly. Η γλώσσα είναι συγκεκριμένη για μια οικογένεια επεξεργαστών και λειτουργικών.

Example listing of assembly language source code

Address	Label	Instruction (AT&T syntax)	Object code[14]
		.begin	
		.org 2048	
	a_start	.equ 3000	
2048		ld length,%	
2064		be done	00000010 10000000 00000000 00000110
2068		addcc %r1,-4,%r1	10000010 10000000 01111111 11111100
2072		addcc %r1,%r2,%r4	10001000 10000000 01000000 00000010
2076		ld %r4,%r5	11001010 00000001 00000000 00000000
2080		ba loop	00010000 10111111 11111111 11111011
2084		addcc %r3,%r5,%r3	10000110 10000000 11000000 00000101
2088	done:	jmpl %r15+4,%r0	10000001 11000011 11100000 00000100
2092	length:	20	00000000 00000000 00000000 00010100
2096	address:	a_start	00000000 00000000 00001011 10111000
		.org a_start	
3000	a:		

Εικόνα 2: 2ης γενιάς Γλώσσα Προγραμματισμού

3) *Η τρίτη γενιά (1955-65).* Η 3GL είναι γλώσσες που σχεδιάστηκαν για να είναι εύκολα κατανοητές στο άνθρωπο και περιλαμβάνουν named variables, abstract data types, και algebraic expression syntax. Μια σημαντική διαφορά από τις 2GL είναι η διαφοροποίηση /

ανεξαρτητοποίηση από τον επεξεργαστή. Οι πρώτες γλώσσες εμφανίστηκαν στα τέλη 1950s. Η FORTRAN, ALGOL και COBOL είναι από τα πρώτα παραδείγματα γλωσσών τρίτης γενιάς.

Γλώσσα	**Παράδειγμα Hello World**
FORTRAN (FORmula TRANslator)	c c Hello, world. c Program Hello implicit none logical DONE DO while (.NOT. DONE) write(*,10) END DO 10 format('Hello, world.') END
COBOL (COmmon Business Orientated Language)	IDENTIFICATION DIVISION. PROGRAM-ID. HELLO-WORLD. ENVIRONMENT DIVISION. DATA DIVISION. PROCEDURE DIVISION. MAIN. DISPLAY 'Hello, world.' STOP RUN.
BASIC (Beginner's All-purpose Symbolic Instruction Code)	Source Code 10 PRINT "Hello World!" 20 GOTO 10

Pascal	program HelloWorld;
	
	begin
	writeln('Hello World');
	end.

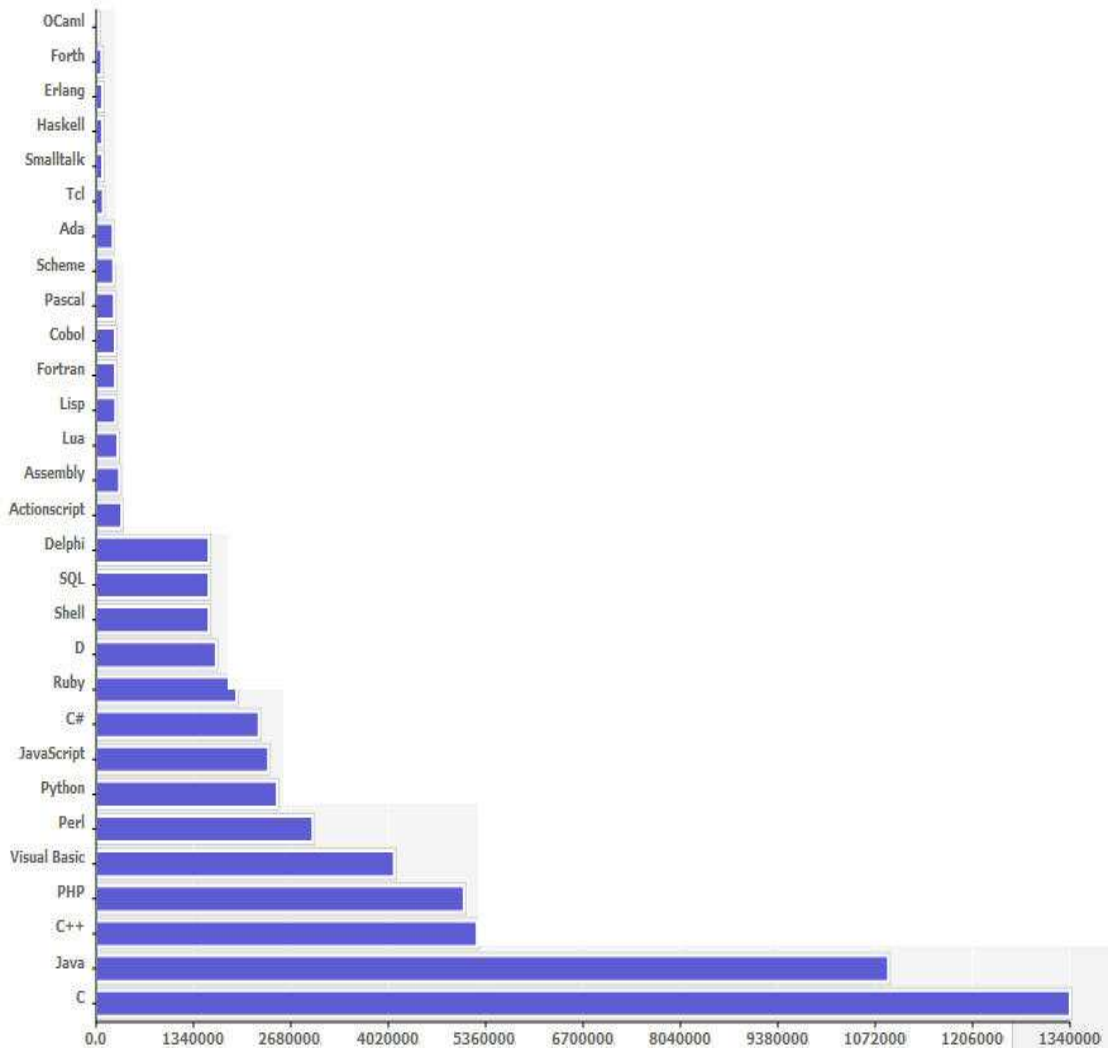
C was developed by Dennis Ritchie at Bell Labs in the mid-1970s.	1
	2 #include <stdio.h>
	3
	4 main()
	5 {
	6 for(;;)
	7 {
	8 printf ("Hello World!\n");
	9 }
	10 }

Εικόνα 3: 3ης γενιάς Γλώσσα Προγραμματισμού

4) *Η τέταρτη γενιά (1980).* Η τέταρτης γενιάς γλώσσα προγραμματισμού (4GL) είναι γλώσσες προγραμματισμού ή περιβάλλοντα προγραμματισμού που σχεδιάστηκαν με συγκεκριμένο σκοπό, όπως την υλοποίηση επιχειρηματικού λογισμικού. Όλες οι 4GLs σχεδιάστηκαν για να

μειώνουν α)την προγραμματιστική προσπάθεια, β)το χρόνο που χρειάζεται για την ανάπτυξη ενός λογισμικού και το κόστος δημιουργίας λογισμικού.

5) *Η πέμπτη γενιά.* Η πέμπτης γενιάς γλώσσα προγραμματισμού (5GL) είναι γλώσσες προγραμματισμού που βασίζονται στην επίλυση προβλημάτων με την χρήση περιορισμών που τίθενται σε ένα πρόγραμμα και όχι με την χρήση αλγόριθμων γραμμένων από προγραμματιστές.



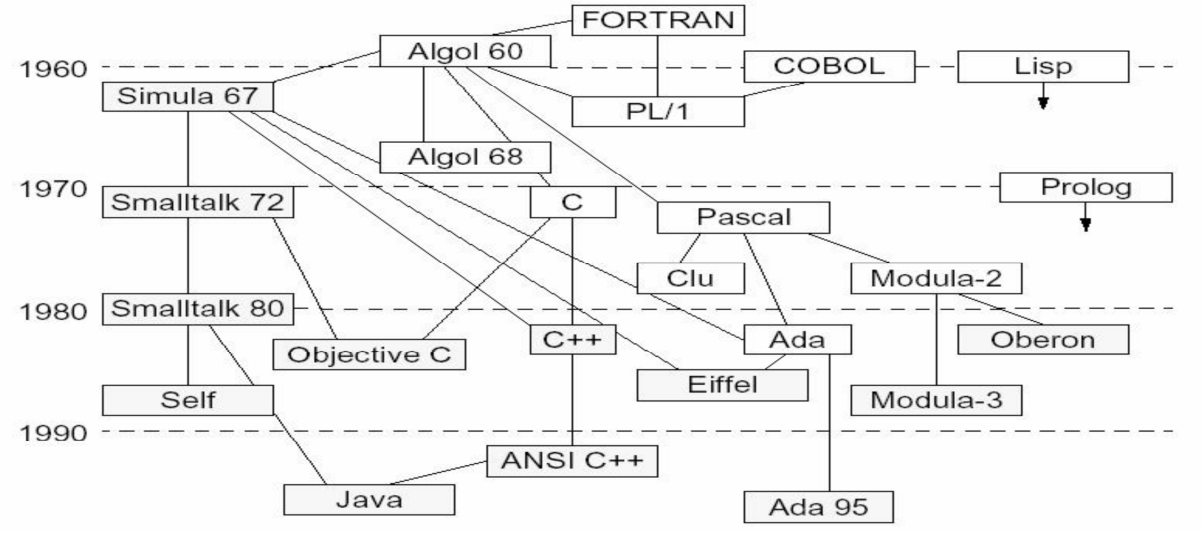
Εικόνα 4: Γλώσσες Προγραμματισμού-Δημοτικότητα

2.1.1 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Στις αρχές του 1991, η *Sun* αναζητούσε το κατάλληλο εργαλείο για να αποτελέσει την πλατφόρμα ανάπτυξης λογισμικού σε μικρο-συσκευές (έξυπνες οικιακές συσκευές έως πολύπλοκα συστήματα παραγωγής γραφικών). Τα εργαλεία της εποχής ήταν γλώσσες όπως η C++ και η C. Μετά από διάφορους πειραματισμούς προέκυψε το συμπέρασμα ότι οι υπάρχουσες γλώσσες δεν μπορούσαν να καλύψουν τις ανάγκες τους. Ο "πατέρας" της Java, James Gosling, που εργαζόταν εκείνη την εποχή για την Sun, έκανε ήδη πειραματισμούς πάνω στη C++ και είχε παρουσιάσει κατά καιρούς κάποιες πειραματικές γλώσσες (C++ ++) ως πρότυπα για το νέο εργαλείο που αναζητούσαν στην *Sun*. Τελικά μετά από λίγο καιρό κατέληξαν με μια πρόταση για το επιτελείο της εταιρίας, η οποία ήταν η γλώσσα *Oak*.

Η *Oak* ήταν μία γλώσσα που διατηρούσε μεγάλη συγγένεια με την C++. Παρόλα αυτά είχε πολύ πιο έντονο αντικειμενοστραφή (*object oriented*) χαρακτήρα σε σχέση με την C++ και χαρακτηριζόταν για την απλότητα της. Σύντομα οι υπεύθυνοι ανάπτυξης της νέας γλώσσας ανακάλυψαν ότι το όνομα *Oak* ήταν ήδη κατοχυρωμένο οπότε κατά την διάρκεια μιας εκ των πολλών συναντήσεων σε κάποιο τοπικό καφέ αποφάσισαν να μετονομάσουν το νέο τους δημιούργημα σε Java που εκτός των άλλων ήταν το όνομα της αγαπημένης ποικιλίας καφέ για τους δημιουργούς της. Η επίσημη εμφάνιση της *Java* αλλα και του *HotJava* (πλοηγός με υποστήριξη *Java*) στη βιομηχανία της πληροφορικής έγινε το Μάρτιο του 1995 όταν η *Sun* την ανακοίνωσε στο συνέδριο *Sun World* 1995. Ο πρώτος μεταγλωττιστής (*compiler*) της ήταν γραμμένος στη γλώσσα C από τον James Gosling. Το 1994, ο A.Van Hoff ξαναγράφει τον μεταγλωττιστή της γλώσσας σε *Java*, ενώ το Δεκέμβριο του 1995 πρώτες οι IBM, Borland, Mitsubishi Electronics, Sybase και Symantec ανακοινώνουν σχέδια να χρησιμοποιήσουν τη *Java* για την δημιουργία λογισμικού. Από εκεί και πέρα η Java ακολουθεί μία ανοδική πορεία και είναι πλέον μία από τις πιο δημοφιλείς γλώσσες στον χώρο της πληροφορικής. Στις 13 Νοεμβρίου του 2006 η Java έγινε πλέον μια γλώσσα ανοιχτού κώδικα(GPL) όσον αφορά το μεταγλωττιστή (javac) και το πακέτο ανάπτυξης (JDK, Java Development Kit).

Στις 27 Απριλίου 2010 η εταιρία λογισμικού Oracle Corporation ανακοίνωσε ότι μετά από πολύμηνες συζητήσεις ήρθε σε συμφωνία για την εξαγορά της Sun Microsystems και των τεχνολογιών (πνευματικά δικαιώματα/ πατέντες) που η δεύτερη είχε στην κατοχή της ή δημιουργήσει. Η συγκεκριμένη συμφωνία θεωρείται σημαντική για το μέλλον της Java και του γενικότερου οικοσυστήματος τεχνολογιών γύρω από αυτή μιας και ο έμμεσος έλεγχος της τεχνολογίας και η εξέλιξη της περνάει σε άλλα χέρια.

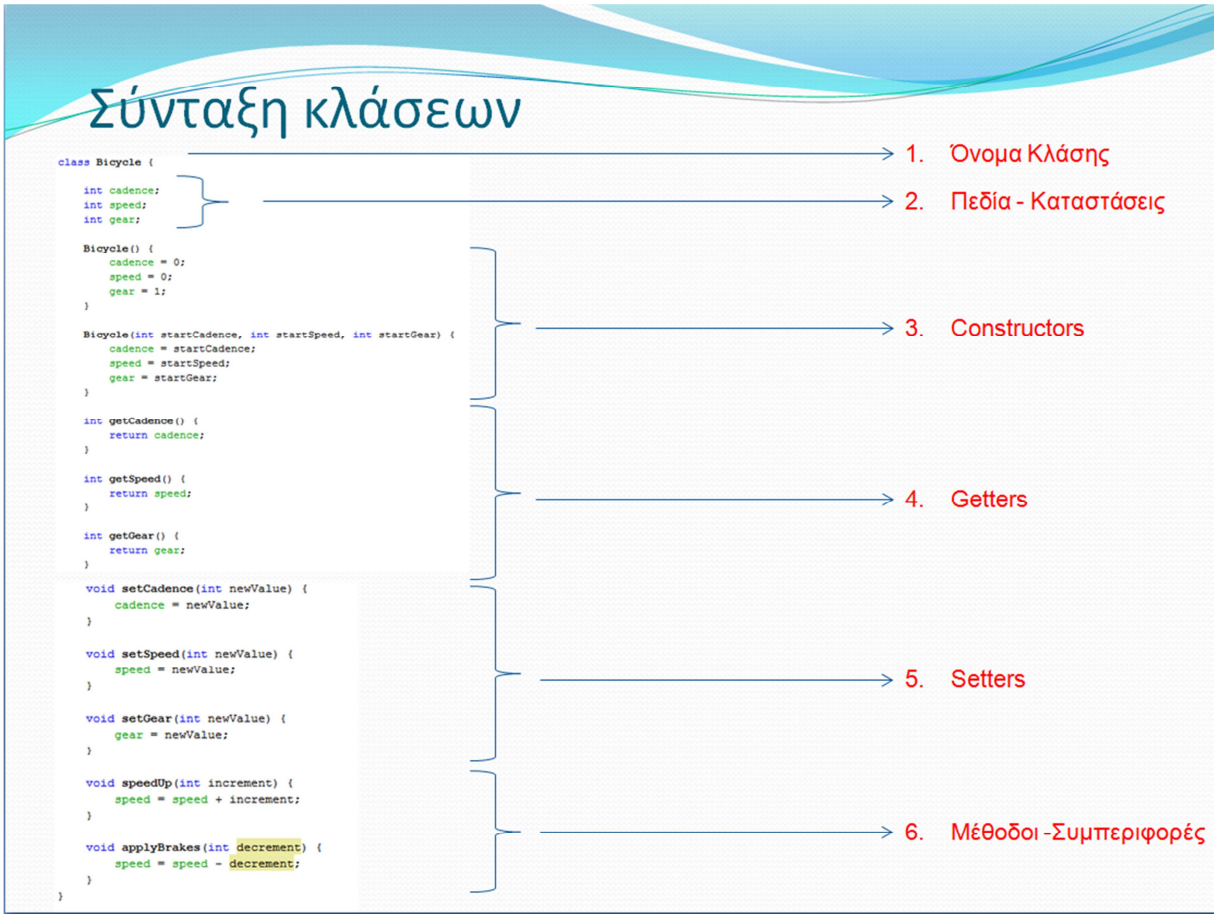


Εικόνα 5: Ιστορική Αναδρομή

2.1.2 ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Ο ***αντικειμενοστραφής προγραμματισμός*** (OOP-Object Oriented Programming), είναι μια προγραμματιστική φιλοσοφία όπως και ο προστακτικός ή ο λογικός προγραμματισμός. Σύμφωνα με τον OPP ένα πρόγραμμα δεν αποτελείται από τα δεδομένα και τον κώδικα που τα επεξεργάζεται αλλά από αντικείμενα (objects) τα οποία εμπεριέχουν τα δεδομένα και τα οποία ανταλλάσουν μεταξύ τους πληροφορίες και μηνύματα προκειμένου να επιτευχθεί ο στόχος του προγράμματος.

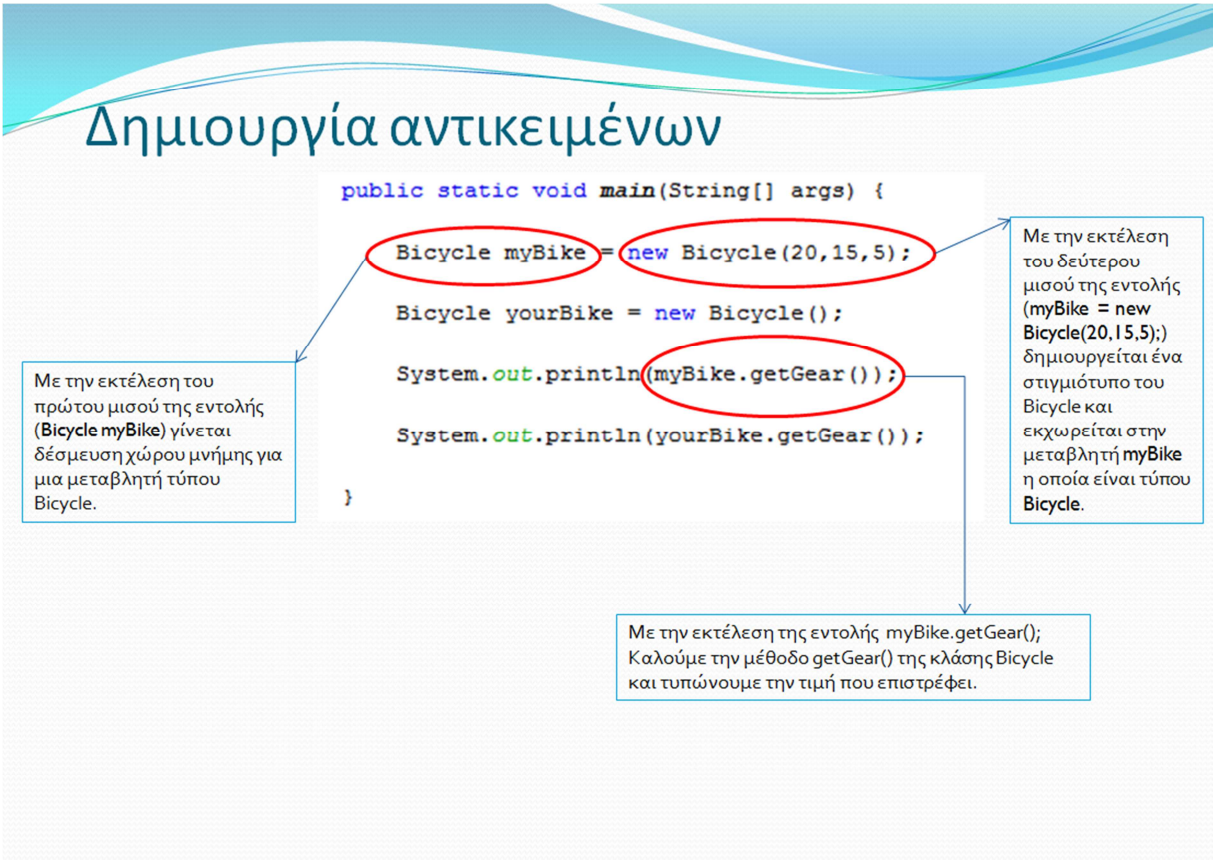
Κεντρική ιδέα στον αντικειμενοστρεφή προγραμματισμό είναι η ***κλάση*** (class), μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων, είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών, εννοιολογικών αντικειμένων, σε ένα περιβάλλον προγραμματισμού. Πρακτικώς είναι ένας τύπος δεδομένων, ή αλλιώς το προσχέδιο μίας δομής δεδομένων με δικά της περιεχόμενα, τόσο μεταβλητές όσο και διαδικασίες. Τα περιεχόμενα αυτά δηλώνονται είτε ως *δημόσια* (public) είτε ως *ιδιωτικά* (private), με τα ιδιωτικά να μην είναι προσπελάσιμα από κώδικα εκτός της κλάσης. Οι διαδικασίες των κλάσεων συνήθως καλούνται *μέθοδοι* (methods) και οι μεταβλητές τους *γνωρίσματα* (attributes) ή *πεδία* (fields). Μία κλάση πρέπει ιδανικά να είναι εννοιολογικά αυτοτελής, να περιέχει δηλαδή μόνο πεδία τα οποία περιγράφουν μία κατηγορία αντικειμένων και δημόσιες μεθόδους οι οποίες επενεργούν σε αυτά όταν καλούνται από το εξωτερικό πρόγραμμα, χωρίς να εξαρτώνται από άλλα δεδομένα ή κώδικα εκτός της κλάσης, και επαναχρησιμοποιήσιμη, να αποτελεί δηλαδή μαύρο κουτί δυνάμενο να λειτουργήσει χωρίς τροποποιήσεις ως τμήμα διαφορετικών προγραμμάτων.



Εικόνα 6: Τρόπος Σύνταξης Κλάσεων

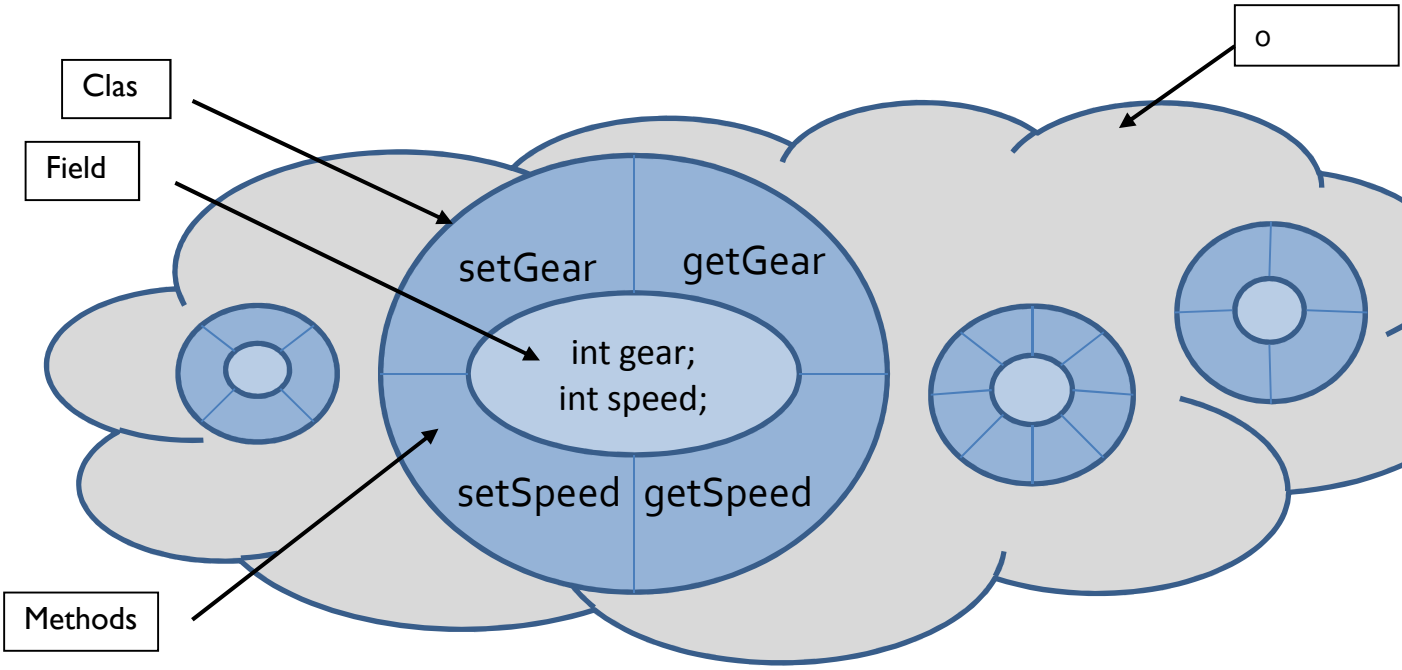
Αντικείμενο (object) είναι το στιγμιότυπο μίας κλάσης, δηλαδή αυτή καθαυτή η δομή δεδομένων (με αποκλειστικά δεσμευμένο χώρο στη μνήμη) βασισμένη στο «καλούπι» που προσφέρει

η κλάση. Παραδείγματος χάρη, σε μία αντικειμενοστραφή γλώσσα προγραμματισμού θα μπορούσαμε να ορίσουμε κάποια κλάση ονόματι BankAccount, η οποία αναπαριστά έναν τραπεζικό λογαριασμό, και να δηλώσουμε ένα αντικείμενο της με όνομα MyAccount. Το αντικείμενο αυτό θα έχει δεσμεύσει χώρο στη μνήμη με βάση τις μεταβλητές και τις μεθόδους που περιγράψαμε όταν δηλώσαμε την κλάση. Έτσι, στο αντικείμενο θα μπορούσε να περιέχεται ένα γνώρισμα Balance (=υπόλοιπο) και μία μέθοδος GetBalance (=επέστρεψε το υπόλοιπο). Ακολούθως θα μπορούσαμε να δημιουργήσουμε ακόμα ένα ή περισσότερα αντικείμενα της ίδιας κλάσης τα οποία θα είναι διαφορετικές δομές δεδομένων (διαφορετικοί τραπεζικοί λογαριασμοί στο παράδειγμα). Ας σημειωθεί εδώ πως τα αντικείμενα μίας κλάσης μπορούν να προσπελάσουν τα ιδιωτικά περιεχόμενα άλλων αντικειμένων της ίδιας κλάσης.



Εικόνα 7: Δημιουργία αντικειμένων

Ενθυλάκωση δεδομένων (data encapsulation) καλείται η ιδιότητα που προσφέρουν οι κλάσεις να «κρύβουν» τα ιδιωτικά δεδομένα τους από το υπόλοιπο πρόγραμμα και να εξασφαλίζουν πως μόνο μέσω των δημόσιων μεθόδων τους θα μπορούν αυτά να προσπελαστούν. Αυτή η τακτική παρουσιάζει μόνο οφέλη καθώς εξαναγκάζει κάθε εξωτερικό πρόγραμμα να φιλτράρει το χειρισμό που επιθυμεί να κάνει στα πεδία μίας κλάσης μέσω των ελέγχων που μπορούν να περιέχονται στις δημόσιες μεθόδους της κλάσης. Δηλώνουμε τα fields ως private και κάνουμε προσπέλαση (ανάγνωση ή αλλαγή τιμής) στις μεταβλητές μιας κλάσης μέσω μεθόδων(π.χ. **setters**, **getters**).



Εικόνα 8: Ενθυλάκωση

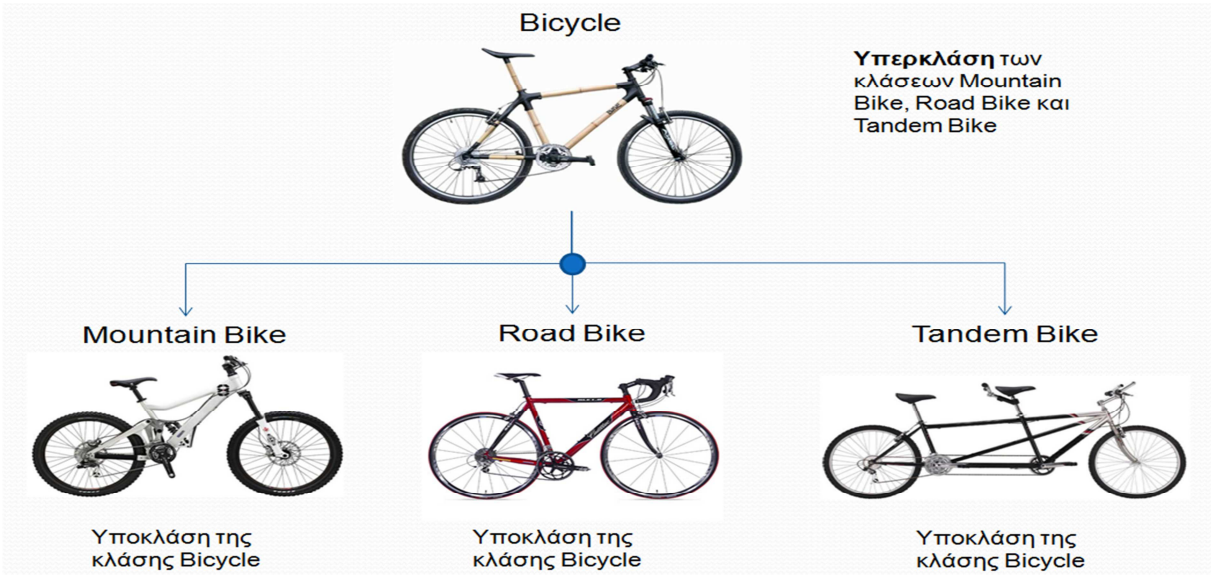
Αφαίρεση δεδομένων καλείται η ιδιότητα των κλάσεων να αναπαριστούν αφαιρετικά πολύπλοκες οντότητες στο προγραμματιστικό περιβάλλον. Μία κλάση αποτελεί ένα αφαιρετικό μοντέλο κάποιας κατηγορίας αντικειμένων. Επίσης οι κλάσεις προσφέρουν και αφαίρεση ως προς τον υπολογιστή, εφόσον η καθεμία μπορεί να θεωρηθεί ένας μικρός και αυτάρκης υπολογιστής (με δική του κατάσταση, μεθόδους και μεταβλητές).

Κληρονομικότητα ονομάζεται η ιδιότητα των κλάσεων να επεκτείνονται σε νέες κλάσεις, ρητά δηλωμένες ως κληρονόμους (*υποκλάσεις* ή 'θυγατρικές κλάσεις'), οι οποίες μπορούν να επαναχρησιμοποιήσουν τις μεταβιβάσιμες μεθόδους και ιδιότητες της γονικής τους κλάσης αλλά και να προσθέσουν δικές τους. Στιγμιότυπα των θυγατρικών κλάσεων μπορούν να χρησιμοποιηθούν όπου απαιτούνται στιγμιότυπα των γονικών (εφόσον η θυγατρική είναι κατά κάποιον τρόπο μία πιο εξειδικευμένη εκδοχή της γονικής), αλλά το αντίστροφο δεν ισχύει. Παράδειγμα κληρονομικότητας είναι μία γονική κλάση Vehicle (=Όχημα) και οι δύο πιο εξειδικευμένες υποκλάσεις της Car (=Αυτοκίνητο) και Bicycle (=Ποδήλατο), οι οποίες λέμε ότι "κληρονομούν" από αυτήν. Πολλαπλή κληρονομικότητα είναι η δυνατότητα που προσφέρουν ορισμένες γλώσσες προγραμματισμού μία κλάση να κληρονομεί ταυτόχρονα από περισσότερες από μία γονικές. Από μία υποκλάση μπορούν να προκύψουν νέες υποκλάσεις που κληρονομούν από αυτήν, με αποτέλεσμα μία ιεραρχία κλάσεων που συνδέονται μεταξύ τους "ανά γενιά" με σχέσεις κληρονομικότητας.

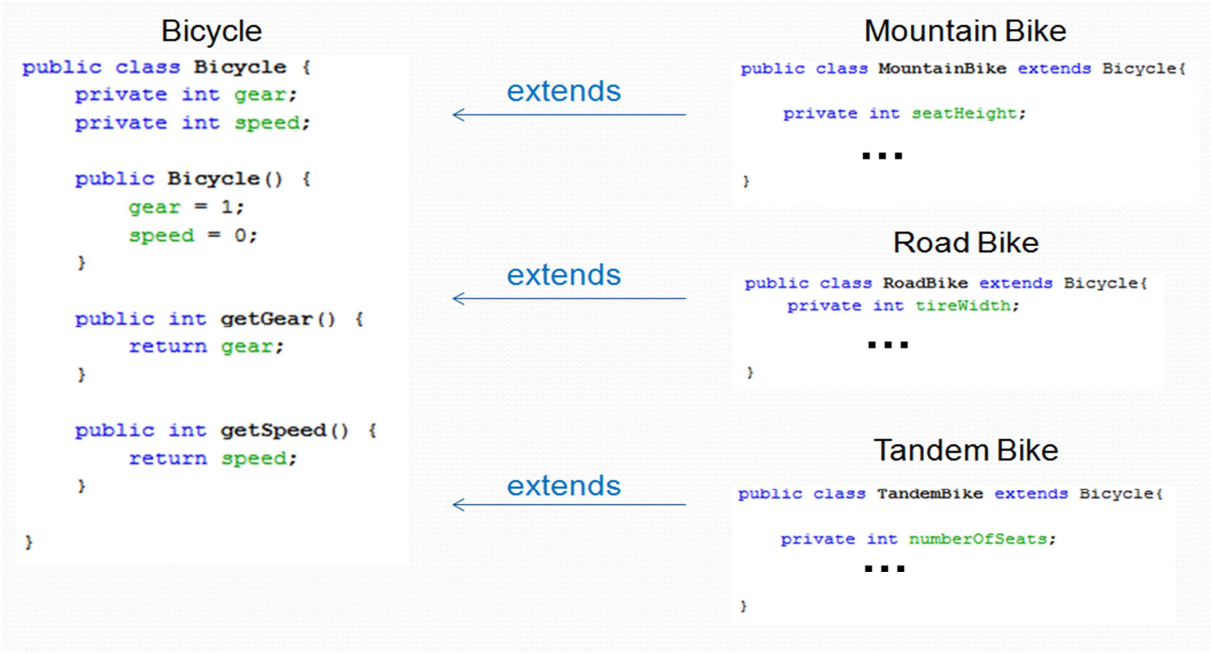
Η ιδέα της κληρονομικότητας είναι απλή αλλά ισχυρότατη. Όταν χρειαστεί να δημιουργηθεί μια καινούργια κλάση και υπάρχει ήδη μια κλάση που περιλαμβάνει μέρος κώδικα που είναι απαραίτητος, μπορεί να προέλθει η νέα κλάση από την ήδη υπάρχων. Με αυτό, μπορεί να επαναχρησιμοποιηθούν πεδία και μέθοδοι από την ήδη υπάρχουσα κλάση χωρίς να χρειαστεί να επαναγραφεί όλος ο κώδικας. Μία υποκλάση κληρονομεί όλα τα μέλη (πεδία, μεθόδους και εμφωλευμένες κλάσεις) από την υπερκλάση της. Οι κατασκευαστές δεν είναι μέλη, έτσι δεν κληρονομούνται από τις υποκλάσεις, αλλά ο κατασκευαστής της υπερκλάσης μπορεί να κληθεί από τις υποκλάσεις.

Μία κλάση που προέρχεται / επεκτείνει (extends) μία άλλη κλάση ονομάζεται υποκλάση (subclass). Η κλάση από την οποία προήλθε (πατρική) ονομάζεται υπερκλάση (superclass). Με εξαίρεση την κλάση Object, η οποία δεν έχει υπερκλάση, όλες οι άλλες έχουν μία και μόνο μία απευθείας υπερκλάση (μοναδιαία κληρονομικότητα). Σε περίπτωση απουσίας προσδιορισμένης κλάσης, η κάθε κλάση είναι αυστηρά υποκλάση της Object. Κλάσεις μπορεί να προέλθουν από

κλάσεις που έχουν προέλθει από άλλες κλάσεις, οι οποίες και αυτές με την σειρά τους έχουν προέλθει , και ούτω καθεξής, με τελική κατάληξη την προέλευση από την κλάση Object.



Εικόνα 9: Κληρονομικότητα 1



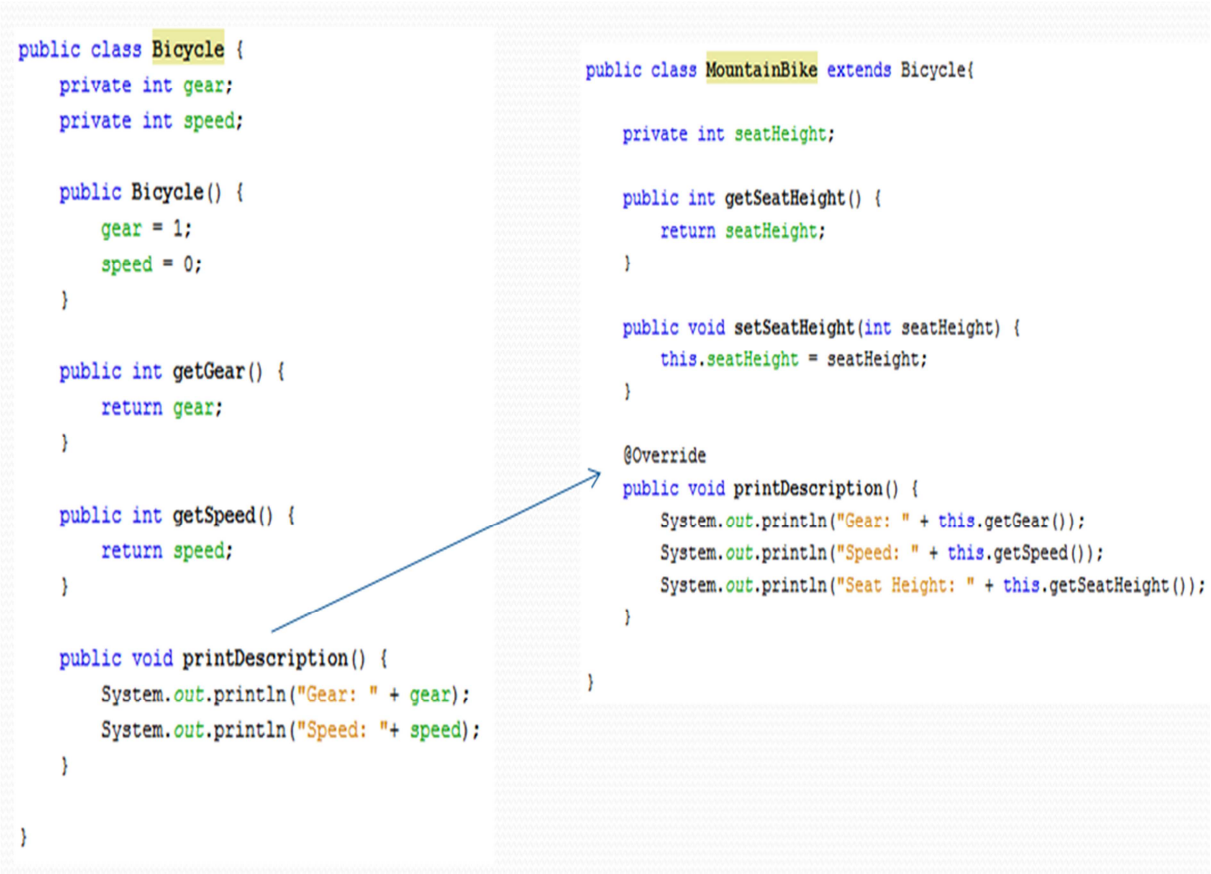
Εικόνα 10: Κληρονομικότητα 2

Υπερφόρτωση μεθόδου (method overloading) είναι η κατάσταση κατά την οποία υπάρχουν, στην ίδια ή σε διαφορετικές κλάσεις, μέθοδοι με το ίδιο όνομα και πιθανώς διαφορετικά ορίσματα. Αν πρόκειται για μεθόδους της ίδιας κλάσης διαφοροποιούνται μόνο από τις διαφορές τους στα ορίσματα και στον τύπο επιστροφής.

```
public class Bicycle {

    int cadence;
    int speed;
    int gear;

    void speedUp(int increment) {
        speed = speed + increment;
    }

    void speedUp() {
        speed += 10;
    }

    void speedUp(int increment, int maxSpeed) {
        if ((speed + increment) < maxSpeed) {
            speed += increment;
        } else {
            speed = maxSpeed;
        }
    }
}
```

Εικόνα 11: Method Overloading

Πολυμορφισμός ονομάζεται η ιδιότητα των υποκλάσεων τόσο να ορίζουν δικές τους συμπεριφορές όσο και ταυτόχρονα να διατηρούν κάποιες άλλες όπως έχουν οριστεί στις υπερκλάσεις τους.



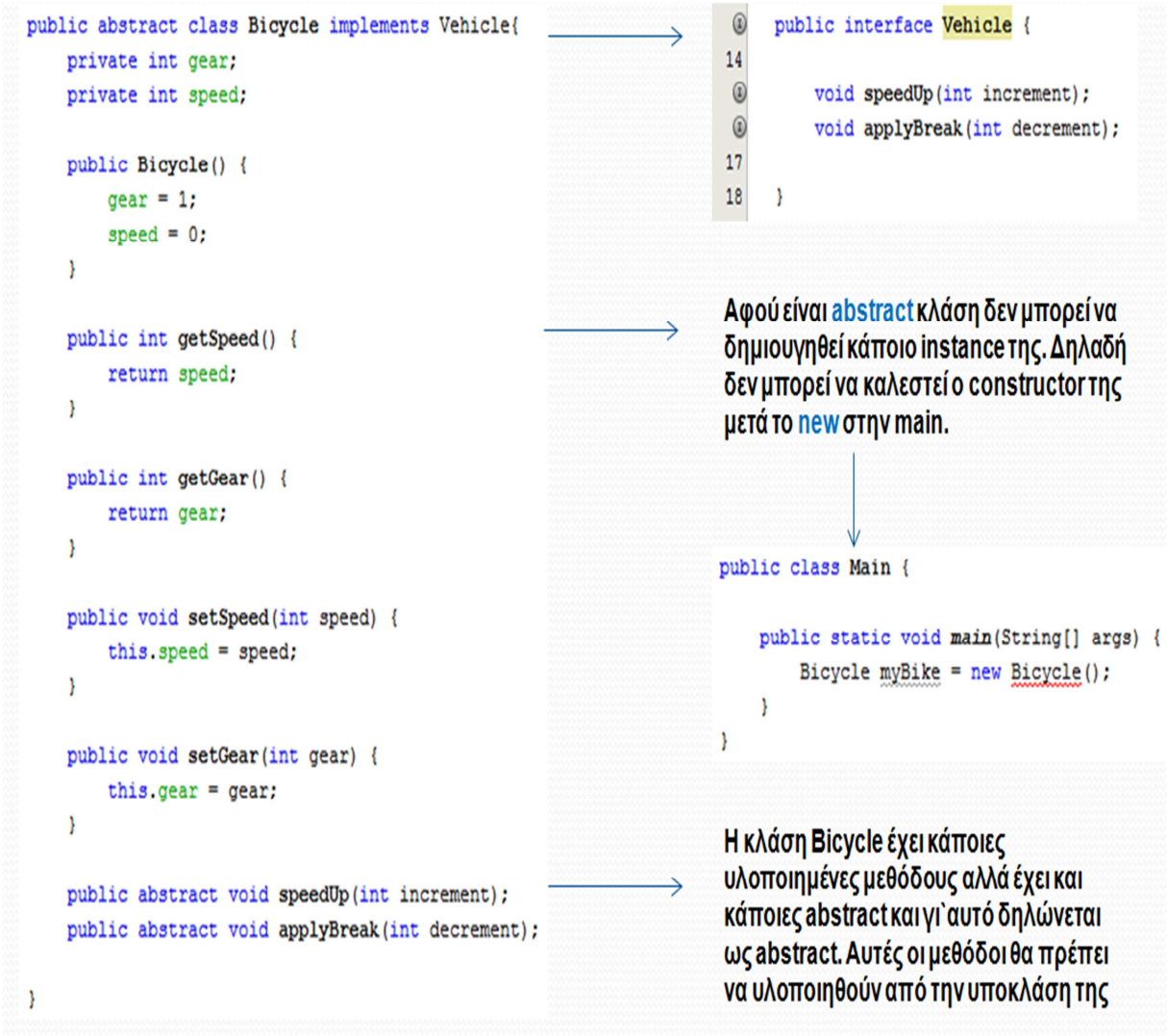
```
public class Bicycle {
    private int gear;
    private int speed;

    public Bicycle() {
        gear = 1;
        speed = 0;
    }

    public int getGear() {
        return gear;
    }

    public int getSpeed() {
        return speed;
    }

    public void printDescription() {
        System.out.println("Gear: " + gear);
        System.out.println("Speed: "+ speed);
    }

}
```

```
public class MountainBike extends Bicycle{

    private int seatHeight;

    public int getSeatHeight() {
        return seatHeight;
    }

    public void setSeatHeight(int seatHeight) {
        this.seatHeight = seatHeight;
    }

    @Override
    public void printDescription() {
        System.out.println("Gear: " + this.getGear());
        System.out.println("Speed: " + this.getSpeed());
        System.out.println("Seat Height: " + this.getSeatHeight());
    }

}
```

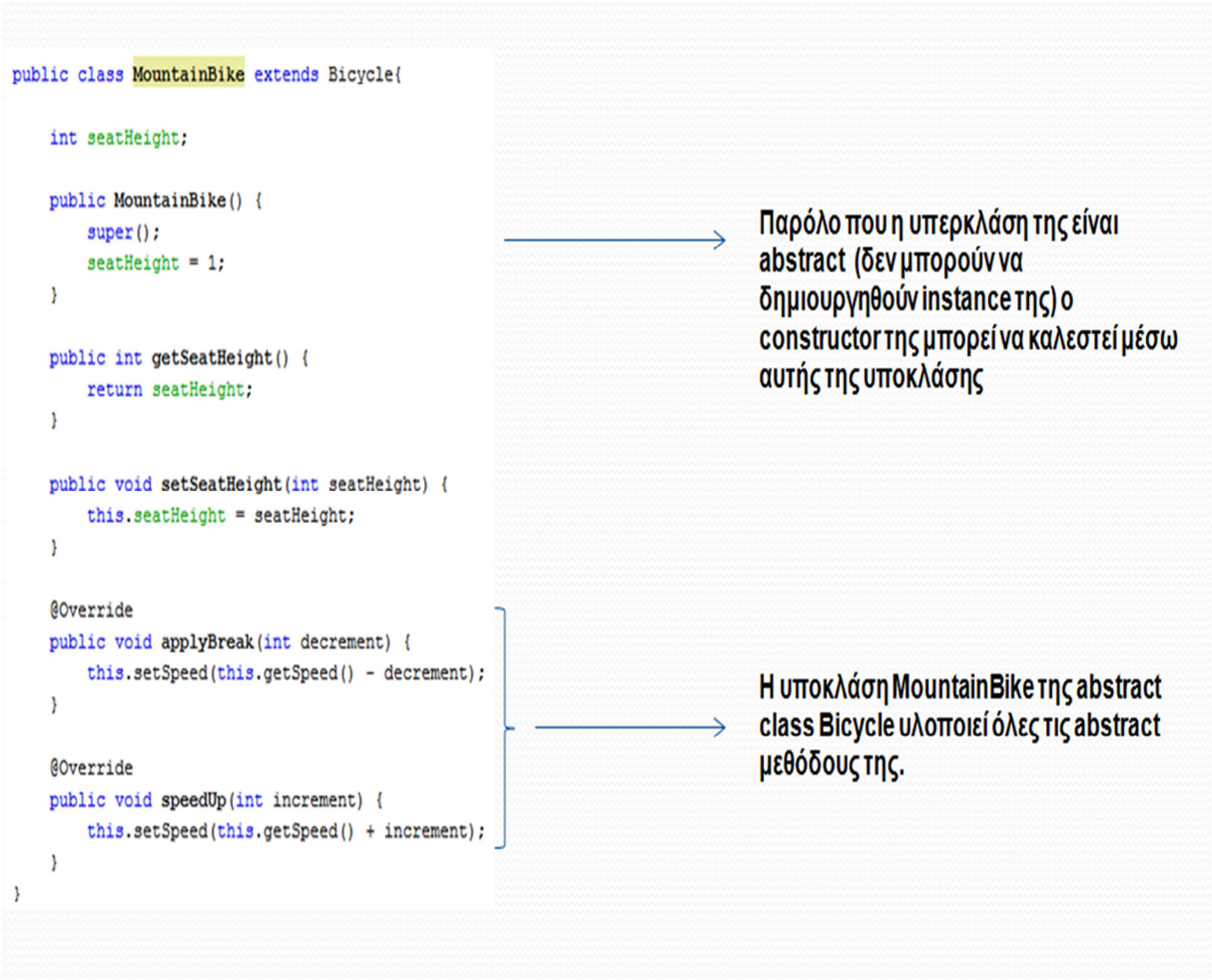
Εικόνα 12: Πολυμορφισμός-Method Overriding

Η διαδικασία αυτή λέγεται "***method overriding***" (***Υποσκέλιση μεθόδου***). Η method overriding είναι η κατάσταση κατά την οποία μία θυγατρική κλάση και η γονική της έχουν μία μέθοδο ομώνυμη και με τα ίδια ορίσματα. Χάρη στη δυνατότητα του *πολυμορφισμού* ο μεταγλωττιστής «ξέρει» πότε να καλέσει ποια μέθοδο, βασισμένος στον τύπο του τρέχοντος αντικειμένου.

Αφηρημένη κλάση (abstract class) είναι μία κλάση που ορίζεται μόνο για να κληρονομηθεί σε θυγατρικές υποκλάσεις και δεν υπάρχουν δικά της στιγμιότυπα (αντικείμενα). Η αφηρημένη κλάση ορίζει απλώς ένα "συμβόλαιο" το οποίο θα πρέπει να ακολουθούν οι υποκλάσεις της όσον αφορά τις υπογραφές των μεθόδων τους (όπου ως υπογραφή ορίζεται το όνομα, τα ορίσματα και η τιμή επιστροφής μίας διαδικασίας). Μία αφηρημένη κλάση μπορεί να έχει και μη αφηρημένες μεθόδους οι οποίες υλοποιούνται στην ίδια την κλάση (αν και φυσικά μπορούν να υποσκελίζονται σε υποκλάσεις). Αντιθέτως οι αφηρημένες μέθοδοί της είναι απλώς ένας ορισμός της υπογραφής τους και εναπόκειται στις υποκλάσεις να τις υλοποιήσουν. Μία αφηρημένη κλάση που δεν έχει γνωρίσματα και όλες οι μέθοδοί της είναι αφηρημένες και δημόσιες καλείται **διασύνδεση** (interface). Οι κλάσεις που κληρονομούν από μία διασύνδεση λέγεται ότι την "υλοποιούν".

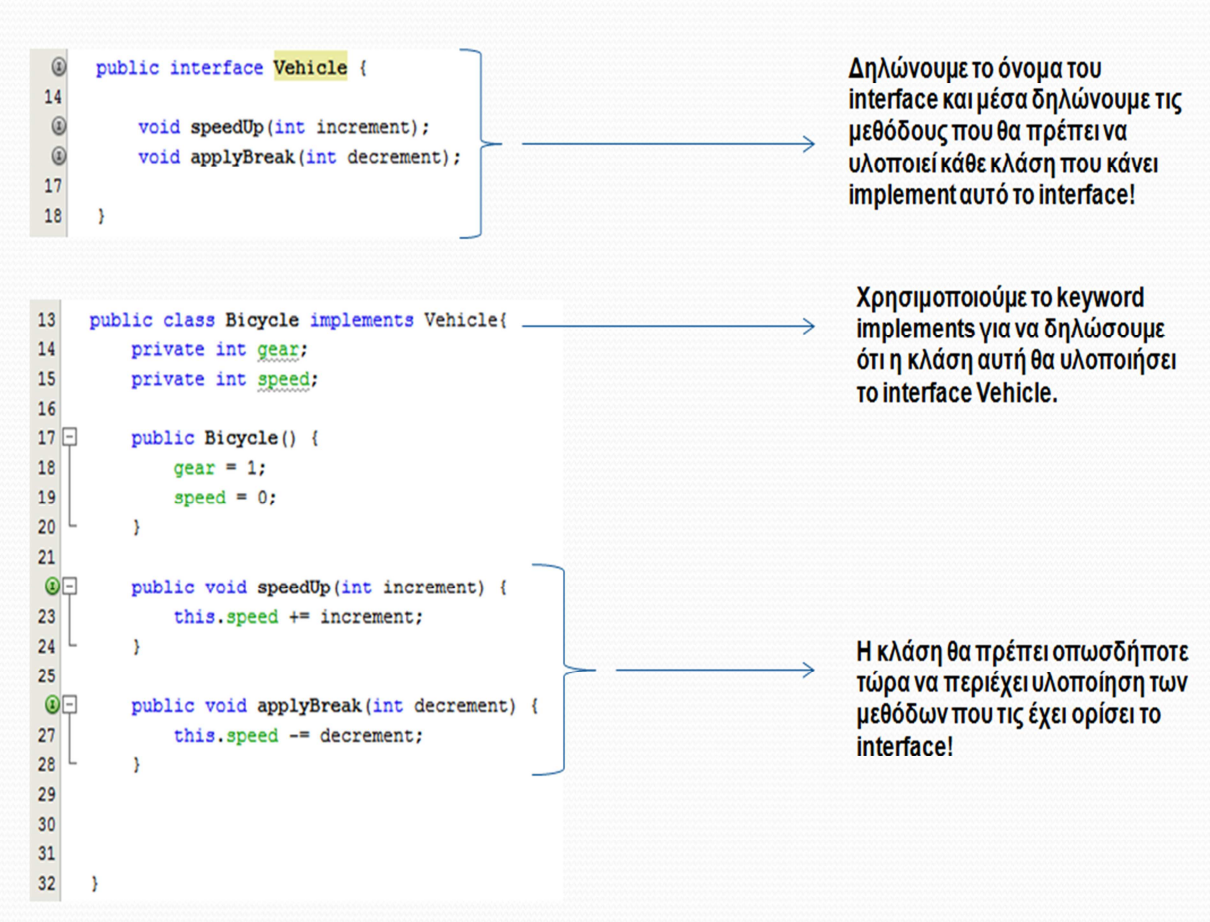


Εικόνα 13: Abstract class 1



Εικόνα 14: Abstract class 2

Interface είναι ένα «συμβόλαιο» που ορίζει σε μία κλάση αντικειμένων το πώς θα αλληλεπιδράει με άλλες κλάσεις και τον υπόλοιπο κόσμο. Αποτελεί μια δομή η οποία δηλώνεται με το keyword **interface** και περιέχει μία **συλλογή από μη υλοποιημένες μεθόδους**. Οι μέθοδοι αυτοί ορίζουν στην ουσία τις συμπεριφορές που θα πρέπει οπωσδήποτε να έχει ένα αντικείμενο που «υλοποιεί» το συγκεκριμένο interface. Ορίζεται ως μέσο προτυποποίησης (standardization) του τρόπου επικοινωνίας διαφορετικών τύπων αντικειμένων. Αποτελεί μια κατευθυντήρια γραμμή (guideline) που υποβοηθά τον προγραμματιστή να αναπτύξη μια κλάση. Επειδή οι μέθοδοι ενός interface είναι πάντα public, θα πρέπει να δηλωθούν ως public όταν υλοποιηθούν μέσα σε μία κλάση. Τα interface έχουν σταθερές μεταβλητές και μεθόδους χωρίς σώμα. Ένα interface μπορεί να κάνει extends άλλα (ένα ή περισσότερα) interfaces και να κληρονομήσει τις μεθόδους του. Μια κλάση μπορεί να κάνει implements πολλά interfaces (αλλά extends μόνο μία κλάση). Κάθε κλάση που κάνει implements ένα interface είναι αναγκασμένη να υλοποιήσει όλες τις μεθόδους του.



Εικόνα 15: Interface

Διαφορές μεταξύ Interface και Abstract Class

- Οι αφηρημένες κλάσεις μπορούν να περιέχουν και fields που δεν είναι σταθερές (static – final) ενώ τα Interface όχι
- Οι αφηρημένες κλάσεις μπορούν να περιέχουν υλοποιημένες μεθόδους ενώ τα Interface δεν μπορούν. Αν μια αφηρημένη κλάση περιέχει μόνο abstract methods θα έπρεπε να δηλωθεί σαν Interface
- Οι αφηρημένες κλάσεις περιέχουν μέρος της υλοποίησης και αφήνουν στις υποκλάσεις την υπόλοιπη. Τα Interfaces δεν έχουν καθόλου υλοποίηση παρά μόνο δηλώσεις
- Τα Interfaces υλοποιούνται οπουδήποτε στην ιεραρχία κλάσεων ενώ οι abstract classes όχι

Μια αφηρημένη κλάση χωρίζεται συνήθως σε πολλές παρόμοιες υποκλάσειςπου έχουν πολλά κοινά στοιχεία (υλοποιημένες μέθοδοι) αλλά και κάποιες διαφορές (αφηρημένες μέθοδοι)

2.1.3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ JAVA

Η εταιρεία Sun περιγράφει τη Java ως «μια απλή, αντικειμενοστραφή, κατανεμημένη, ερμηνευόμενη, συμπαγή, ασφαλή, ανεξάρτητη αρχιτεκτονικής, μεταφέρσιμη, υψηλής απόδοσης, υποστηρίζουσα πολλαπλά νήματα, και δυναμική γλώσσα».

ΑΠΛΗ: Η Java είναι μια απλή (simple) γλώσσα, με την έννοια ότι η εκμάθησή της είναι σχετικά εύκολη. Περιέχει λίγες προγραμματιστικές δομές με καλά ορισμένη σημασιολογία.

ΑΝΤΙΚΕΙΜΕΝΙΟΣΤΡΑΦΗΣ: Ως αντικειμενοστραφής (object-oriented) γλώσσα, η Java εστιάζει τη δραστηριότητα του προγραμματιστή στον ορισμό αντικειμένων και λειτουργιών πάνω σε αυτά.

ΚΑΤΑΝΕΜΗΜΕΝΗ: Η Java χαρακτηρίζεται ως κατανεμημένη (distributed) γλώσσα προγραμματισμού γιατί υποστηρίζει την ανάπτυξη κατανεμημένων εφαρμογών δικτύων. Συγκεκριμένα, η Java επιτρέπει την προσπέλαση αντικειμένων που βρίσκονται σε απομακρυσμένες θέσεις στο δίκτυο, καθώς και τη δικτυακή επικοινωνία με άλλες εφαρμογές.

ΕΡΜΗΝΕΥΟΜΕΝΗ: Ο μεταγλωττιστής δεν παράγει τελικό κώδικα, για κάποιο συγκεκριμένο υπολογιστή, αλλά ενδιάμεσο κώδικα σε μορφή bytes, που ονομάζεται bytecode. Ο ενδιάμεσος κώδικας στη συνέχεια εκτελείται από ένα διερμηνέα της Java. Το πλεονέκτημα είναι ότι ο κώδικας μπορεί να εκτελεστεί σε πολλά διαφορετικά περιβάλλοντα υπολογιστών, αν φυσικά κάποιος διερμηνέας Java είναι διαθέσιμος σε αυτά.

ΣΥΜΠΑΓΗΣ: Η γλώσσα έχει ένα ισχυρό σύστημα τύπων, το οποίο επιτρέπει εκτενείς ελέγχους κατά τη διάρκεια της μετάφρασης των προγραμμάτων, βοηθώντας έτσι την ανάπτυξη αξιόπιστου λογισμικού.

ΑΣΦΑΛΗΣ: Προγράμματα τα οποία έχουν αναπτυχθεί με αυτήν μπορούν να εκτελεστούν από πολλούς χρήστες με διαφορετικά δικαιώματα πρόσβασης στο σύστημα, χωρίς να είναι δυνατό να υπάρξουν ανεπιθύμητες παρενέργειες ακούσιες ή εκούσιες.

ΑΝΕΞΑΡΤΗΤΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ: Τα προγράμματα Java μεταφράζονται σε ενδιάμεσο κώδικα, ο οποίος δεν είναι προσανατολισμένος σε μια συγκεκριμένη αρχιτεκτονική ή τύπο υπολογιστή.

ΜΕΤΑΦΕΡΣΙΜΗ: Το γεγονός ότι η Java είναι ανεξάρτητη αρχιτεκτονικής καθιστά τα προγράμματα Java μεταφέρσιμα. Οι προδιαγραφές της Java καθορίζουν σαφώς τα μεγέθη των διαφόρων πρωταρχικών τύπων δεδομένων καθώς και τα αποτελέσματα των διαφόρων λειτουργιών πάνω σε μεταβλητές αυτών των τύπων. Κατά συνέπεια, ένα πρόγραμμα Java θα δώσει τα ίδια αποτελέσματα όταν δεχθεί τα ίδια δεδομένα, ανεξάρτητα της αρχιτεκτονικής στην οποία θα εκτελεστεί.

ΥΨΗΛΗΣ ΑΠΟΔΟΣΗΣ: Η Java σαν ερμηνευόμενη γλώσσα δεν μπορεί να φτάσει την απόδοση γλωσσών προγραμματισμού όπως η C και η C++, που υλοποιούνται σε μεταγλωττιστές. Παρ'όλα αυτά, η απόδοση της γλώσσας είναι αρκετά υψηλή. Κατά μέσο όρο η εκτέλεση προγραμμάτων Java είναι περίπου 20% αργότερη από την εκτέλεση των αντίστοιχων προγραμμάτων σε C ή C++.

ΥΠΟΣΤΗΡΙΖΟΥΣΑ ΠΟΛΛΑΠΛΑ ΝΗΜΑΤΑ: Τα πολλαπλά νήματα εκτέλεσης αποτελούν ένα προγραμματιστικό μοντέλο για ανάπτυξη λογισμικού, που απαιτεί την 'ταυτόχρονη' εκτέλεση πολλών διεργασιών.

ΔΥΝΑΜΙΚΗ: Η Java είναι μια δυναμική (dynamic) γλώσσα, η οποία έχει σχεδιαστεί για προσαρμογή σε ένα δυναμικά εξελισσόμενο περιβάλλον.

ΥΠΟΣΤΗΡΙΞΗ ΠΟΛΥΜΕΣΩΝ: Η Java καθιστά δυνατή τη μεταφορά εκτελέσιμου περιεχομένου σε εφαρμογές πολυμέσων.

2.1.4 ΠΛΕΟΝΕΚΤΗΜΑΤΑ & ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΤΗΣ JAVA

ΠΛΕΟΝΕΚΤΗΜΑΤΑ:

1) Αντικειμενοστραφής.

2) Κατάλληλη για τον Παγκόσμιο Ιστό (μικρή, ασφαλής, μεταφερτή).

3) Μαθαίνεται εύκολα.

4) Γρήγορη ανάπτυξη προγραμμάτων.

5) Διανέμεται δωρεάν.

ΜΕΙΟΝΕΚΤΗΜΑΤΑ:

1) Όχι πολύ γρήγορη.

2) Χρειάζεται εγκατάσταση μεταφραστή Java (runtime) πριν τρέξει οτιδήποτε.

2.1.5 ΕΙΚΟΝΙΚΗ ΜΗΧΑΝΗ ΤΗΣ JAVA

Αφού γραφεί κάποιο πρόγραμμα σε Java, στη συνέχεια μεταγλωττίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode). Ο κώδικας byte είναι η μορφή που παίρνει ο πηγαίος κώδικας της Java όταν μεταγλωττιστεί. Όταν πρόκειται να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, το Java Virtual Machine που πρέπει να είναι εγκατεστημένο σε αυτό θα αναλάβει να διαβάσει τα αρχεία .class. Στη συνέχεια τα μεταφράζει σε γλώσσα μηχανής που να υποστηρίζεται από το λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να εκτελεστεί (να σημειωθεί εδώ ότι αυτό συμβαίνει με την παραδοσιακή Εικονική Μηχανή (Virtual Machine).

Πιο σύγχρονες εφαρμογές της εικονικής Μηχανής μπορούν και μεταγλωττίζουν εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (εγγενή κώδικα ή native code) με αποτέλεσμα να βελτιώνεται η ταχύτητα). Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java. Πρέπει να σημειωθεί ότι η JVM είναι λογισμικό που εξαρτάται από την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές κλπ.

Οτιδήποτε θέλει να κάνει ο προγραμματιστής (ή ο χρήστης) γίνεται μέσω της εικονικής μηχανής. Αυτό βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη - υπολογιστή. Ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Από την άλλη μεριά ούτε ο χρήστης μπορεί να κατεβάσει «κακό» κώδικα από το δίκτυο και να τον εκτελέσει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα όπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως.

Ακόμα μία ιδέα που βρίσκεται πίσω από τη *Java* είναι η ύπαρξη του συλλέκτη απορριμμάτων (*Garbage Collector*). Συλλογή απορριμμάτων είναι μία κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο. Αυτή η απελευθέρωση μνήμης στη Java είναι αυτόματη και γίνεται μέσω του συλλέκτη απορριμμάτων. Υπεύθυνη για αυτό είναι και πάλι η εικονική μηχανή η οποία μόλις «καταλάβει» ότι ο σωρός (heap) της μνήμης (στη Java η συντριπτική πλειοψηφία των αντικειμένων αποθηκεύονται στο σωρό σε αντίθεση με τη C++ όπου αποθηκεύονται

κυρίως στη στοίβα) κοντεύει να γεμίσει ενεργοποιεί το συλλέκτη απορριμμάτων. Έτσι ο προγραμματιστής δε χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο

τμήμα της μνήμης, ούτε και για σφάλματα δεικτών. Αυτό είναι ιδιαίτερα σημαντικό γιατί είναι κοινά τα σφάλματα προγραμμάτων που οφείλονται σε λανθασμένο χειρισμό της μνήμης.

2.1.6 SWING

Το Swing είναι μια εργαλειοθήκη της Java που περιλαμβάνει όλα τα απαραίτητα συστατικά ώστε να δημιουργήσουμε διεπαφές με γραφικά και να προσθέσουμε διαδραστικότητα σε Java εφαρμογές.

Για να κάνουμε σωστή και πλήρη χρήση των δυνατοτήτων του Swing θα πρέπει να έχουμε ανοικτό το API Specification της Java και να βλέπουμε τι προσφέρεται στα πακέτα:

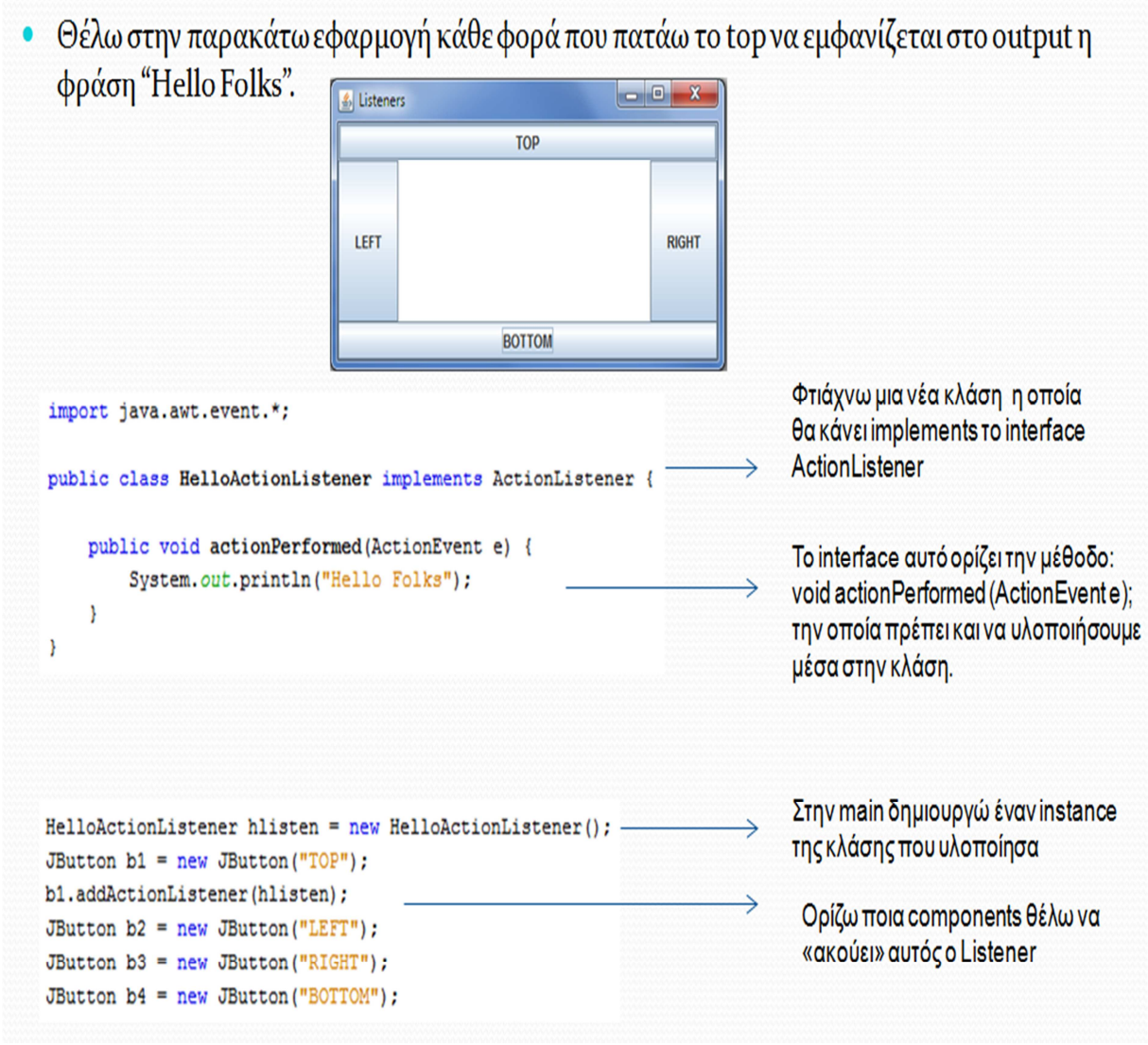
- java.awt → "Contains all of the classes for creating user interfaces and for painting graphics and images."
- javax.swing → "Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms."

```
import javax.swing.*;
import java.awt.*;
public class Main {
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setTitle("Hello World!");
        frame.setSize(400, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);

        frame.setLayout(new BorderLayout());

        JButton b1 = new JButton("TOP");
        JButton b2 = new JButton("LEFT");
        JButton b3 = new JButton("RIGHT");
        JButton b4 = new JButton("BOTTOM");

        JLabel label = new JLabel("Hello Folks! I'm at the center!");

        frame.add(b1,BorderLayout.PAGE_START);
        frame.add(b2,BorderLayout.LINE_START);
        frame.add(b3,BorderLayout.LINE_END);
        frame.add(b4,BorderLayout.PAGE_END);
        frame.add(label,BorderLayout.CENTER);
    }
}
```

Εικόνα 16: Παράδειγμα Swing

2.1.7 LISTENERS & EVENTS

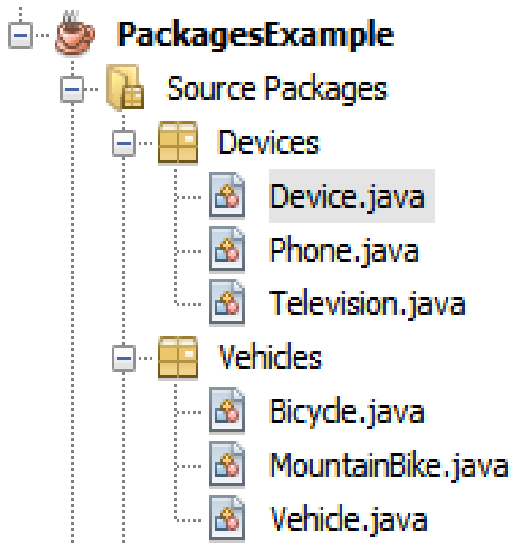
Κάθε φορά που αλληλεπιδράει ο χρήστης με ένα component δημιουργήται ένα event object το οποίο περιέχει πληροφορίες για το συγκεκριμένο event και προσδιορίζει ποιά ήταν η πηγή του. Π.χ. Σε μία εφαρμογή υπάρχει ένα JButton με κείμενο «Α». Πατώντας το κουμπί Α δημιουργήται ένα action event object του οποίου η πηγή είναι το κουμπί Α. Για να διαχειριστούμε αυτά τα events (event handling) χρησιμοποιούμε τους event Listeners στους οποίους ορίζουμε τι θα γίνεται κάθε φορά που θα εμφανίζεται ένα event. Μπορούν να δημιουργηθούν πολλοί listeners για να διαχειριστούν συγκεκριμένου τύπου events από μια συγκεκριμένη πηγή. Επίσης ο ίδιος listener μπορεί να χρησιμοποιηθεί για να διαχειριστεί events από διαφορετικά objects.



Εικόνα 17: Παράδειγμα με Listeners

2.1.8 ΠΑΚΕΤΑ - PACKAGES

Είναι μια δομή που μας βοηθάει να ομαδοποιήσουμε κλάσεις και interfaces. Κάτι σαν τους φάκελους στον υπολογιστή μας. Το Java Platform API Specification μας δίνει μια λίστα από όλα τα υλοποιημένα packages, interfaces, classes, fields, και methods που μας παρέχονται έτοιμα από την Java.



Εικόνα 18: Παράδειγμα ομαδοποίησης κλάσεων

2.2 ΕΙΣΑΓΩΓΗ ΣΤΗ UNIFIED MODELING LANGUAGE (UML)

Η Unified Modeling Language (UML) είναι μία γλώσσα που χρησιμοποιείται για προδιαγραφές, αναπαράσταση με οπτικό τρόπο (visualizing), δημιουργία και τεκμηρίωση των τμημάτων των συστημάτων λογισμικού, καθώς και για μοντελοποίηση εταιρικών και άλλων συστημάτων που δεν αφορούν λογισμικό. Η UML αποτελεί ένα συνδυασμό των καλύτερων πρακτικών, οι οποίες ήδη έχουν αποδείξει πόσο επιτυχημένες ήταν στη μοντελοποίηση μεγάλων και σύνθετων συστημάτων.

2.2.1 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ ΤΗΣ UML

Η UML έχει αναπτυχθεί από την Rational Software και τους εταίρους της. Επίσης πολλές εταιρίες έχουν ενσωματώσει τη UML ως πρότυπο στη διαδικασία ανάπτυξής τους και στα προϊόντα τους, τα οποία καλύπτουν περιοχές όπως η business modeling, η διαχείριση απαιτήσεων, η ανάλυση και ο σχεδιασμός, ο προγραμματισμός και ο έλεγχος.

Από τα μέσα της δεκαετίας του 70 έως τα τέλη της δεκαετίας του 80 άρχισαν να εμφανίζονται αντικειμενοστραφείς γλώσσες μοντελοποίησης, καθώς και οι διάφοροι ερευνητές είχαν αρχίσει να πειραματίζονται με διαφορετικές προσεγγίσεις στην αντικειμενοστραφή ανάλυση και σχεδιασμό. Οι γλώσσες αυτές επηρεάστηκαν από άλλες τεχνικές, όπως η μοντελοποίηση Οντοτήτων-Συσχετίσεων, η γλώσσα προδιαγραφών και περιγραφής (Specification & Description Language - SDL, περίπου 1976, CCITT) και άλλες τεχνικές. Το πλήθος των γλωσσών μοντελοποίησης αυξήθηκε από λιγότερες από 10 σε πάνω από 50 στην περίοδο 1989-1994. Πολλοί από τους χρήστες των αντικειμενοστραφών μεθόδων δεν ικανοποιούντο πλήρως από μία από αυτές τις γλώσσες μοντελοποίησης δημιουργώντας «πολέμους μεθοδολογίας». Στα μέσα της δεκαετίας του 90, είχαν αρχίσει να εμφανίζονται νέες εκδόσεις των μεθοδολογιών αυτών, όπως η Booch 93, η συνεχιζόμενη εξέλιξη της OMT και η Fusion.

Οι μεθοδολογίες αυτές άρχισαν να ενσωματώνουν η μία τις τεχνικές της άλλης και άρχισαν να εμφανίζονται μεθοδολογίες που ήταν αρκετά υποσχόμενες όπως οι OOSE, OMT-2 και Booch 93. Κάθε μία από αυτές ήταν μία πλήρης μεθοδολογία και είχε τα δικά της ισχυρά σημεία. Με απλά λόγια, η OOSE ήταν μία προσέγγιση που βασιζόταν σε περιπτώσεις χρήσης και προσέφερε άριστη υποστήριξη για business engineering και ανάλυση απαιτήσεων. Η OMT-2 ήταν ιδιαίτερα εκφραστική για ανάλυση και πληροφοριακά συστήματα με έμφαση στα δεδομένα. Η Booch 93 ήταν ιδιαίτερα εκφραστική κατά τις φάσεις του σχεδιασμού και της υλοποίησης για έργα, ενώ ήταν διάσημη για εφαρμογές που είχαν έμφαση στην τεχνολογία.

Η ανάπτυξη της UML ξεκίνησε τον Οκτώβριο του 1994 όταν οι Graddy Booch και Jim Rumbaugh της Rational Software Corporation άρχισαν να δουλεύουν για την ενοποίηση των μεθοδολογιών Booch και OMT (Object Modeling Technique). Δεδομένου του ότι οι δύο μεθοδολογίες ήδη αναπτύσσονταν ξεχωριστά και αναγνωρίζονταν ήδη ως οι κυρίαρχες αντικειμενοστραφείς μεθοδολογίες παγκόσμια, οι Booch και Rumbaugh ένωσαν τις δυνάμεις τους για να επιτευχθεί μια πλήρης ενοποίηση των μεθοδολογιών τους. Το πρώτο σχέδιο της έκδοσης 0.8 της Unified Method, όπως ονομαζόταν τότε, εμφανίστηκε τον Οκτώβριο του 1995. Στο φθινόπωρο του 1995 ο Ivar Jacobson και η εταιρία του Objectory ενώθηκε με την Rational και την προσπάθεια ενοποίησης, συνδυάζοντας και τη μεθοδολογία OOSE (Object-Oriented Software Engineering). Το όνομα της Objectory χρησιμοποιείται πλέον στη Rational κυρίως για να δηλωθεί η συμβατή με τη UML διαδικασία που παρέχει, η Rational Unified Process.

Ως βασικοί συγγραφείς των μεθοδολογιών οι Booch OMT και OOSE, οι Grady Booch, Jim Rumbaugh και Ivar Jacobson είχαν το κίνητρο για να δημιουργήσουν μία ενοποιημένη γλώσσα μοντελοποίησης για τρείς λόγους. Κατ' αρχάς οι μέθοδοι ήδη εξελίσσονταν προς τις άλλες ανεξάρτητα. Είχε νόημα να συνεχίσουν την προσπάθεια εξέλιξης μαζί, αντί να το κάνουν ο καθένας ξεχωριστά, απαλείφοντας έτσι την πιθανότητα για διαφορές που δε θα είχαν νόημα και απλά θα μπέρδευαν τους χρήστες. Δεύτερον, ενοποιώντας τη σημασιολογία και το συμβολισμό θα υπήρχε μία σταθερότητα στην αντικειμενοστραφή αγορά, που θα επέτρεπε στα έργα να χρησιμοποιήσουν μία ώριμη γλώσσα μοντελοποίησης και να αφήσουν όσους θα έφτιαχναν τα εργαλεία να δώσουν βάρος στο πώς θα παρέχουν τα χρήσιμα χαρακτηριστικά. Τρίτον, περίμεναν ότι η συνεργασία τους θα προσέφερε βελτιώσεις και στις τρεις προηγούμενες μεθόδους, βοηθώντας τους να συλλάβουν τα μαθήματα που είχαν ήδη πάρει και να αντιμετωπίσουν προβλήματα που δεν μπορούσε να χειριστεί ορθά καμία από τις προηγούμενες μεθοδολογίες.

Καθώς ξεκίνησαν τη διαδικασία ενοποίησης, έβαλαν τέσσερις στόχους για να εστιάσουν τις προσπάθειες τους:

1. Να διευκολύνουν τη μοντελοποίηση των συστημάτων (όχι του λογισμικού) χρησιμοποιώντας αντικειμενοστραφείς έννοιες.
2. Να δημιουργήσουν μία ρητή σύζευξη προς τα σημασιολογικά αλλά και τα εκτελέσιμα στοιχεία.
3. Να αντιμετωπίσουν ζητήματα κλίμακας που είναι έμφυτα στα σύνθετα και κρίσιμα συστήματα.
4. Να δημιουργήσουν μία γλώσσα μοντελοποίησης που θα μπορούν να χρησιμοποιήσουν τόσο άνθρωποι όσο και μηχανές.

Οι προσπάθειες των Booch, Rumbaugh και Jacobson είχαν ως αποτέλεσμα την εμφάνιση της τεκμηρίωσης της UML 0.9 και 0.91 τον Ιούνιο και τον Οκτώβριο του 1996. Το 1996 οι συγγραφείς της UML ζήτησαν και έλαβαν σχόλια από την κοινότητα. Ενσωμάτωσαν τα σχόλια αλλά ήταν σαφές ότι ήταν απαραίτητη επιπλέον εστιασμένη προσοχή.

Το 1996 έγινε σαφές ότι αρκετοί οργανισμοί έβλεπαν τη UML ως στρατηγική επιλογή για τις εταιρίες τους. Μία αίτηση για προτάσεις (Request for Proposal -RFP) από το Object Management Group (OMG) ήταν ο καταλύτης ώστε οι οργανισμοί αυτοί να ενώσουν τις δυνάμεις τους και να παρουσιάσουν μια κοινή απάντηση στο RFP. Η Rational δημιούργησε την κοινοπραξία των εταίρων της UML με αρκετούς οργανισμούς που είχαν τη διάθεση να διαθέσουν τους πόρους για να δημιουργηθεί ένας ισχυρός ορισμός της UML. Εκείνοι που συμμετείχαν περισσότερο στον ορισμό της

UML περιλαμβάνουν τους: Digital Equipment Corp., HP, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI και Unisys. Η συνεργασία αυτή παρήγαγε τη UML, μία γλώσσα μοντελοποίησης που ήταν καλά ορισμένη, εκφραστική, ισχυρή και γενικά εφαρμόσιμη.

Τον Ιανουάριο του 1997 η IBM με την ObjecTime, η Platinum Technology, η Ptech, οι Taskon και Reich Technologies και τέλος η SofTeam υπέβαλλαν με τη σειρά τους απαντήσεις στο RFP του OMG. Οι εταιρίες αυτές ενώθηκαν με τους εταίρους της UML για να υποβάλλουν τις ιδέες και το αποτέλεσμα ήταν η UML 1.1. Η UML 1.1 εστίασε στο να ξεκαθαριστεί η σημασιολογία της UML 1.0 και να ενσωματωθούν οι συνεισφορές των νέων εταίρων.

2.2.2 ΚΥΡΙΑ ΣΤΟΙΧΕΙΑ ΤΗΣ UML

Η επιλογή σχετικά με το ποιά μοντέλα και ποιά διαγράμματα θα δημιουργηθούν επηρεάζει σημαντικά τον τρόπο προσέγγισης και επίλυσης του προβλήματος. Το βασικό στοιχείο στην προσπάθεια επικοινωνίας και μάθησης είναι η έννοια της *αφαίρεσης*, να εστιάζουμε δηλαδή σε σχετικές λεπτομέρειες, ενώ αγνοούμε τις υπόλοιπες. Συνέπεια αυτού είναι τα ακόλουθα:

1. Κάθε σύνθετο σύστημα προσεγγίζεται καλύτερα χρησιμοποιώντας ένα μικρό σύνολο ανεξαρτήτων όψεων του συστήματος. Δεν αρκεί μία όψη.
2. Κάθε μοντέλο μπορεί να εκφραστεί σε διαφορετικά επίπεδα πιστότητας.
3. Τα καλύτερα μοντέλα συνδέονται με την πραγματικότητα.

Στην UML υπάρχουν 13 τύποι διαγραμμάτων, στοιχείο που υποδηλώνει την ανάπτυξη της έκδοσης 2.0 σε σχέση με την 1.5, αφού στην τελευταία λιγότερα διαγράμματα ήταν διαθέσιμα. Για την καλύτερη κατανόησή τους κρίνεται απαραίτητη η κατηγοριοποίησή τους σε τρεις διαφορετικές κατηγορίες, οι οποίες παρουσιάζονται παρακάτω:

Διαγράμματα Δομής

Τα διαγράμματα δομής τονίζουν ποια είναι τα δομικά συστατικά του συστήματος που μοντελοποιείται:

- Διαγράμματα κλάσεων
- Διαγράμματα συστατικών
- Διαγράμματα αντικειμένων
- Διαγράμματα πολύπλοκης δομής
- Διαγράμματα ανάπτυξης
- Διαγράμματα συσκευασίας

Διαγράμματα Συμπεριφοράς

Τα διαγράμματα συμπεριφοράς τονίζουν τι πρέπει να γίνει στο εσωτερικό του μοντέλου που μοντελοποιείται:

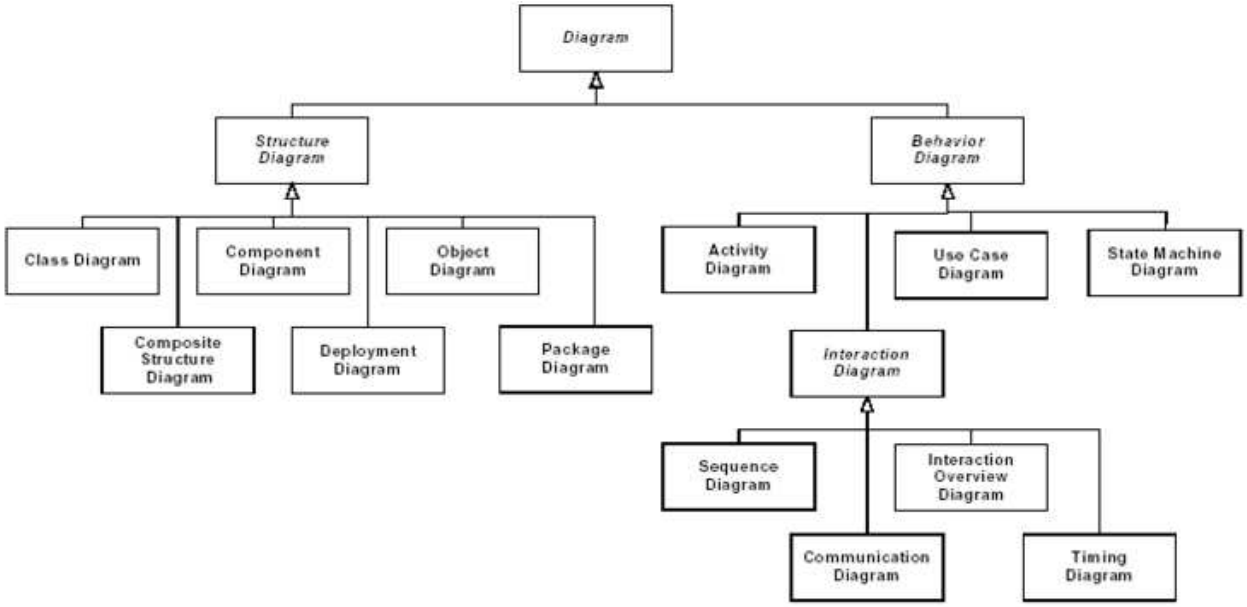
- Διαγράμματα δραστηριοτήτων
- Διαγράμματα use case
- Διαγράμματα κατάστασης μηχανής

Διαγράμματα Αλληλεπίδρασης

Τα διαγράμματα αυτά αποτελούν υποκατηγορία των διαγραμμάτων συμπεριφοράς, και τονίζουν την ροή ελέγχου και δεδομένων που υπάρχει ανάμεσα στα αντικείμενα του συστήματος, που μοντελοποιείται:

- Διαγράμματα αλληλουχίας
- Διαγράμματα επικοινωνίας (UML 2.0)
- Διαγράμματα εποπτείας συναλλαγών (UML 2.0)

- Διαγράμματα χρονισμού (UML 2.0)



Εικόνα 19: Η ιεραρχία των διαγραμμάτων UML 2.0 παρουσιαζόμενη σαν διάγραμμα κλάσεων

Η γλώσσα UML δεν περιορίζει τους τύπους των στοιχείων σε ένα συγκεκριμένο τύπο διαγραμμάτων UML. Γενικά, κάθε στοιχείο UML μπορεί να παρουσιάζεται σε όλους τους τύπους διαγραμμάτων. Αυτή η δυνατότητα έχει περιοριστεί στην έκδοση UML 2.0. Τα διαγράμματα μπορούν να ανταλλαχθούν ανάμεσα στα εργαλεία UML χρησιμοποιώντας το πρότυπο XML Metadata Interchange (XMI). Τα διαγράμματα UML συνοδεύονται πάντα από ένα σχόλιο ή σημείωμα, το οποίο περιγράφει την χρήση, τους περιορισμούς και τον σκοπό του μοντέλου UML.

Τα διαγράμματα αυτά παρέχουν διαφορετικές απόψεις του συστήματος κατά τη φάση της ανάλυσης και της ανάπτυξης. Το υποκείμενο μοντέλο ολοκληρώνει τις απόψεις αυτές, ούτως ώστε να μπορεί να αναλυθεί και να δομηθεί ένα πλήρες, συνεπές με τον εαυτό του σύστημα. Τα διαγράμματα αυτά, καθώς και η σχετική τεκμηρίωση είναι τα βασικά στοιχεία που βλέπει αυτός που μοντελοποιεί, αν και η UML και τα εργαλεία που την υποστηρίζουν θα παρέχουν και ένα πλήθος παραγόμενων όψεων.

Τα περισσότερα από τα διαγράμματα της UML και ορισμένα από τα σύνθετα σύμβολά της είναι γράφοι με κορυφές οι οποίες ενώνονται μέσω ακμών. Η πληροφορία βρίσκεται περισσότερο στην τοπολογία, παρά στο μέγεθος ή στη θέση των συμβόλων (υπάρχουν μερικές εξαιρέσεις όπως τα διαγράμματα ακολουθίας με άξονα του χρόνου). Υπάρχουν τρία είδη οπτικής σχέσης που μας ενδιαφέρουν:

1. σύνδεση (συνήθως σε γραμμές σε δυσδιάστατα σχήματα),
2. ενσωμάτωση (containment) (συμβόλων από δυσδιάστατα σχήματα που έχουν όρια), και
3. οπτική προσκόλληση (visual attachment) (ένα μεγάλο σύμβολο είναι κοντά σε κάποιο άλλο σε ένα διάγραμμα).

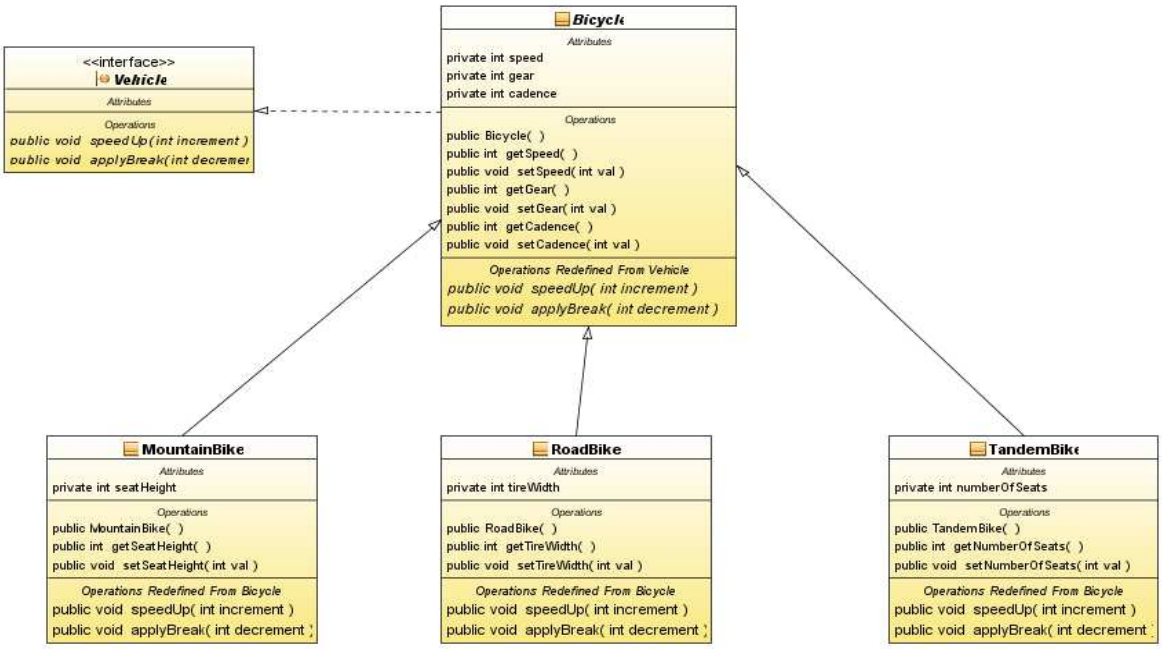
Αυτές οι οπτικές σχέσεις απεικονίζονται σε συνδέσεις κόμβων σε ένα γράφο.

Ο συμβολισμός της UML έχει φτιαχτεί για σχέδια σε δύο διαστάσεις. Ορισμένα από τα σχέδια είναι δυσδιάστατες προβολές τρισδιάστατων σχεδίων (όπως κύβοι), αλλά συνεχίζουν να αναπαριστώνται ως εικονίδια σε δυσδιάστατες επιφάνειες.

Υπάρχουν τέσσερα βασικά είδη γραφικών δομών που χρησιμοποιούνται στο συμβολισμό της UML:

1. Εικονίδια - Ένα εικονίδιο είναι ένα γραφικό σχήμα συγκεκριμένου μεγέθους και μορφής. Δεν επεκτείνεται για να χωρέσει τα περιεχόμενά του. Ένα εικονίδιο μπορεί να εμφανιστεί μέσα σε σύμβολα περιοχής (area symbols), όπως τερματισμούς (terminators), σε ακμές ή ως αυτοδύναμο σύμβολο που μπορεί να συνδέεται ή όχι με ακμές.
2. Δυσδιάστατα σύμβολα - Τα δυσδιάστατα σύμβολα έχουν μεταβλητές διαστάσεις, ώστε να μπορούν να χωράνε άλλα στοιχεία, όπως λίστες συμβολοσειρών ή άλλα σύμβολα. Πολλά από αυτά διαιρούνται σε παρόμοια ή διαφορετικά τμήματα (compartments). Οι ακμές συνδέονται με τα δυσδιάστατα σύμβολα τερματίζοντας την ακμή στο όριο του συμβόλου. Η μεταφορά ή διαγραφή ενός δυσδιάστατου συμβόλου επηρεάζει τα περιεχόμενα του συμβόλου και τις ακμές που συνδέονται με αυτό.
3. Ακμές - Είναι ακολουθίες τμημάτων γραμμών των οποίων τα τελικά σημεία είναι προσκολλημένα. Εννοιολογικά μία ακμή είναι μία τοπολογική οντότητα, αν και μπορούμε να χειριστούμε γραφικά τα τμήματα. Ένα τμήμα δεν μπορεί να υπάρξει ανεξάρτητα από την ακμή του. Οι ακμές είναι πάντα προσκολλημένες σε άλλα γραφικά σύμβολα και στα δύο άκρα (δεν υπάρχουν dangling γραμμές). Οι ακμές μπορούν να έχουν τερματισμούς (terminators), δηλαδή, εικονίδια που εμφανίζονται με κάποια σειρά στο τέλος της ακμής και προσδιορίζουν την έννοια του συμβόλου της ακμής.
4. Συμβολοσειρές - Αναπαριστούν συγκεκριμένο είδος πληροφορίας σε μία μη επεξεργασμένη (unparsed) μορφή, η οποία υποθέτει ότι κάθε χρήση μιας συμβολοσειράς στο συμβολισμό έχει συγκεκριμένη σύνταξη η οποία μπορεί να επεξεργαστεί συντακτικά (parsed) στην πληροφορία του υποκείμενου μοντέλου. Για παράδειγμα, η σύνταξη για τα ιδιοχαρακτηριστικά (attributes), τις πράξεις (operations) και τις μεταβάσεις (transitions). Αυτοί οι τρόποι σύνταξης μπορεί να διαφέρουν στα εργαλεία, ως επιλογές παρουσίασης. Οι συμβολοσειρές υπάρχουν ως αυτόνομα στοιχεία των συμβόλων ή των τμημάτων (compartements) των συμβόλων, ως στοιχεία σε λίστες (στην οποία περίπτωση η θέση τους στη λίστα δηλώνει κάποια πληροφορία), ως ετικέτες οι οποίες έχουν προσκολληθεί σε σύμβολα ή ακμές ή ως αυτοδύναμα στοιχεία σε ένα διάγραμμα.

Η γλώσσα UML είναι εφαρμόσιμη για την επίλυση αντικειμενοστραφών προβλημάτων. Όλα ξεκινούν, με την δημιουργία ενός μοντέλου. Το μοντέλο (model) αποτελεί ουσιαστικά την αφαίρεση του υπάρχοντος προβλήματος. Ο τομέας (domain) αποτελεί το πραγματικό περιβάλλον, από το οποίο προέρχεται το πρόβλημα. Το μοντέλο αποτελείται από τα αντικείμενα (objects), τα οποία επικοινωνούν μεταξύ τους ανταλλάσοντας μηνύματα. Τα αντικείμενα περιέχουν στοιχεία τα οποία τα χαρακτηρίζουν, που λέγονται χαρακτηριστικά (attributes) και ενέργειες που μπορούν να εκτελέσουν, που λέγονται συμπεριφορές και λειτουργίες (behaviors and operations). Το περιεχόμενο των χαρακτηριστικών του αντικειμένου καθορίζουν την κατάστασή τους. Οι κλάσεις (classes) αποτελούν τα προσχέδια των αντικειμένων, δηλαδή τα αντικείμενα είναι απλά στιγμιότυπα των κλάσεων. Οι κλάσεις ενσωματώνουν τα χαρακτηριστικά (δεδομένα) και τις συμπεριφορές (μεθόδους και συναρτήσεις) σε μία μοναδική ενότητα.



Εικόνα 20: Παράδειγμα UML

2.2.3 ΣΥΓΚΡΙΣΗ ΤΗΣ UML ΚΑΙ ΑΛΛΩΝ ΓΛΩΣΣΩΝ ΜΟΝΤΕΛΟΠΟΙΗΣΗΣ

Η UML είναι πιο εκφραστική, αλλά και πιο ξεκάθαρη και ενοποιημένη από μεθοδολογίες όπως η Booch, OMT και OOSE. Αυτό σημαίνει ότι όταν μεταφερόμαστε στη UML κερδίζουμε γιατί μπορούμε να μοντελοποιήσουμε έργα που δεν μπορούσαμε πριν. Επίσης υπάρχει κέρδος γιατί αφαιρούνται οι άχρηστες διαφορές στο συμβολισμό και την ορολογία που δεν αφήνουν να φανούν οι ομοιότητες των προσεγγίσεων αυτών.

Σε σχέση με άλλες οπτικές γλώσσες μοντελοποίησης, συμπεριλαμβανομένων της μοντελοποίησης οντοτήτων-συσχετίσεων, της Business Reengineering Process (BRP), των διαγραμμάτων ροής και των γλωσσών που προσανατολίζονται στις καταστάσεις, η UML προσφέρει επιπλέον εκφραστικότητα καθώς και ολιστική ακεραιότητα.

2.3 ΕΙΣΑΓΩΓΗ ΣΤΗΝ OCR

Η **Οπτική Αναγνώριση Χαρακτήρων** (Αγγλ. **O**ptical **C**haracter **R**ecognition) ή αλλιώς Αυτόματη Αναγνώριση Χαρακτήρων Κειμένου ονομάζεται η διαδικασία μετατροπής σαρωμένων εικόνων χειρογράφων ή έντυπων κειμένων σε κείμενο αναγνώσιμο από ηλεκτρονικό υπολογιστή. Η Οπτική Αναγνώριση Χαρακτήρων καθιστά εφικτή την εκ νέου επεξεργασία του κειμένου, αποφεύγοντας την δακτυλογράφηση του από την αρχή. Τα συστήματα Οπτικής Αναγνώρισης Χαρακτήρων απαιτούν βαθμονόμηση για να διαβάσουν μια συγκεκριμένη γραμματοσειρά. Οι πρώτες εκδόσεις ήταν προγραμματισμένες με εικόνες για κάθε χαρακτήρα και δούλευαν μια γραμματοσειρά την φορά. Τα ευφυή συστήματα με υψηλό δείκτη αναγνώρισης είναι πλέον κοινά. Μερικά συστήματα είναι ικανά να αναπαράγουν ακόμη και τις πληροφορίες που δεν είναι κείμενο σε ένα έγγραφο, όπως εικόνες, στήλες, γραμμές, γωνίες κτλ.

2.3.1 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Το 1929, ο Gustav Tauschek απέκτησε ευρεσιτεχνία για το OCR στην Γερμανία, ακολουθούμενος από τον Paul W. Handel που απέκτησε την ευρεσιτεχνία για την OCR στις Ηνωμένες

Πολιτείες το 1933. Το 1935 ο Tauschek πήρε επίσης την ευρεσιτεχνία στην μέθοδο του στις ΗΠΑ. Το μηχάνημα του Tauschek ήταν μια μηχανική συσκευή που χρησιμοποιούσε πρότυπα και αισθητήρα φωτός. Το 1949, οι μηχανικοί της RCA δημιούργησαν τον πρώτο OCR σύστημα για να βοηθήσουν τους τυφλούς για το US Veterans Administration, αλλά αντί να μετατρέπουν εκτυπωμένους χαρακτήρες σε χαρακτήρες αναγνώσιμους από υπολογιστή, η συσκευή τους μετέτρεπε και τους διάβασε. Η συσκευή είχε υψηλό κόστος και δεν δόθηκε για παραγωγή. Το 1950, ο David H. Shepard, ένας κρυπταναλητής των Armed Forces Security Agency των ΗΠΑ δημιούργησε μια συσκευή που μετέτρεπε τα εκτυπωμένα μηνύματα σε κείμενο αναγνώσιμο από ηλεκτρονικό υπολογιστή αφού έκδωσε την δική του πατέντα. Έπειτα, ο Shepard ίδρυσε την Intelligent Machines Research Corporation (IMR), η οποία ήταν η πρώτη που έβαλε σε εμπορική λειτουργία τα συστήματα OCR.

Το 1955, το πρώτο εμπορικό σύστημα εγκαταστάθηκε στο Reader's Digest. Το δεύτερο σύστημα πουλήθηκε στην Standard Oil για να διαβάζει αριθμούς πιστωτικών καρτών για λογαριασμούς. Άλλα συστήματα που πουλήθηκαν από την IMR γύρω στο 1950s είχαν αναγνωστέα αποκόμματος λογαριασμού στην Ohio Bell Telephone Company και έναν σαρωτή σελίδας στις United States Air Force για ανάγνωση και μετάδοση χειρόγραφων μηνυμάτων από τον. Η IBM και άλλες αγόρασαν τις άδειες ευρεσιτεχνίας OCR του Shepard.

Το 1965, το Reader's Digest και η RCA συνεργάστηκαν για να φτιάξουν μια συσκευή OCR για να διαβάζει και να ψηφιοποιεί τους σειριακούς αριθμούς από τα κουπόνια του Reader's Digest από τις διαφημίσεις. Οι γραμματοσειρά που χρησιμοποιήθηκε για την εκτύπωση των κουπονιών ήταν η OCR-A font. Η συσκευή ήταν συνδεδεμένη σε ένα RCA 301 υπολογιστή. Η συσκευή επίσης είχε έναν ειδικό αναγνωστέα TWA. Η συσκευή μπορούσε να επεξεργαστεί 1,500 έγγραφα ανά λεπτό, απορρίπτοντας ότι δεν μπορεί να αναγνωρίσει σωστά.

Το Ταχυδρομείο των ΗΠΑ χρησιμοποιεί τεχνολογία οπτικής αναγνώρισης από το 1965 βασιζόμενο σε τεχνολογία που ανέπτυξε ο εφευρέτης Jacob Rabinow. Η πρώτη χρήση της Οπτικής Αναγνώρισης στην Ευρώπη έγινε από το Ταχυδρομείο της Αγγλίας. Το 1965 ξεκίνησε την κατασκευή ενός τραπεζικού συστήματος βασιζόμενο στην τεχνολογία OCR, μια διαδικασία που έφερε επανάσταση στα συστήματα πληρωμής λογαριασμών στην Μ. Βρετανία. Το ταχυδρομείο του Καναδά υιοθέτησε τα συστήματα OCR από το 1971.

Το 1974, ο Ray Kurzweil ίδρυσε την εταιρία Kurzweil Computer Products, Inc. και δημιούργησε το πρώτο σύστημα οπτικής αναγνώρισης χαρακτήρων που αναγνώριζε εκτυπωμένο κείμενο διαφόρων γραμματοσειρών. Η εταιρία εστίασε στην δημιουργία μιας συσκευής που θα βοηθήσει τους τυφλούς να διαβάζουν κείμενο με βοήθεια υπολογιστή. Η συσκευή απαιτούσε την εφεύρεση δύο τεχνολογιών – μια συσκευή σάρωσης και ένα σύστημα ανάγνωσης κειμένου από τον υπολογιστή.

Το 1978, η εταιρία Kurzweil Computer Products άρχισε να πουλά εταιρικές εκδόσεις του λογισμικού οπτικής αναγνώρισης. Η LexisNexis ήταν από τους πρώτους πελάτες που αγόρασαν το λογισμικό για να μεταφορτώνουν έγγραφα στην online βάση δεδομένων τους. Δύο χρόνια μετά, ο Kurzweil πούλησε την εταιρία στην Xerox, που έδειξε ενδιαφέρον για την επέκταση της τεχνολογίας οπτικής αναγνώρισης.

2.3.2 ΛΕΙΤΟΥΡΓΙΑ ΤΗΣ OCR

Υπάρχουν δύο κύριοι τρόποι εφαρμογής της Οπτικής Αναγνώρισης, η "Αντιστοίχηση με Πρότυπα" και η "Εξαγωγή Χαρακτηριστικών". Η πρώτη μέθοδος είναι πιο διαδεδομένη και κοινή αλλά περιορίζεται αρκετά σε σχέση με την 2η τεχνική. Η σημερινή τεχνολογία χρησιμοποιεί τον συνδυασμό και των δύο τεχνολογιών για την καλύτερη επίτευξη αποτελεσμάτων, κυρίως σε χειρόγραφα έγγραφα.

Αντιστοίχηση με πρότυπα: Η αντιστοίχηση με πρότυπα αφορά την αναγνώριση χαρακτήρων από έτοιμα πρότυπα ή περιγράμματα χαρακτήρων. Ο σαρωτής ψηφιοποιεί την εικόνα ενός εγγράφου στον υπολογιστή και το λογισμικό Οπτικής Αναγνώρισης προσπαθεί να ταιριάξει, με ένα βαθμό πιθανότητας, τους χαρακτήρες από το σαρωμένο αρχείο εικόνας με τα πρότυπα που έχει

αποθηκευμένα. Αν η εικόνα ενός χαρακτήρα αντιστοιχεί με αναγνωρισμένο χαρακτήρα, τότε αντιστοιχίζεται με χαρακτήρα κειμένου για τον ηλεκτρονικό υπολογιστή.

Τα περισσότερα εκτυπωμένα έγγραφα κειμένου ήταν με γραμματοσειρές Times, Courier ή Helvetica με μέγεθος 10 ως 14. Ένα πρόγραμμα αναγνώρισης χαρακτήρων έχει εικόνες σε μορφή bitmap για κάθε χαρακτήρα κάθε μεγέθους κάθε γραμματοσειράς. Το λογισμικό διάβαζε την εικόνα που σάρωνε ο σαρωτής γραμμή-γραμμή και προσπαθούσε να αντιστοιχήσει κάθε χαρακτήρα με την αντίστοιχη εικόνα. Για παράδειγμα αν το πρόγραμμα εντόπιζε ένα χαρακτήρα "Γ" τότε το πρόγραμμα έψαχνε όλα τα πρότυπα από το Α μέχρι το ω σε όλα τα αποθηκευμένα μεγέθη και αν εντόπιζε κάποια εικόνα που έμοιαζε το Γ, το αντιστοίχιζε.Η όλη διαδικασία είναι χρονοβόρα γιατί απαιτούνται πολλές επαναλήψεις για κάθε χαρακτήρα.

Εξαγωγή Χαρακτηριστικών: Η εξαγωγή χαρακτηριστικών είναι επίσης γνωστή ως Ευφυής Αναγνώριση Χαρακτήρων (Αγγλ. Intelligent Character Recognition – ICR), ή τοπολογική ανάλυση χαρακτηριστικών. Πρόκειται για ένα είδος οπτικής αναγνώρισης που δεν βασίζεται σε ακριβείς αντιστοιχήσεις με πρότυπα. Το λογισμικό λειτουργεί με ένα πιο σοφιστικό τρόπο αναγνώρισης χαρακτήρων, όπως ανίχνευση επιμέρους συστατικών στοιχείων ενός χαρακτήρα, όπως γωνίες, γραμμές, ενώσεις κτλ) Η εφαρμογή των αντιστοιχήσεων γίνεται με μορφή κανόνων. Ένας κανόνας θα μπορούσε να είναι ως έξης: Αν εντοπιστούν δύο κάθετες που κλίνουν οι μια στην άλλη "/" και "\" και η κορυφές τους ενώνονται και στο κέντρο υπάρχει μια γραμμή "-" τότε είναι το γράμμα "Α". Η εφαρμογή αυτού του κανόνα θα μπορούσε να εντοπίσει όλα τα "Α" ανεξάρτητα από την μέγεθος ή τον τύπο γραμματοσειράς που χρησιμοποιήθηκε στο έγγραφο.

Υβριδική Αναγνώριση: Οι παραπάνω μέθοδοι χρησιμοποιούνται κυρίως για αναγνώριση κειμένου που εκτυπώθηκε από ηλεκτρονικό υπολογιστή ή δακτυλογραφήθηκε. Η αναγνώριση χειρόγραφων χαρακτήρων είναι πιο πολύπλοκη διαδικασία και απαιτεί τον συνδυασμό των παραπάνω τεχνικών, καθώς και στοιχεία όπως γνώσεις για τον συγγραφέα και το περιεχόμενο του κειμένου. Τα προβλήματα με την αναγνώριση χειρογράφων οφείλονται στην καλλιγραφία (συνεχόμενη γραφή χαρακτήρων χωρίς κενό) διότι δεν μπορούν να ξεχωρίσουν πότε τελειώνει ένα γράμμα και πότε ξεκινάει ένα άλλο. Επίσης, κάθε άνθρωπος έχει διαφορετικό γραφικό χαρακτήρα, δυσχεραίνοντας την διαδικασία εφαρμογής προτύπων ή εξαγωγής χαρακτηριστικών για τον κάθε ένα. Όταν ένα λογισμικό πρέπει να αναγνωρίσει τέτοιες λέξεις, χρησιμοποιεί το νόημα του κειμένου, την γνώση του για τον συγγραφέα και τις λέξεις που ήδη αναγνώρισε.

2.3.3 ΛΟΓΙΣΜΙΚΟ ΑΝΑΓΝΩΡΙΣΗΣ ΧΑΡΑΚΤΗΡΩΝ

Desktop & Server Λογισμικό Αναγνώρισης Χαρακτήρων. Το λογισμικό Οπτικής Αναγνώρισης και Ευφυούς Αναγνώρισης χαρακτήρων είναι συστήματα τεχνίτης νοημοσύνης που θεωρούν το κείμενο ως μια ακολουθία χαρακτήρων και όχι μεμονωμένες λέξεις ή φράσεις. Βασιζόμενα στην ανάλυση των γραμμών και των καμπυλών κάθε χαρακτήρα, προσπαθούν να μαντέψουν ποιος χαρακτήρας απεικονίζεται χρησιμοποιώντας βάσεις με πρότυπα που ταιριάζει.

WebOCR & OnlineOCR. Με την ανάπτυξη της τεχνολογία της πληροφορίας, οι πλατφόρμες χρήσης λογισμικού αναγνώρισης χαρακτήρων άλλαξαν σε πολύ-πλατφόρμες με την χρήση του ηλεκτρονικού υπολογιστή, του διαδικτύου, του υπολογιστικού νέφους και τις κινητές συσκευές. Μετά από 30 χρόνια, το λογισμικό οπτικής αναγνώρισης υιοθετεί νέες μεθόδους όπως χρήση της αναγνώρισης χαρακτήρων ως υπηρεσία ιστού. Χωρίς την χρήση εξειδικευμένο λογισμικού ή την υπολογιστική ισχύ ενός υπολογιστή, ο χρήστης μπορεί να χρησιμοποιήσει την αναγνώριση χαρακτήρων με εξαιρετικά αποτελέσματα.

OCR Ειδικής Χρήσης. Λόγω του μεγάλου εύρους χρήσης της τεχνολογίας Οπτικής Αναγνώρισης Χαρακτήρων, υπήρξε η ανάγκη ανάπτυξης λογισμικού ειδικής χρήσης. Το λογισμικό ειδικής χρήσης δίνει καλύτερα αποτελέσματα σε συγκεκριμένες περιπτώσεις, παρά σε γενικές. Το λογισμικό χρησιμοποιεί κάποιους κανόνες ή κάποια φίλτρα που αντιστοιχούν μόνο σε ορισμένες

εικόνες κειμένων και εξάγει το κείμενο. Για παράδειγμα, κάποιο λογισμικό αναγνώρισης των χαρακτηριστικών μιας ταυτότητας, θα πρέπει να εφαρμόσει ειδικά φίλτρα και να διαβάσει ορισμένες περιοχές για να είναι πιο πετυχημένη η αναγνώριση.

2.4 ΕΙΣΑΓΩΓΗ ΣΤΟ NETBEANS

Το NetBeans είναι ένα ολοκληρωμένο, εξελιγμένο περιβάλλον στο οποίο αναπτύσσεται κυρίως Java γλώσσα, αλλά και άλλες γλώσσες, πιο συγκεκριμένα οι PHP, C/C++ και HTML5. Είναι επίσης μια εφαρμογή πλατφόρμας για Java εφαρμογές και άλλα. Το NetBeans είναι γραμμένο σε Java γλώσσα και μπορεί να τρέξει σε Windows, OS X, Linux, Solaris και σε άλλες πλατφόρμες που υποστηρίζουν JVM. Η πλατφόρμα NetBeans επιτρέπει την ανάπτυξη εφαρμογών από ένα σύνολο στοιχείων λογισμικού που ονομάζονται modules. Εφαρμογές βασισμένες στην πλατφόρμα του NetBeans μπορούν να επεκταθούν από τρίτους προγραμματιστές

2.4.1 ΙΣΤΟΡΙΑ ΤΟΥ NETBEANS

Το NetBeans αρχίζει το 1966 με την ονομασία Xelfi από ένα φοιτητή της Java IDE καθοδηγούμενος από το τμήμα Μαθηματικών και Φυσικής του Πανεπιστημίου του Καρόλου στην Πράγα. Το 1997 ο Roman Stanek σχηματίζει μια επιχείρηση γύρω από το έργο και παράγονται εμπορικές εκδόσεις του NetBeans IDE έως ότου αγοράστηκε από τη Sun Microsystems το 1999. Το 2010, η Sun (και έτσι και το NetBeans) εξαγοράστηκε από την Oracle.

2.4.2 ΤΩΡΙΝΕΣ ΕΚΔΟΣΕΙΣ

Το NetBeans IDE 6.0 εισήγαγε την υποστήριξη για την ανάπτυξη IDE ενότητες και πλούσιες εφαρμογές client που βασίζεται στην πλατφόρμα NetBeans , ένα Java Swing GUI Builder (παλαιότερα γνωστή ως "Project Matisse ") , βελτιωμένη υποστήριξη CVS , WebLogic 9 και JBoss 4 υποστήριξη , και πολλές βελτιώσεις editor . NetBeans 6 είναι διαθέσιμο σε επίσημα αποθετήρια των μεγάλων διανομών Linux. Το NetBeans IDE 6.5 , που κυκλοφόρησε τον Νοέμβριο του 2008 , διεύρυνε τις υφιστάμενες δυνατότητες Java EE (συμπεριλαμβανομένης της υποστήριξης Java Percistence, EJB 3 και JAX - WS) . Επιπλέον , το NetBeans Enterprise Pack υποστηρίζει την ανάπτυξη των εφαρμογών Java EE 5 των επιχειρήσεων , συμπεριλαμβανομένων των SOA εργαλείων σχεδιασμού οπτικής, εργαλεία XML σχήματος , υπηρεσίες web ενορχήστρωσης (για BPEL) , και μοντελοποίησης UML . Το NetBeans IDE Bundle για την C / C + + υποστηρίζει C / C + + και FORTRAN ανάπτυξης .

Το NetBeans IDE 6.8 είναι το πρώτο IDE που παρέχει πλήρη υποστήριξη της Java EE 6 και το GlassFish Enterprise Server, v3. Το NetBeans IDE 6.9 , που κυκλοφόρησε τον Ιούνιο του 2010 , πρόσθεσε υποστήριξη για OSGi , Spring Framework 3.0 , Java EE (JSR - 299) , Zend-πλαίσιο για την PHP , και ευκολότερη πλοήγηση κώδικα , μορφοποίηση , υποδείξεις και επανακατασκευή σε διάφορες γλώσσες. Το NetBeans IDE 7.0 κυκλοφόρησε τον Απρίλιο του 2011 . Την 1η Αυγούστου 2011, η ομάδα NetBeans IDE NetBeans κυκλοφόρησε 7.0.1 , η οποία έχει την πλήρη υποστήριξη για την επίσημη κυκλοφορία της Java SE 7 πλατφόρμα. Το NetBeans IDE 7.3 κυκλοφόρησε τον Φεβρουάριο του 2013, η οποία πρόσθεσε υποστήριξη για HTML5 και web τεχνολογίες . Το NetBeans IDE 7.4 είναι σήμερα σε τελική δοκιμή και αναμένεται αργότερα το 2013.

2.4.3 NETBEANS PLATFORM

Πλαίσιο για την απλοποίηση της ανάπτυξης εφαρμογών Java desktop Swing. Το NetBeans IDE πακέτο για Java SE περιέχει ό, τι χρειάζεται για να αρχίσει να αναπτύσσει NetBeans plugins και NetBeans Platform βασισμένες εφαρμογές· καμία πρόσθετη SDK δεν απαιτείται. Οι αιτήσεις μπορούν να εγκαταστήσουν μονάδες δυναμικά. Κάθε αίτηση μπορεί να περιλαμβάνει τη μονάδα Κέντρο Update για να επιτρέψει στους χρήστες της εφαρμογής να κατεβάσουν ψηφιακή υπογραφή αναβάθμισης και νέες δυνατότητες απευθείας στην εφαρμογή που εκτελείται. Επανεγκατάσταση μιας αναβάθμισης ή μιας νέα έκδοσης δεν επιβάλλει στους χρήστες να κατεβάσουν ολόκληρη την εφαρμογή ξανά.

Η πλατφόρμα προσφέρει επαναχρησιμοποιήσιμες υπηρεσίες κοινές για desktop εφαρμογές , επιτρέποντας στους προγραμματιστές να επικεντρωθούν στη λογική ειδικά για την εφαρμογή τους.

Ανάμεσα στα χαρακτηριστικά της πλατφόρμας είναι:

-User interface διαχείρισης (π.χ. μενού και γραμμές εργαλείων)

-Χρήστης ρύθμισης διαχείρισης

-Διαχείριση αποθήκευσης (αποθήκευση και τη φόρτωση κάθε είδους δεδομένα)

-Διαχείριση παραθύρων

-Οδηγός (υποστηρίζει βήμα-προς- βήμα διαλόγου)

-NetBeans Visual Βιβλιοθήκη

-Ολοκληρωμένα εργαλεία ανάπτυξης

Το NetBeans IDE είναι ένα δωρεάν , ανοιχτού κώδικα , cross-platform IDE με ενσωματωμένη υποστήριξη για τη γλώσσα προγραμματισμού Java.

2.4.4 NETBEANS IDE

Το NetBeans IDE είναι ένα open -source ολοκληρωμένο περιβάλλον ανάπτυξης. Το NetBeans IDE υποστηρίζει την ανάπτυξη όλων των τύπων εφαρμογών Java (Java SE (συμπεριλαμβανομένων JavaFX) , Java ME , web , EJB και εφαρμογών κινητής τηλεφωνίας) από το κουτί.

Επεκτασιμότητα : Όλες οι λειτουργίες του IDE παρέχονται από τις μονάδες. Κάθε ενότητα παρέχει μια καλά καθορισμένη λειτουργία, όπως υποστήριξη για τη γλώσσα Java , μοντάζ , ή υποστήριξη για το σύστημα εκδόσεων CVS και SVN. Το NetBeans περιέχει όλες τις ενότητες που απαιτούνται για την ανάπτυξη Java σε μία μόνο λήψη , επιτρέποντας στο χρήστη να αρχίσει να λειτουργεί αμέσως. Τα Modules επιτρέπουν επίσης στο NetBeans να επεκταθεί. Νέα χαρακτηριστικά , όπως η υποστήριξη για άλλες γλώσσες προγραμματισμού, μπορούν να προστεθούν με την εγκατάσταση πρόσθετων μονάδων. Για παράδειγμα, η Sun Studio, Sun Java Studio Enterprise, και Sun Java Studio Creator από την Sun Microsystems είναι όλα με βάση το NetBeans IDE.

2.4.5 INTEGRATED MODULES (ΕΝΣΩΜΑΤΩΜΕΝΕΣ ΜΟΝΑΔΕΣ)

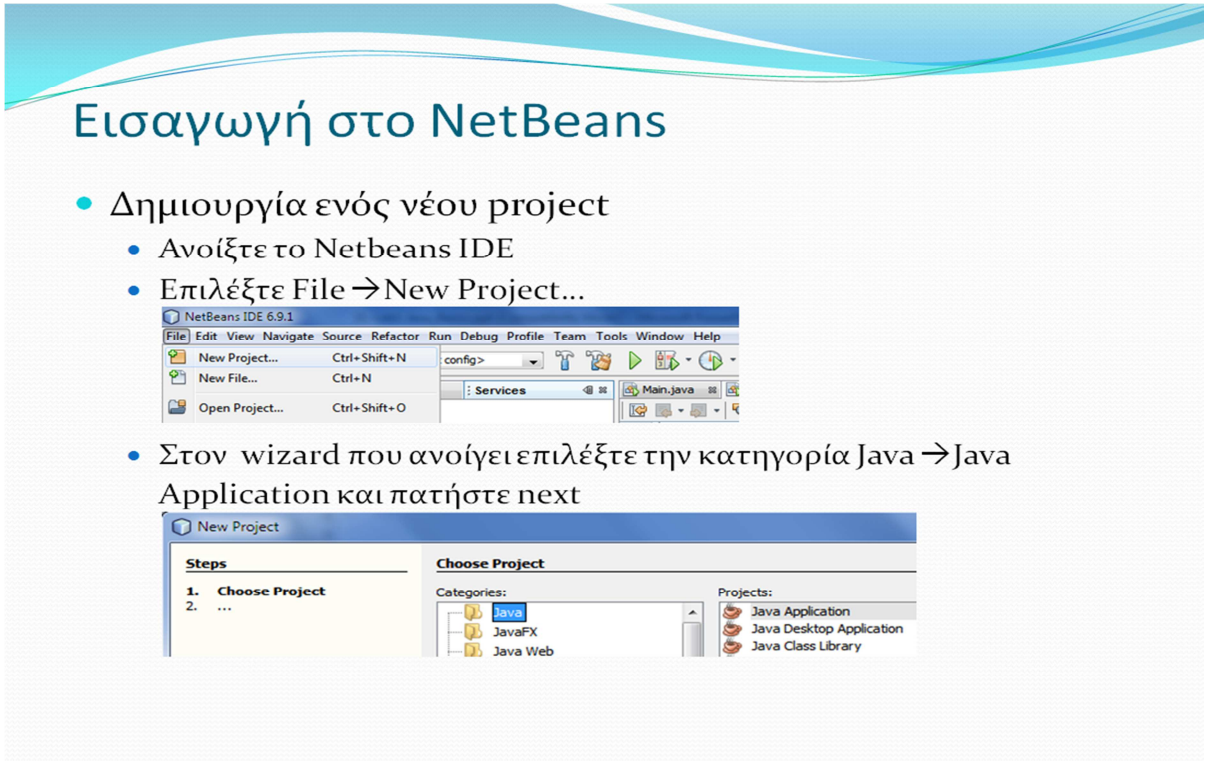
NetBeans Profiler: Το NetBeans Profiler είναι ένα εργαλείο για την παρακολούθηση των εφαρμογών Java. Βοηθά τους προγραμματιστές να βρουν τις διαρροές μνήμης και να

βελτιστοποίησουν την ταχύτητα. Το Profiler βασίζεται σε ένα έργο Sun που ονομάστηκε JFluid. Αυτή η έρευνα αποκάλυψε συγκεκριμένες τεχνικές που μπορούν να χρησιμοποιηθούν για να μειώσουν την επιβάρυνση των χαρακτηριστικών μιας εφαρμογής Java. Μία από αυτές τις τεχνικές είναι δυναμική οργάνων bytecode, το οποίο είναι ιδιαίτερα χρήσιμο για τη σκιαγράφηση μεγάλων εφαρμογών Java. Χρησιμοποιώντας δυναμική οργάνων bytecode και επιπλέον αλγόριθμων, το NetBeans Profiler είναι σε θέση να αποκτήσει runtime πληροφορίες σχετικά με τις αιτήσεις που είναι πολύ μεγάλες ή σύνθετες για άλλα profilers.

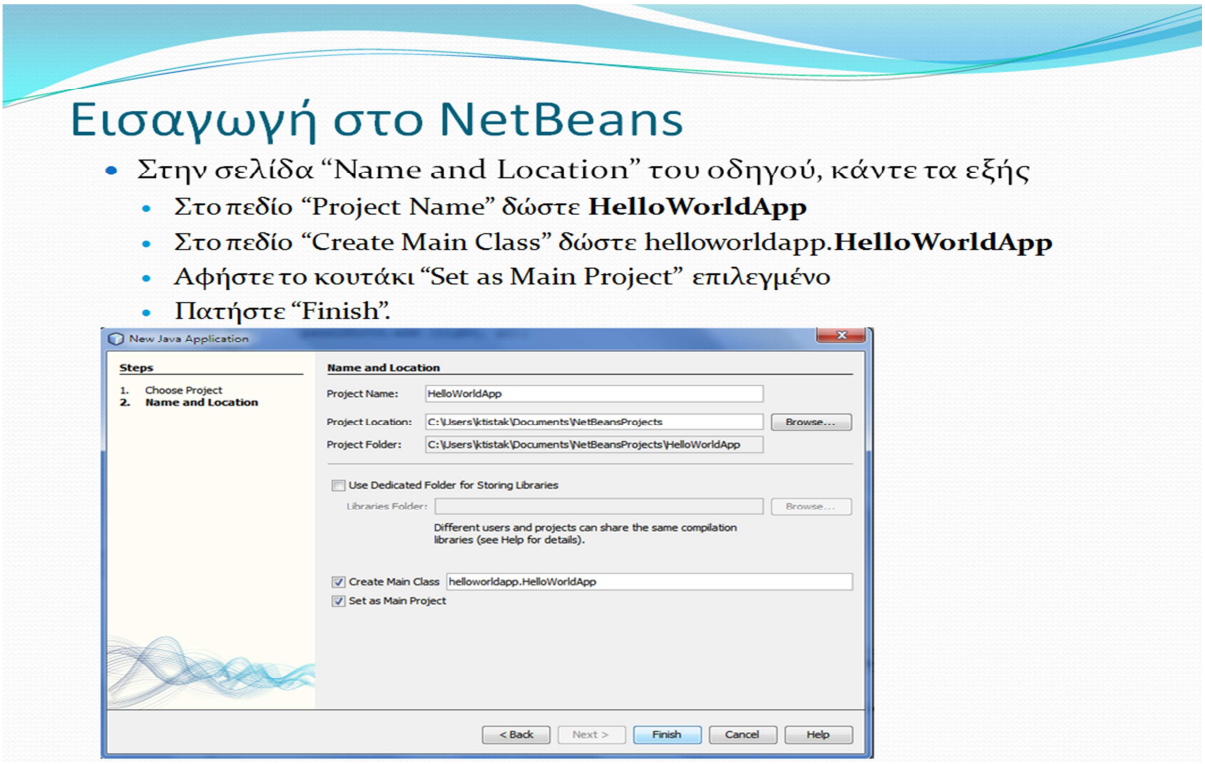
GUI design tool: Παλαιότερα ήταν γνωστό ως έργο Matisse, το GUI design-εργαλείο επιτρέπει στους προγραμματιστές να Swing το πρωτότυπο και το σχεδιασμό GUIs, σύροντας και τοποθετώντας τα GUI στοιχεία. Ο κατασκευαστής του GUI έχει ενσωματωμένη υποστήριξη για το JSR 295, αλλά η υποστήριξη για το JSR 296 (Swing Application Framework) απομακρύνθηκε σε 7.1.

NetBeans JavaScript editor: Το NetBeans JavaScript editor παρέχει εκτεταμένη υποστήριξη για JavaScript, Ajax, και CSS. Τα JavaScript χαρακτηριστικά περιλαμβάνουν επεξεργαστή επισήμανσης σύνταξης, επανκατασκευής, την ολοκλήρωση κώδικα για τα εγγενή αντικείμενα και τις λειτουργίες, την παραγωγή της JavaScript τάξης σκελετών, παραγωγή Ajax callbacks από ένα πρότυπο, και αυτοματοποιημένους ελέγχους συμβατότητας προγράμματος περιήγησης.

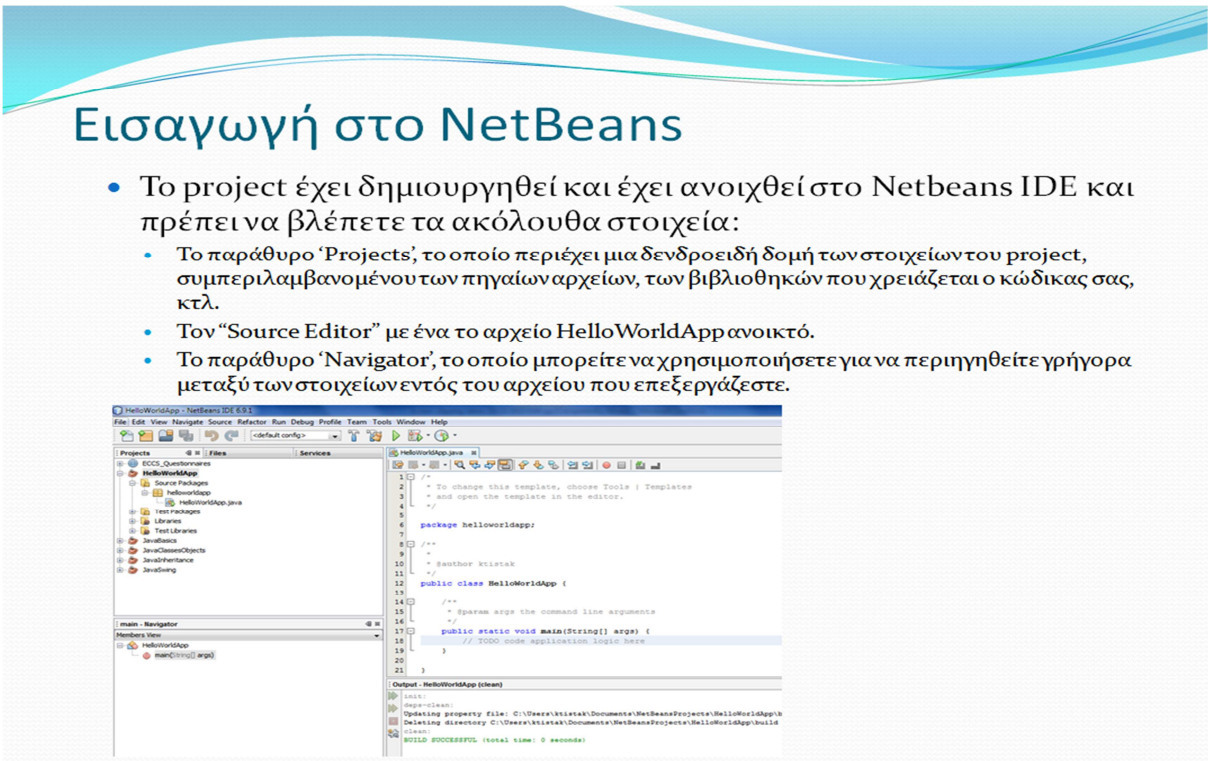
2.4.6 ΠΑΡΑΔΕΙΓΜΑ ΔΗΜΙΟΥΡΓΙΑΣ PROJECT ΣΤΟ NETBEANS



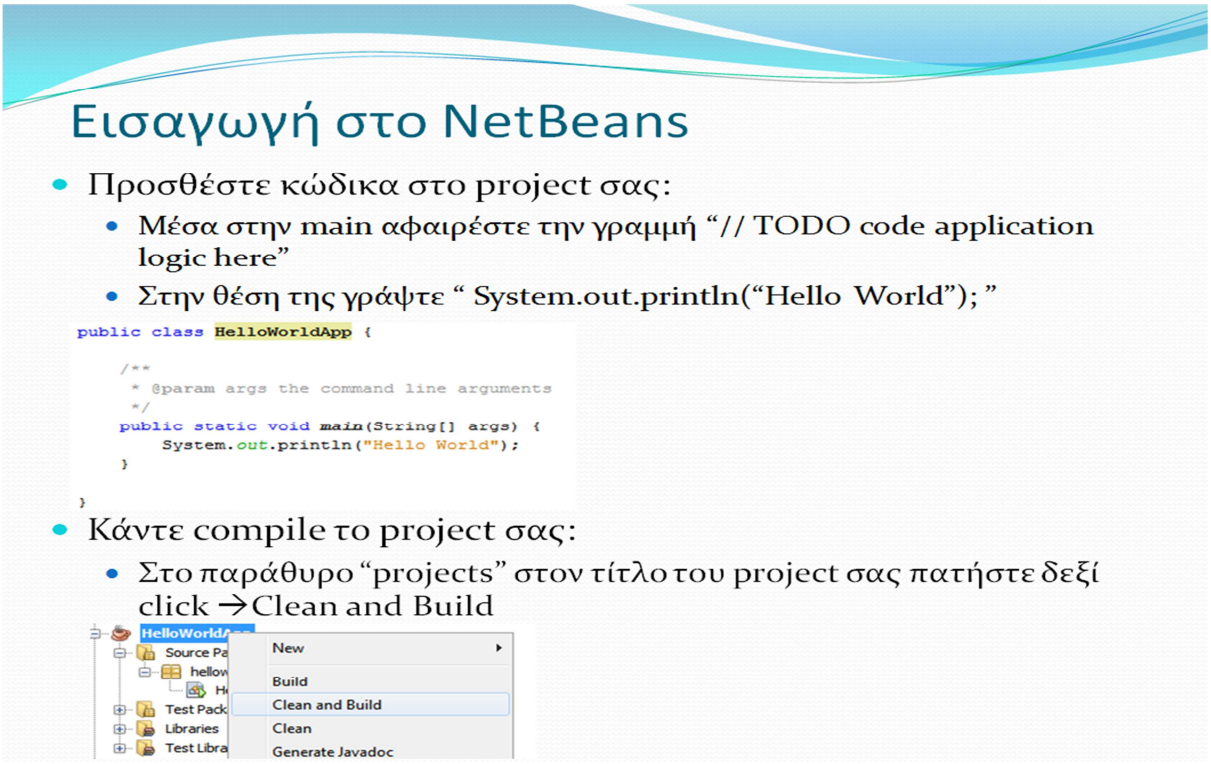
Εικόνα 21: Παράδειγμα NetBeans 1



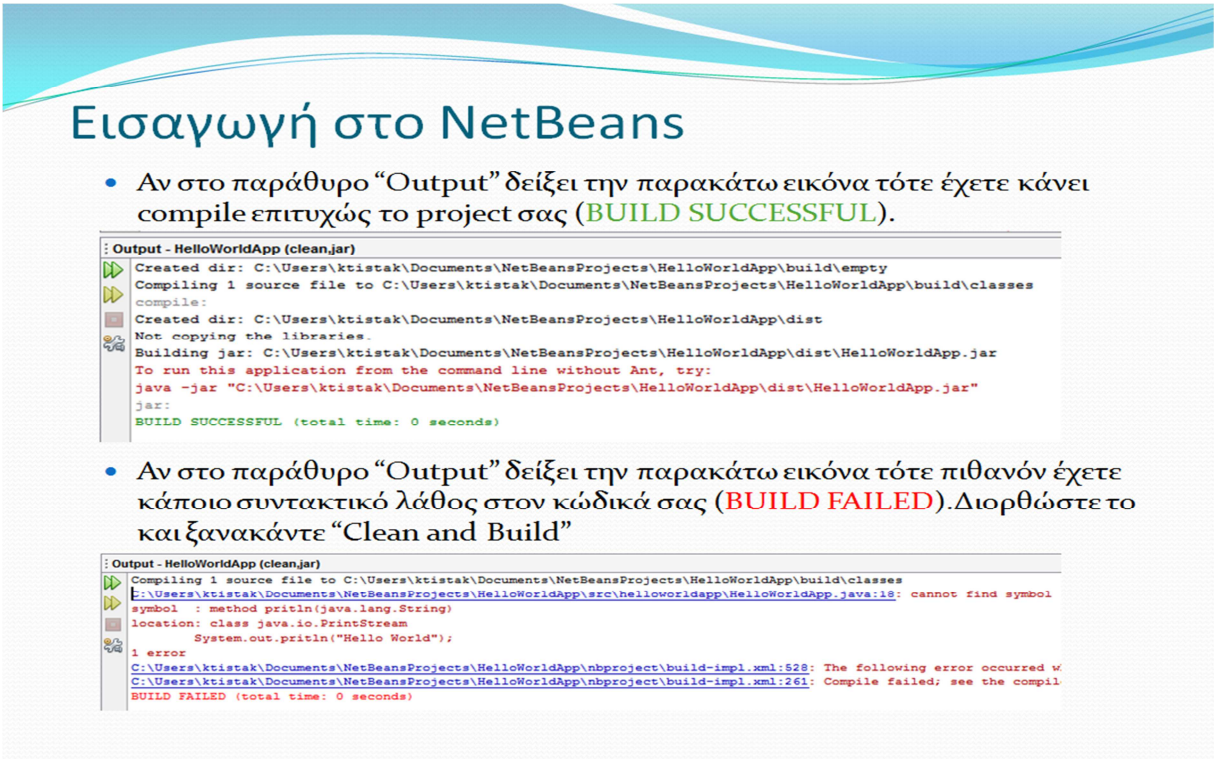
Εικόνα 22: Παράδειγμα NetBeans 2



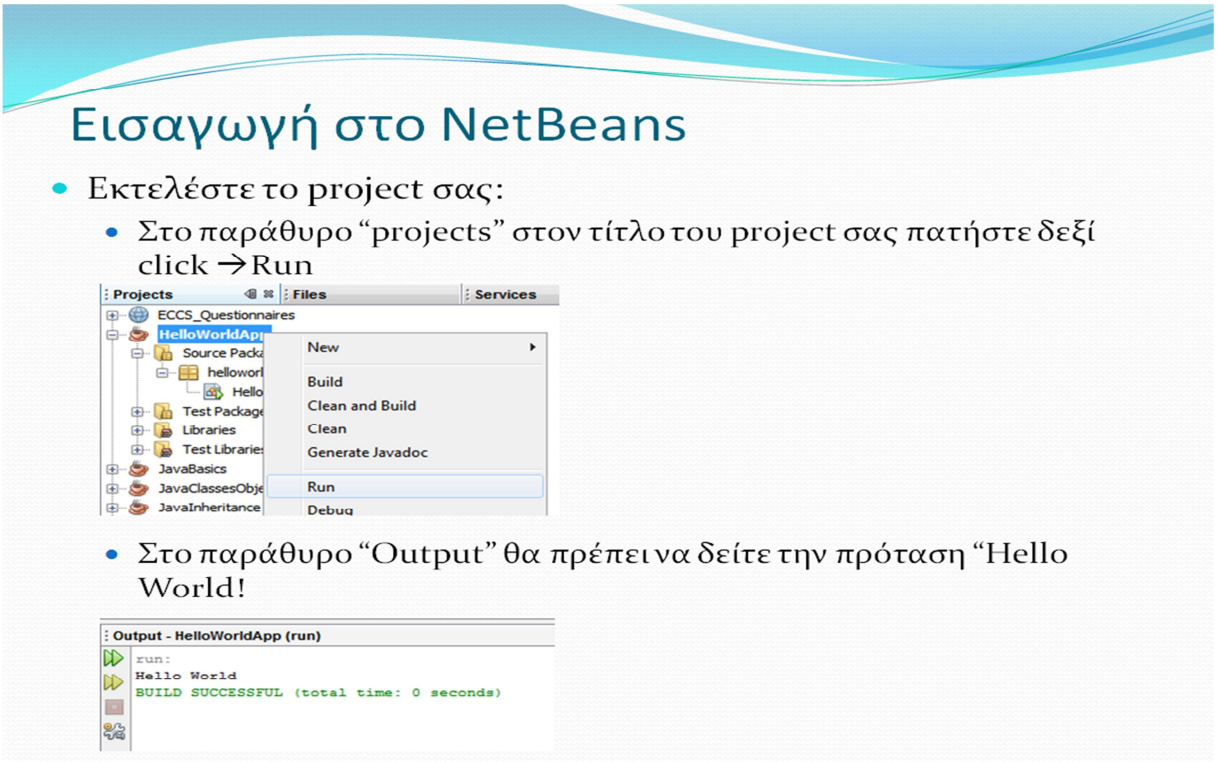
Εικόνα 23: Παράδειγμα NetBeans 3



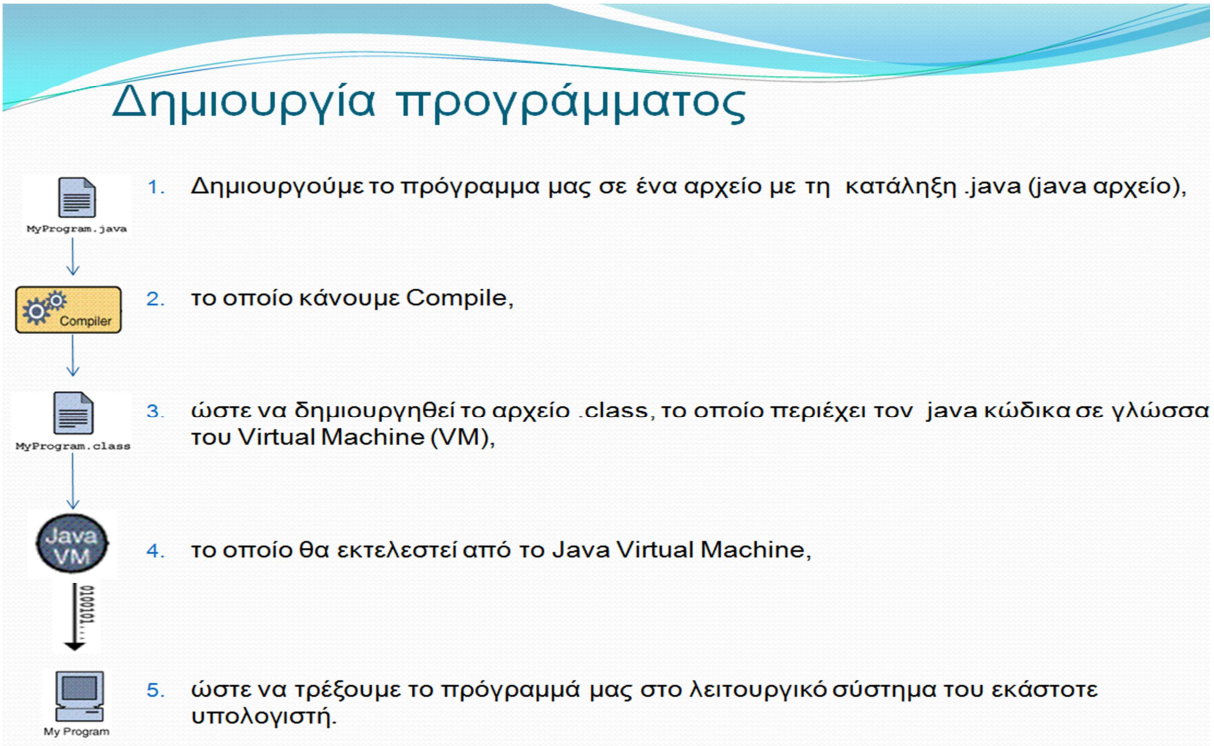
Εικόνα 24: Παράδειγμα NetBeans 4



Εικόνα 25: Παράδειγμα NetBeans 5



Εικόνα 26: Παράδειγμα NetBeans 6



Εικόνα 27: Παράδειγμα δημιουργίας προγράμματος στο NetBeans

3 ΑΥΤΟΜΑΤΗ ΔΙΟΡΘΩΣΗ

Η Αυτόματη Διόρθωση σαν γενικός όρος είναι μια αυτόματη σύγκριση μεταξύ διαφόρων ειδών αρχείων. Επίσης, συγκρίνει το περιεχόμενο των αρχείων, εντοπίζει τα κοινά περιεχόμενά τους και τις διαφορές τους. Το αποτέλεσμα της σύγκρισης, που συχνά αποκαλείται «diff», είναι δυνατό να παρουσιάζεται σε ένα γραφικό περιβάλλον χρήστη ή να χρησιμοποιηθεί ως μέρος διεργασιών σε networks, file systems, ή revision control. Μερικά ευρέως χρησιμοποιούμενα προγράμματα σύγκρισης αρχείων είναι τα diff, cmp, FileMerge, WinMerge, Beyond Compare, και Microsoft File Compare. Πολλά προγράμματα επεξεργασίας και επεξεργαστές κειμένου εκτελούν σύγκριση αρχείων για να τονίσουν τις αλλαγές σε ένα έγγραφο.

Τα περισσότερα εργαλεία σύγκρισης αρχείων βρίσκουν την πιο κοινή υποακολουθία μεταξύ δύο αρχείων. Οποιαδήποτε δεδομένα που δεν είναι στην κοινή υποαλληλουχία, παρουσιάζονται ως «εισαγωγή» ή «διαγραφή». Το 1978, ο Paul Heckel δημοσίευσε έναν αλγόριθμο που προσδιορίζει τα πιο μετακινούμενα τμήματα του κειμένου. Αυτό χρησιμοποιείται στο IBM History Flow tool. Άλλα προγράμματα σύγκρισης αρχείων βρίσκουν block moves. Ορισμένα εξειδικευμένα εργαλεία σύγκρισης αρχείων βρίσκουν τη μεγαλύτερη αυξανόμενη ακολουθία μεταξύ δύο αρχείων. Το rsync πρωτόκολλο χρησιμοποιεί μία κυλιόμενη hash συνάρτηση για να συγκρίνει δύο αρχεία σε δύο μακρινούς υπολογιστές με χαμηλή επιβάρυνση επικοινωνίας. Η σύγκριση αρχείων σε επεξεργαστές κειμένου είναι συνήθως σε επίπεδο λέξης, ενώ η σύγκριση με τα περισσότερα εργαλεία προγραμματισμού σε επίπεδο γραμμής. Η σύγκριση ανά Byte ή ανά χαρακτήρα είναι χρήσιμη σε κάποιες εξειδικευμένες εφαρμογές.

Στην περίπτωση μας, έχουμε δημιουργήσει μία εφαρμογή που μπορεί να συγκρίνει αρχεία κειμένου μεταξύ τους και να υπολογίζει τις διαφορές τους. Η υλοποίησή της στηρίχθηκε σε ένα σωστό αρχείο κειμένου, το οποίο μπορεί να συγκριθεί με ένα ή περισσότερα αρχεία. Στο πρόγραμμά μας θα υπάρχει η δυνατότητα επιλογής θετικής ή/και αρνητικής βαθμολόγησης (εφόσον υπάρχει). Ο χρήστης θα επιλέγει τα αρχεία που επιθυμεί να συγκρίνει και στη συνέχεια η εφαρμογή μας θα υπολογίζει τους βαθμούς των εξεταζόμενων και θα τους αντιστοιχίζει με τα ΑΜ τους. Τέλος, θα παρέχεται η δυνατότητα εκτύπωσης των αποτελεσμάτων ή εξαγωγή αυτών σε ξεχωριστό αρχείο κειμένου.

3.1 ΑΠΑΙΤΗΣΕΙΣ ΣΥΣΤΗΜΑΤΟΣ

Για να είναι εκτελέσιμο το .jar project μας πρέπει να υπάρχει Java(TM) SE Runtime Environment. Επομένως μπορεί να τρέξει σε όλα σχεδόν τα λειτουργικά συστήματα που υποστηρίζουν java. Opens with: Java(TM) Platform SE binary.[1]

Windows System Requirements:

Windows XP & νεότερα λειτουργικά συστήματα

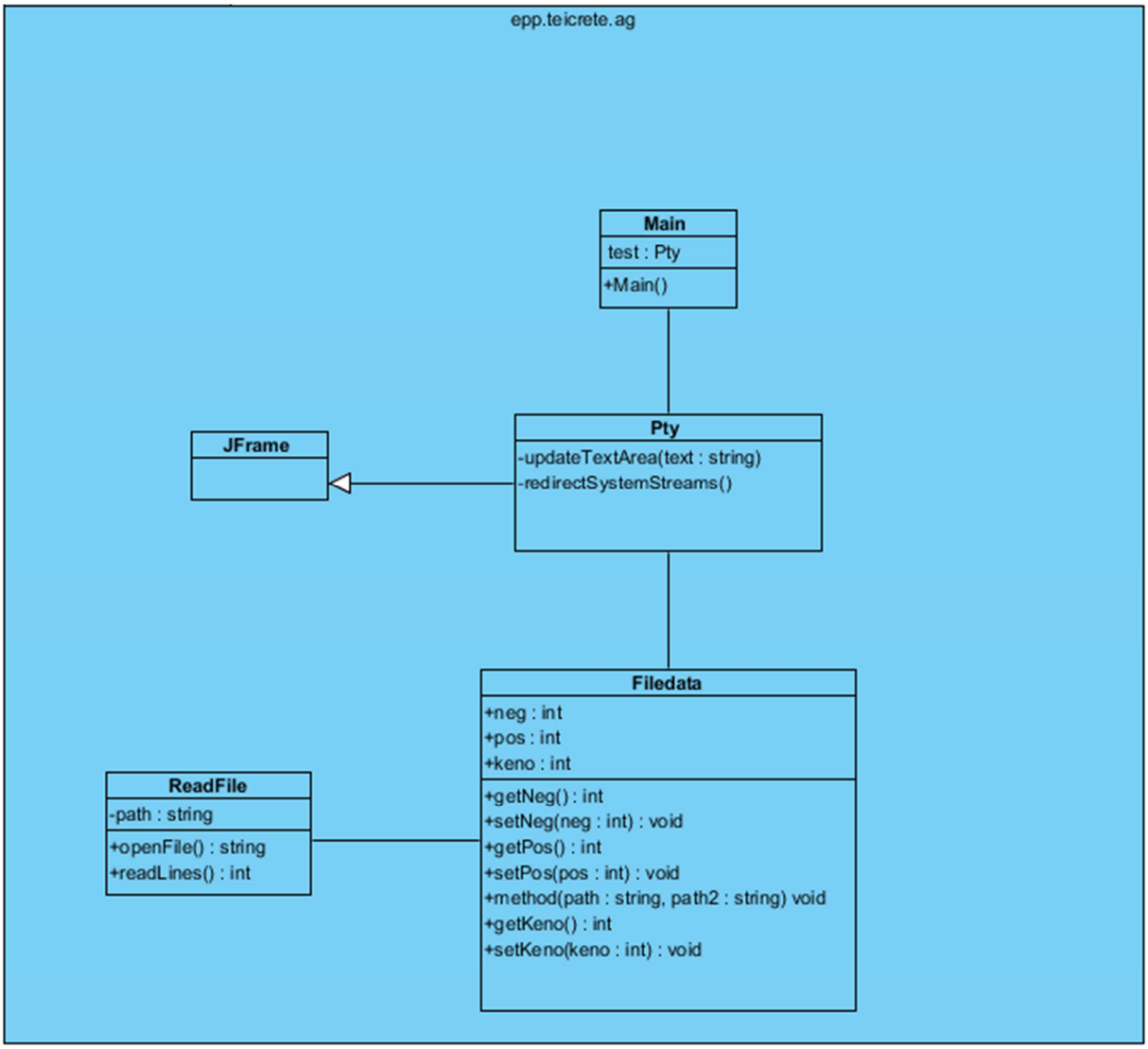
Windows Server 2008

A Pentium 2 266 MHz or faster processor with at least 128 MB of physical RAM is recommended*. You will also need a minimum of 124 MB of free disk space.

*Windows XP minimum is 64MB RAM.

[1] http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html

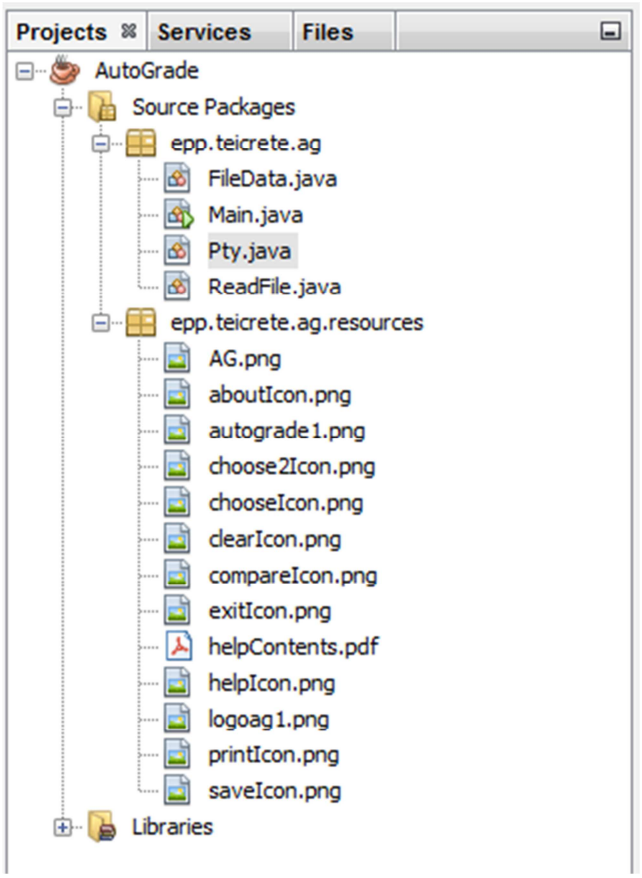
3.2 ΣΧΕΔΙΑΣΜΟΣ ΥΛΟΠΟΙΗΣΗΣ - UML



Εικόνα 28: AutoGrade's UML

3.3 ΑΝΑΠΤΥΞΗ & ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ

Την εφαρμογή μας την αναπτύξαμε χρησιμοποιώντας το NetBeans. Δημιουργήσαμε το project AutoGrade και στο package μας epp.teicrete.AG υλοποιήσαμε τις κλάσεις που χρειαστήκαμε (Main, ReadFile, FileData, Pty).



Εικόνα 29: Η δομή του AutoGrade

Παρακάτω θα παρουσιαστούν οι κλάσεις της εφαρμογής και η πλειοψηφία των μεθόδων τους. Η παρουσίαση θα προσφέρει μία περιληπτική περιγραφή των μεθόδων. Στις εικόνες παραθέτουμε το σημείο που βρίσκεται ο κώδικας και μόνο. Η συνολική μορφή του κώδικα βρίσκεται στο κεφάλαιο 3.2.1.

1. ReadFile

Η μέθοδος readLines() διαβάζει την κάθε γραμμή του κειμένου μας και επιστρέφει τον αριθμό των σειρών.

```
31      int readLines() throws IOException {...}
45  }
```

Εικόνα 30: Μέθοδος readLines

Η μέθοδος OpenFile() καταχωρεί τα δεδομένα της κάθε γραμμής του κειμένου μας στην textData.

```
14
15      public String[] OpenFile() throws IOException {...}
30
```

Εικόνα 31: Μέθοδος OpenFile

2. **FileData**

Η μέθοδος compare() διαχειρίζεται τα δεδομένα των αρχείων που επιλέγει ο χρήστης. Συγκρίνει τα αρχεία μεταξύ τους, υπολογίζει τα λάθη ενώ εξάγει και το ΑΜ του εξεταζόμενου.

```
12
13      public void compare(String path, String path2) throws Exception {...}
64
```

Εικόνα 32: Μέθοδος Compare

Επίσης έχουμε getters & setters για να έχουμε ευελιξία στην χρησιμοποίηση των μεταβλητών που σχετίζονται με την βαθμολογία (αρνητική βαθμολόγηση, κ.α.).

```
65      public int getNeg() {
66          return neg;
67      }
68
69      public void setNeg(int neg) {
70          this.neg = neg;
71      }
72
73      public int getPos() {
74          return pos;
75      }
76
77      public void setPos(int pos) {
78          this.pos = pos;
79      }
80
81      public int getKeno() {
82          return keno;
83      }
84
85      public void setKeno(int keno) {
86          this.keno = keno;
87      }
88  }
```

Εικόνα 33: getters & setters βαθμολόγησης

3. **Pty**

Πρόκειται για μία κλάση όπου φτιάχνουμε το swing και διαχειριζόμαστε όλα τα γραφικά στοιχεία μας. Η updateTextArea() και η redirectSystemStreams() χρησιμοποιούνται για την κατάλληλη λειτουργία και την εμφάνιση των αποτελεσμάτων στο textArea που έχουμε δημιουργήσει.

```
392
393      private void updateTextArea(final String text) {...}
400
401      private void redirectSystemStreams() {...}
422  }
423
```

Εικόνα 34: TextArea

Για να μπορεί ο χρήστης να επιλέξει τα αρχεία του (γραπτά και πρωτότυπο), δημιουργήσαμε τα choose και choose2 Buttons τα οποία με τη βοήθεια των ActionListeners επιτρέπουν στο χρήστη να αναζητήσει τα αρχεία του.

```
200          choose.addActionListener(new ActionListener() {
                 public void actionPerformed(ActionEvent e) {...}
225          });
226
227          choose2.addActionListener(new ActionListener() {
228              @Override
                 public void actionPerformed(ActionEvent e) {...}
252          });
```

Εικόνα 35: Επιλογή αρχείων με τα choose & choose2 Buttons

Για την επιλογή των σωστών χαρακτήρων και μόνο, δημιουργήσαμε τα textfld1 και textfld2.

```
 93
 94          textfld1.addKeyListener(new KeyListener() {...});
111
112          textfld2.addKeyListener(new KeyListener() {...});
```

Εικόνα 36: textfld1 & textfld2

Δημιουργήσαμε την επιλογή του clear text, ώστε να μπορεί ο χρήστης να σβήνει το περιεχόμενο του textArea και των txtfld. Έτσι μπορεί να επαναλάβει την σύγκριση με καινούριο panel και διαφορετικά δεδομένα. Είναι σαν «new game» με τη διαφορά πως δε θα χαθούν τα αρχεία που έχει ήδη επιλέξει.

```
129
130          menuItem.addActionListener(new ActionListener() {
131              @Override
                 public void actionPerformed(ActionEvent e) {
133
134                  textArea.setText("");
135                  textfld1.setText("");
136                  textfld2.setText("");
137                  repaint();
138                  validate();
139              }
140          });
141
```

Εικόνα 37: Clear Output

Δημιουργήσαμε την επιλογή exit, η οποία δίνει τη δυνατότητα στον χρήστη να τερματίσει το πρόγραμμα.

```
142         menuItem2.addActionListener(new ActionListener() {
143             @Override
145             public void actionPerformed(ActionEvent e) {
146                 dispose();
147             }
147         });
```

Εικόνα 38: Επιλογή exit

Δημιουργήσαμε την επιλογή save, η οποία δίνει τη δυνατότητα στον χρήστη να αποθηκεύει τα δεδομένα (αποτελέσματα) σε εξωτερικό αρχείο txt.

```
148
149         save.addActionListener(new ActionListener() {
150             @Override
179             public void actionPerformed(ActionEvent e) {...}
179         });
```

Εικόνα 39: Επιλογή save

Για να μπορεί ο χρήστης να εκτυπώνει τα αποτελέσματά του, δημιουργήσαμε την επιλογή print.

```
181         print.addActionListener(new ActionListener() {
182             @Override
197             public void actionPerformed(ActionEvent ae) {...}
197         });
```

Εικόνα 40: Επιλογή print

Με το Button "AutoGrade" εκτελείται η σύγκριση των αρχείων που επιλέξαμε παραπάνω. Σε αυτή τη μέθοδο αναπτύσουμε τους κανόνες που καθορίζουν τον υπολογισμό του τελικού βαθμού του εξεταζόμενου.

```
254         comp.addActionListener(new ActionListener() {
255             @Override
389             public void actionPerformed(ActionEvent e) {...}
389         });
```

Εικόνα 41: Button AutoGrade

4. Main

```
28          Pty test = new Pty();
29          String imagePath = "icon/tick.png";
30          InputStream imgStream = test.getClass().getResourceAsStream(imagePath);
31          BufferedImage myImg = ImageIO.read(imgStream);
32          test.setIconImage(myImg);
33          test.setTitle("AutoGrade");
34          test.addWindowListener(new WindowAdapter() {
35              public void windowClosing(WindowEvent e) {
36                  System.exit(0);
37              }
38          });
39          test.pack();
40          test.setVisible(true);//Create and show the GUI.
41      }
42  }
```

Εικόνα 42: Στιγμυότυπο της κλάσης main

3.3.1 THE CODE

package epp.teicrete.ag;

public class Pty extends JFrame {

JMenuBar menuBar = new JMenuBar();

JMenu menu = new JMenu("File");

JMenu menu2 = new JMenu("Tools");

JMenu menu3 = new JMenu("Help");

JMenuItem save = new JMenuItem("Save");

JMenuItem print = new JMenuItem("Print");

JMenuItem clear = new JMenuItem("Clear Output");

JMenuItem exit = new JMenuItem("Exit");

JMenuItem help = new JMenuItem("Help Contents");

JMenuItem about = new JMenuItem("About");

JButton comp = new JButton("AutoGrade");

JButton choose = new JButton("Choose Tests");

JTextArea textArea = new JTextArea(30, 15);

String cwd = System.getProperty("user.dir");

final JFileChooser jfc = new JFileChooser(cwd);

JScrollPane sp = new JScrollPane();

JPanel pan1 = new JPanel(new BorderLayout());

```
JPanel pan2 = new JPanel(new FlowLayout());
JPanel pan3 = new JPanel(new FlowLayout());
JLabel ver = new JLabel("v1.0     ");
JPanel pan4 = new JPanel(new BorderLayout());
java.util.List list = new ArrayList();
JLabel pos = new JLabel("Optional rules for each right or wrong answer:          Positive score:");
JLabel neg = new JLabel(" Penalty score:");
JTextField textfld1 = new JTextField(4);
JTextField textfld2 = new JTextField(4);
JButton choose2 = new JButton("Choose Correct Test");
String cwd2 = System.getProperty("user.dir");
final JFileChooser jfc2 = new JFileChooser(cwd2);
JFileChooser chooseSave = new JFileChooser();
public int decimalPlaces = 2;
FileNameExtensionFilter filterdoc = new FileNameExtensionFilter(
      "Microsoft Word Documents 97 - 2003", "doc");
FileNameExtensionFilter filtertxt = new FileNameExtensionFilter(
      "Text File", "txt");
FileNameExtensionFilter filterxls = new FileNameExtensionFilter(
      "Microsoft Excel Documents 97 - 2003", "xls");
public Pty() throws IOException {
   this.add(menuBar, BorderLayout.NORTH);
   this.add(pan1, BorderLayout.CENTER);
   this.add(pan4, BorderLayout.SOUTH);
   pan4.add(ver, BorderLayout.EAST);
   ver.setFont(new Font("Times New Roman",Font.BOLD,11));
   ver.setForeground(Color.DARK_GRAY);
   pos.setFont(new Font("Times New Roman",Font.BOLD,11));
   neg.setFont(new Font("Times New Roman",Font.BOLD,11));
   pan1.add(pan2, BorderLayout.NORTH);
   pan1.add(pan3, BorderLayout.SOUTH);
```

```
pan2.add(pos);
pan2.add(textfld1);
pan2.add(neg);
pan2.add(textfld2);
menuBar.add(menu);
menuBar.add(menu2);
menuBar.add(menu3);
menu.add(save);
menu.add(exit);
menu2.add(clear);
menu2.add(print);
menu3.add(help);
menu3.add(about);
pan3.add(choose);
pan3.add(choose2);
pan3.add(comp);
String textAreaImgPath = "resources/AG.png";
InputStream textAreaImgStream = this.getClass().getResourceAsStream(textAreaImgPath);
final BufferedImage textAreaImg = ImageIO.read(textAreaImgStream);
textArea.setOpaque(false);
textArea.setMargin(new Insets(5, 5, 5, 5));
textArea.setEditable(true);
textArea.setFont(new Font("Times new Roman", Font.BOLD, 11));
textArea.setLineWrap(true);
textArea.setWrapStyleWord(true);
pan1.add(sp, BorderLayout.CENTER);
pan1.setBackground(new Color(220,220,220));
pan2.setBackground(new Color(141,198,63));
pan3.setBackground(new Color(141,198,63));
pan4.setBackground(new Color(220,220,220));
textfld1.addKeyListener(new KeyListener() {
```

```
        @Override
        public void keyTyped(KeyEvent e) {
            char c = e.getKeyChar();
            if (!(Character.isDigit(c) || c == KeyEvent.VK_BACK_SPACE || c == KeyEvent.VK_DELETE || c == KeyEvent.VK_PERIOD)) {
                e.consume();
            }
        }
        @Override
        public void keyPressed(KeyEvent e) {
        }
        @Override
        public void keyReleased(KeyEvent e) {
        }
    });
    textfld2.addKeyListener(new KeyListener() {
        @Override
        public void keyTyped(KeyEvent e) {
            char c = e.getKeyChar();
            if (!(Character.isDigit(c) || c == KeyEvent.VK_BACK_SPACE || c == KeyEvent.VK_DELETE || c == KeyEvent.VK_PERIOD || c == KeyEvent.VK_MINUS)) {
                e.consume();
            }
        }
        @Override
        public void keyPressed(KeyEvent e) {
        }
        @Override
        public void keyReleased(KeyEvent e) {
        }
    });
    clear.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_D, ActionEvent.CTRL_MASK));
```

```
String clearImgPath = "resources/clearIcon.png";
InputStream clearImgStream = this.getClass().getResourceAsStream(clearImgPath);
BufferedImage clearImg = ImageIO.read(clearImgStream);
clear.setIcon(new javax.swing.ImageIcon(clearImg));
clear.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      textArea.setText("");
      textfld1.setText("");
      textfld2.setText("");
      repaint();
      validate();
   }
});
exit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F4, ActionEvent.ALT_MASK));
String exitImgPath = "resources/exitIcon.png";
InputStream exitImgStream = this.getClass().getResourceAsStream(exitImgPath);
BufferedImage exitImg = ImageIO.read(exitImgStream);
exit.setIcon(new javax.swing.ImageIcon(exitImg));
exit.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      dispose();
   }
});
save.setAccelerator(KeyStroke.getKeyStroke(
      KeyEvent.VK_S, ActionEvent.CTRL_MASK));
String saveImgPath = "resources/saveIcon.png";
InputStream saveImgStream = this.getClass().getResourceAsStream(saveImgPath);
BufferedImage saveImg = ImageIO.read(saveImgStream);
save.setIcon(new javax.swing.ImageIcon(saveImg));
```

```
save.addActionListener(new ActionListener() {…});
print.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_P, ActionEvent.CTRL_MASK));
String printImgPath = "resources/printIcon.png";
InputStream printImgStream = this.getClass().getResourceAsStream(printImgPath);
BufferedImage printImg = ImageIO.read(printImgStream);
print.setIcon(new javax.swing.ImageIcon(printImg));
print.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent ae) {
      try {
         boolean com = textArea.print();
         if (com) {
            JOptionPane.showMessageDialog(null, "Printing", "Information",
JOptionPane.INFORMATION_MESSAGE);
         } else {
            JOptionPane.showMessageDialog(null, "Cancel Printing", "Printer",
JOptionPane.ERROR_MESSAGE);
         }
      } catch (PrinterException e) {
         JOptionPane.showMessageDialog(null, e);
      }
   }
});
String helpImgPath = "resources/helpIcon.png";
InputStream helpImgStream = this.getClass().getResourceAsStream(helpImgPath);
BufferedImage helpImg = ImageIO.read(helpImgStream);
help.setIcon(new javax.swing.ImageIcon(helpImg));
help.addActionListener(new ActionListener() {…});☻
String aboutImgPath = "resources/aboutIcon.png";
InputStream aboutImgStream = this.getClass().getResourceAsStream(aboutImgPath);
BufferedImage aboutImg = ImageIO.read(aboutImgStream);
about.setIcon( new javax.swing.ImageIcon(aboutImg));
```

```
        about.addActionListener( new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ae) {
                JOptionPane.showMessageDialog(null, "Name: AutoGrade\nRelease Year: 2013\nDevelopers: Manolarakis G., Kalamoudakos K."
                        + "\nVersion: v1.0\n© Copyright: ProMan..:)", "Information", JOptionPane.INFORMATION_MESSAGE);
            }
        });
        jfc.setMultiSelectionEnabled(true);
        String chooseImgPath = "resources/chooseIcon.png";
        InputStream chooseImgStream = this.getClass().getResourceAsStream(chooseImgPath);
        BufferedImage chooseImg = ImageIO.read(chooseImgStream);
        choose.setIcon(new javax.swing.ImageIcon(chooseImg));
        choose.addActionListener(new ActionListener() {…});
        String choose2ImgPath = "resources/choose2Icon.png";
        InputStream choose2ImgStream = this.getClass().getResourceAsStream(choose2ImgPath);
        BufferedImage choose2Img = ImageIO.read(choose2ImgStream);
        choose2.setIcon(new javax.swing.ImageIcon(choose2Img));
        choose2.addActionListener(new ActionListener() {…});
        String compImgPath = "resources/compareIcon.png";
        InputStream compImgStream = this.getClass().getResourceAsStream(compImgPath);
        BufferedImage compImg = ImageIO.read(compImgStream);
        comp.setIcon(new javax.swing.ImageIcon(compImg));
comp.addActionListener(new ActionListener() {...} );
        redirectSystemStreams();
    }
    private void updateTextArea(final String text) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                textArea.append(text);
            }
```

```
        });
    }
    private void redirectSystemStreams() {
        OutputStream out = new OutputStream() {
            @Override
            public void write(int b) throws IOException {
                updateTextArea(String.valueOf((char) b));
            }
            @Override
            public void write(byte[] b, int off, int len) throws IOException {
                updateTextArea(new String(b, off, len));
            }
            @Override
            public void write(byte[] b) throws IOException {
                write(b, 0, b.length);
            }
        };
        System.setOut(new PrintStream(out, true));
        System.setErr(new PrintStream(out, true));
    }
}
public class ReadFile {
    private String path;
    public ReadFile(String filePath) {
        path = filePath;
    }
    public String[] OpenFile() throws IOException {
        FileReader fr = new FileReader(path);
        BufferedReader textReader = new BufferedReader(fr);
        int numberOfLines = readLines();
        String[] textData = new String[numberOfLines];
```

```
        int i;
        for (i = 0; i < numberOfLines; i++) {
            textData[i] = textReader.readLine();
        }
        textReader.close();
        return textData;
    }
    int readLines() throws IOException {
        FileReader fileToRead = new FileReader(path);
        BufferedReader bf = new BufferedReader(fileToRead);
        String aLine;
        int numberOfLines = 0;
        while ((aLine = bf.readLine()) != null) {
            numberOfLines++;
        }
        bf.close();
        return numberOfLines;
    }
}
public class FileData {
    public int neg;
    public int pos;
    public int keno;
    public void compare(String path, String path2) throws Exception {
        String fileName = path;
        String fileName2 = path2;
        try {
            ReadFile file = new ReadFile(fileName);
            ReadFile file2 = new ReadFile(fileName2);
            String[] aryLines = file.OpenFile();
            String[] aryLines2 = file2.OpenFile();
```

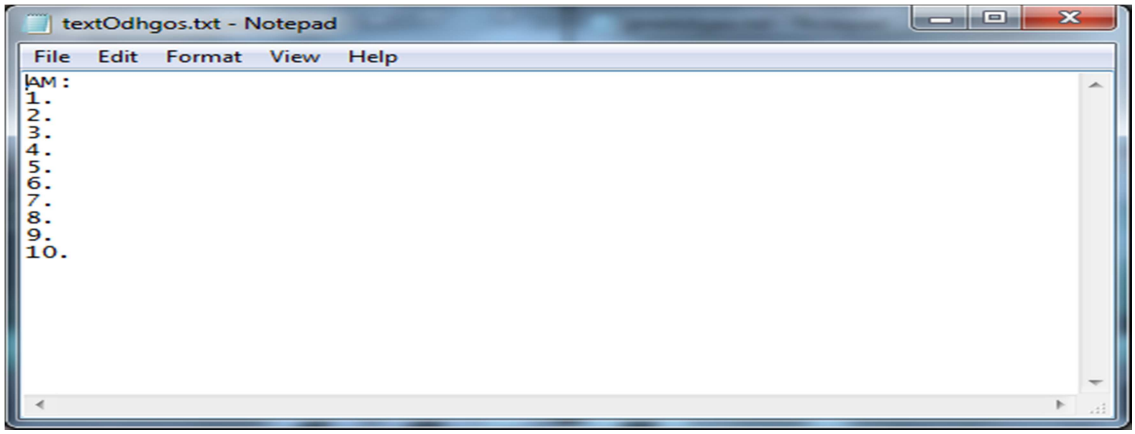
```
            int i;
            for (i = 0; i < aryLines.length; i++) {
               //System.out.println(aryLines[i]);
            }
            for (i = 0; i < aryLines2.length; i++) {
               //System.out.println(aryLines2[i]);
            }
            int j;
            neg = 0;
            pos = 0;
            keno = 0;
            for (j = 1; j < aryLines.length; j++) {
               if ((aryLines[j]).equals(aryLines2[j])) {
                  //System.out.println("To " + (j) + " einai swsto");
                  pos++;
               } else if ((aryLines[j]).equals(j + ".")) {
                  //System.out.println("To " + (j) + " einai keno");
                  keno++;
               } else if ((aryLines[j]).equals(aryLines2[j]) == false && (aryLines[j]).equals(j + ".") == false) {
                  //System.out.println("To " + (j) + " einai lathos");
                  neg++;
               }
            }
            Pattern p = Pattern.compile("-?\\d+");
            Matcher m = p.matcher(aryLines[0]);
            while (m.find()) {
               //System.out.println("To AM einai " + m.group() + " Kai exei " + pos + " swsta kai " + neg + " lathos kai " + keno + " kena ");
               System.out.print("AM: "+ m.group());
            }
         } catch (IOException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
    public int getNeg() {
        return neg;
    }
    public void setNeg(int neg) {
        this.neg = neg;
    }
    public int getPos() {
        return pos;
    }
    public void setPos(int pos) {
        this.pos = pos;
    }
    public int getKeno() {
        return keno;
    }
    public void setKeno(int keno) {
        this.keno = keno;
    }
}
public class Main {
    public static void main(String[] args) throws IOException {
        try {
            UIManager.setLookAndFeel(
                    UIManager.getSystemLookAndFeelClassName());
        } catch (UnsupportedLookAndFeelException e) {
            // handle exception
        } catch (ClassNotFoundException e) {
            // handle exception
```

```
        } catch (InstantiationException e) {
            // handle exception
        } catch (IllegalAccessException e) {
            // handle exception
        }
        Pty test = new Pty();
        String imagePath = "resources/tick.png";
        InputStream imgStream = test.getClass().getResourceAsStream(imagePath);
        BufferedImage myImg = ImageIO.read(imgStream);
        test.setIconImage(myImg);
        test.setTitle("AutoGrade");
        test.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        test.setSize(600, 500);
        test.setVisible(true);//Create and show the GUI.
    }
}
```

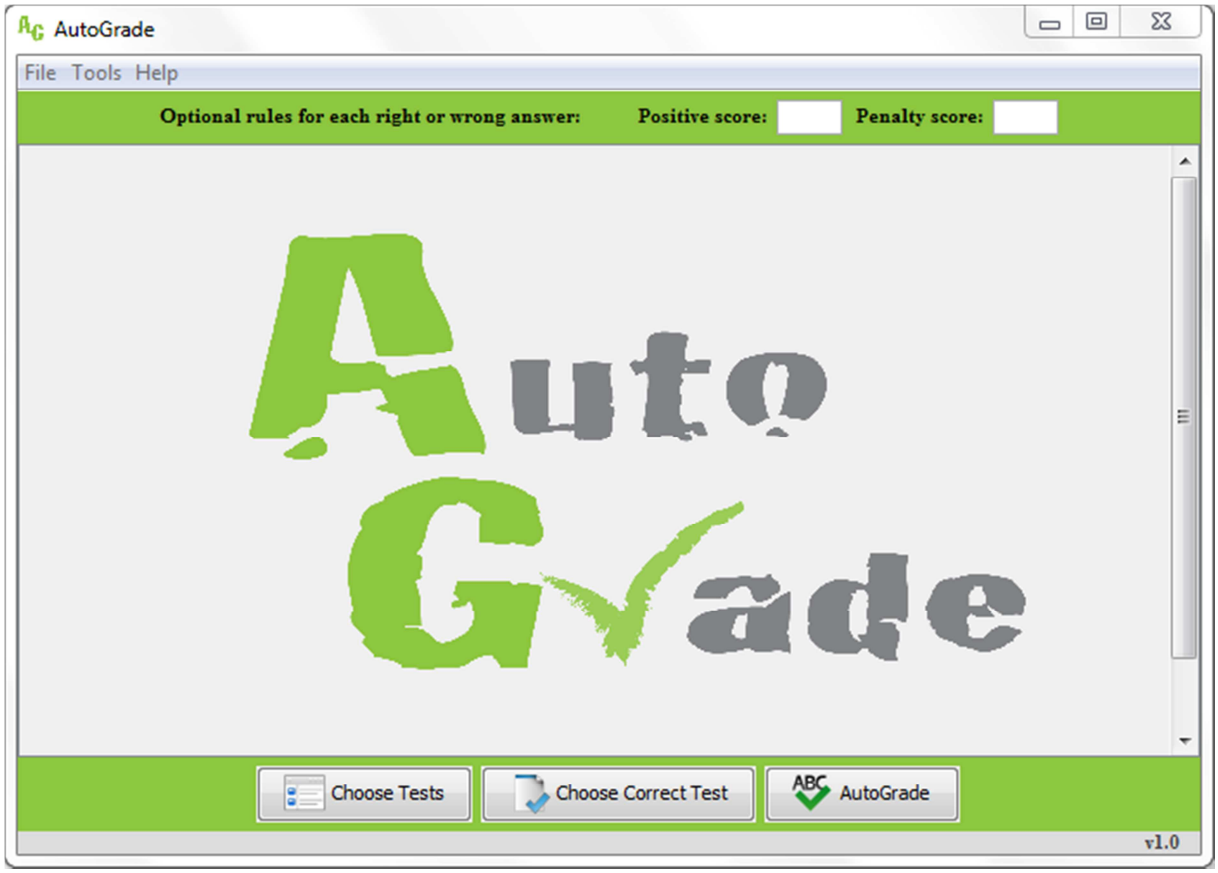
4 ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ

Αυτή τη μορφή πρέπει να έχει το γραπτό που δίνεται στο φοιτητή, σε κόλλα μεγέθους Α4.



Εικόνα 43: Οδηγός μορφής γραπτού φοιτητή

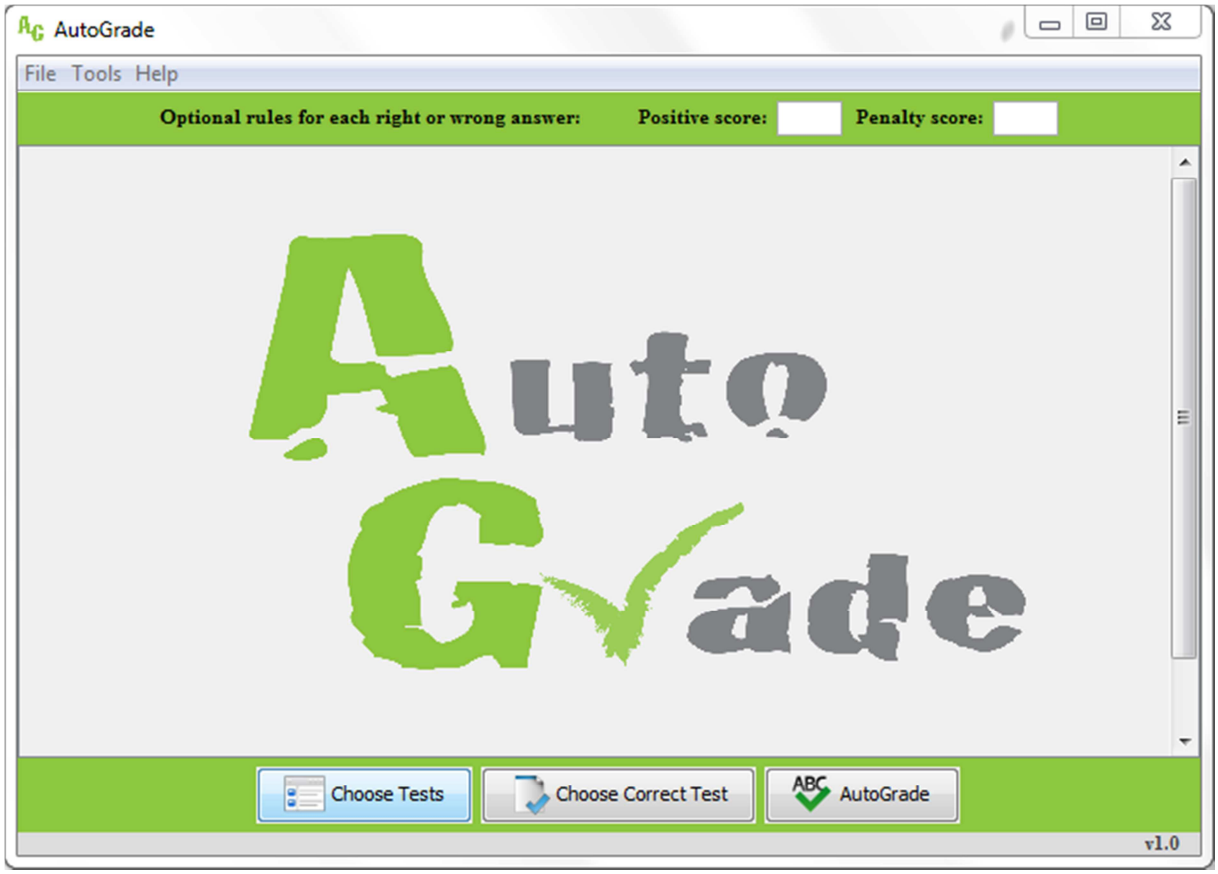
Με την έναρξη της εφαρμογής, εμφανίζεται στον χρήστη το παρακάτω στιγμιότυπο.



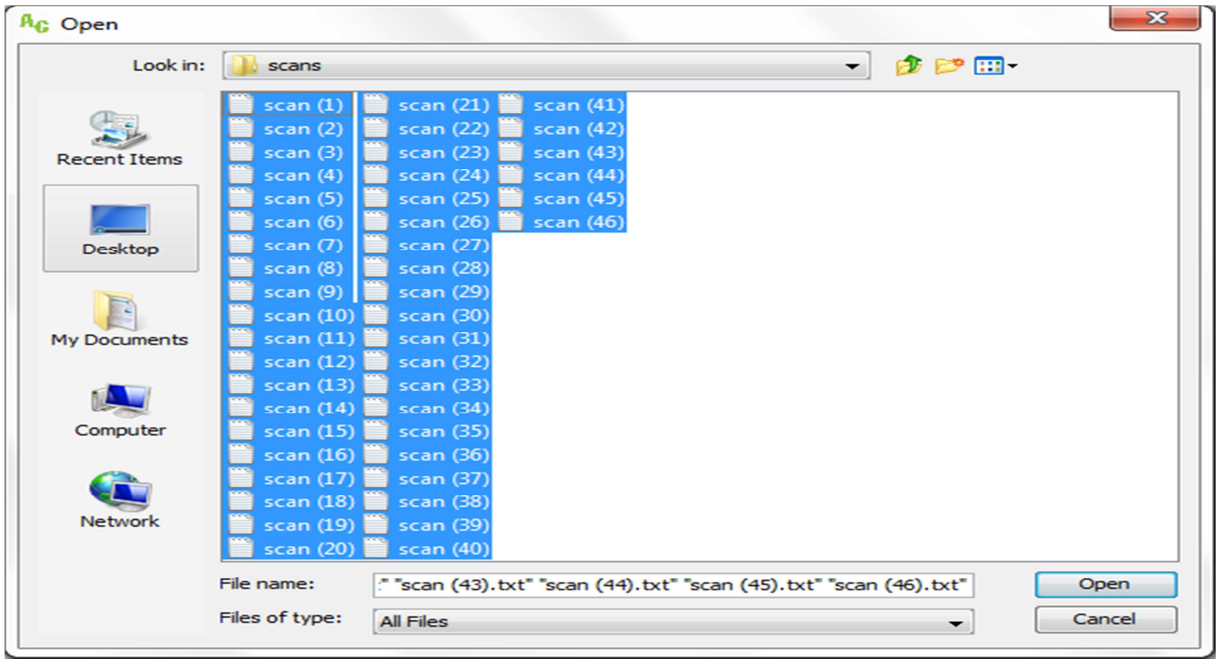
Εικόνα 44: AutoGrade

Θα εξηγήσουμε τον τρόπο χρήσης της εφαρμογής.

Αρχικά, ο χρήστης πρέπει να επιλέξει τα αρχεία με τον παρακάτω τρόπο:

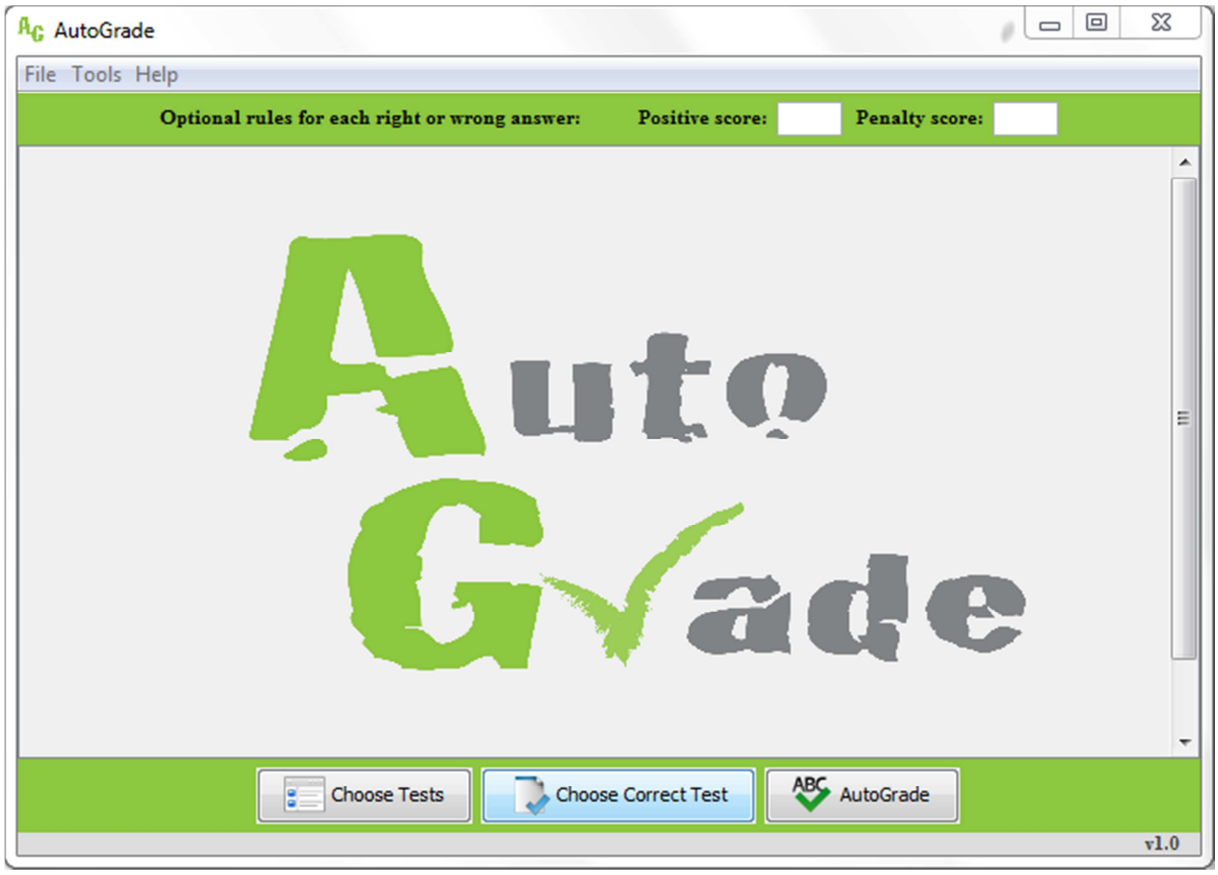


Εικόνα 45: Επιλογή Αρχείων 1- Εγχειρήδιο Χρήσης

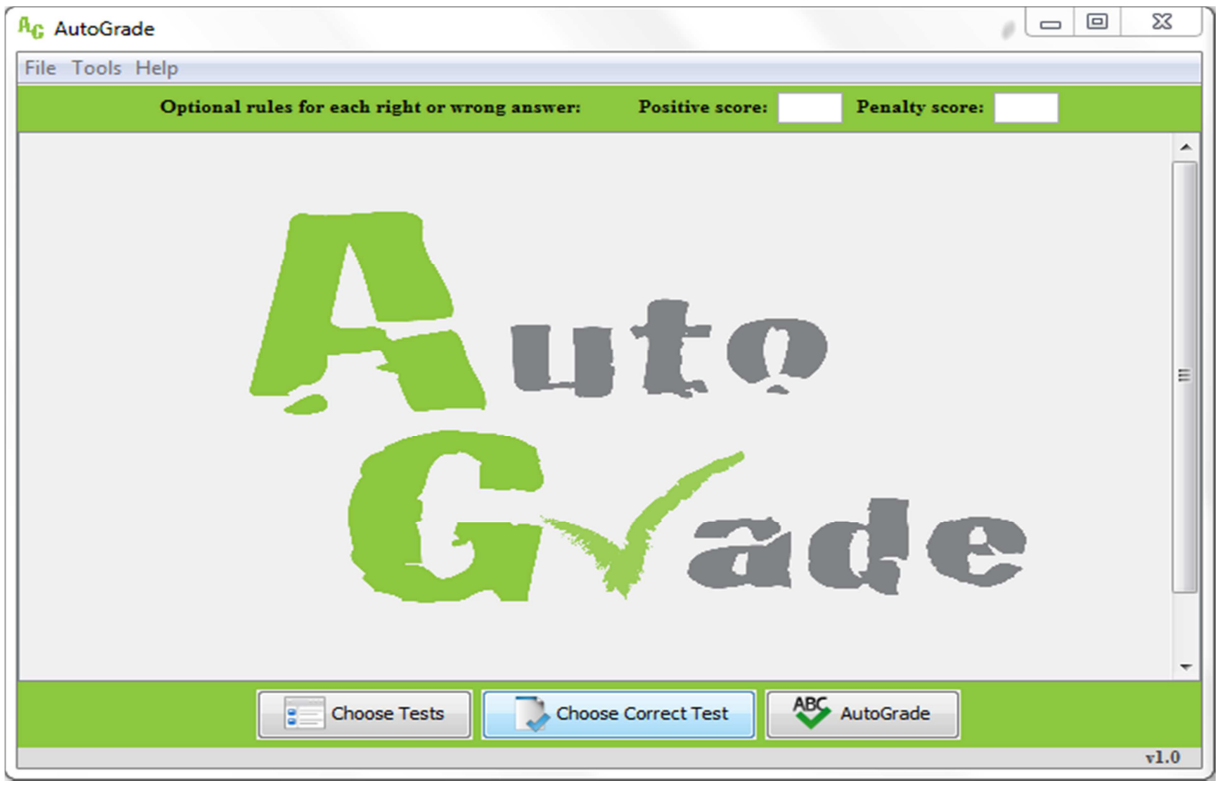


Εικόνα 46: Επιλογή Αρχείων 2

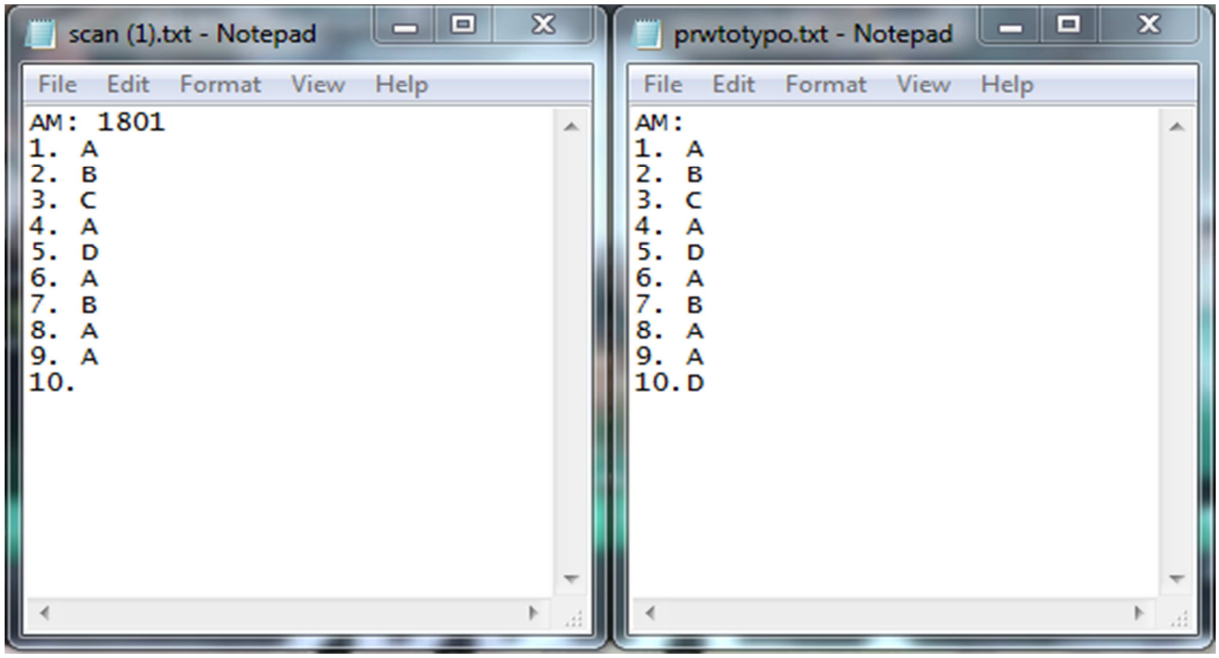
Στη συνέχεια το πρωτότυπο:



Εικόνα 47: Επιλογή Πρωτότυπου Αρχείου

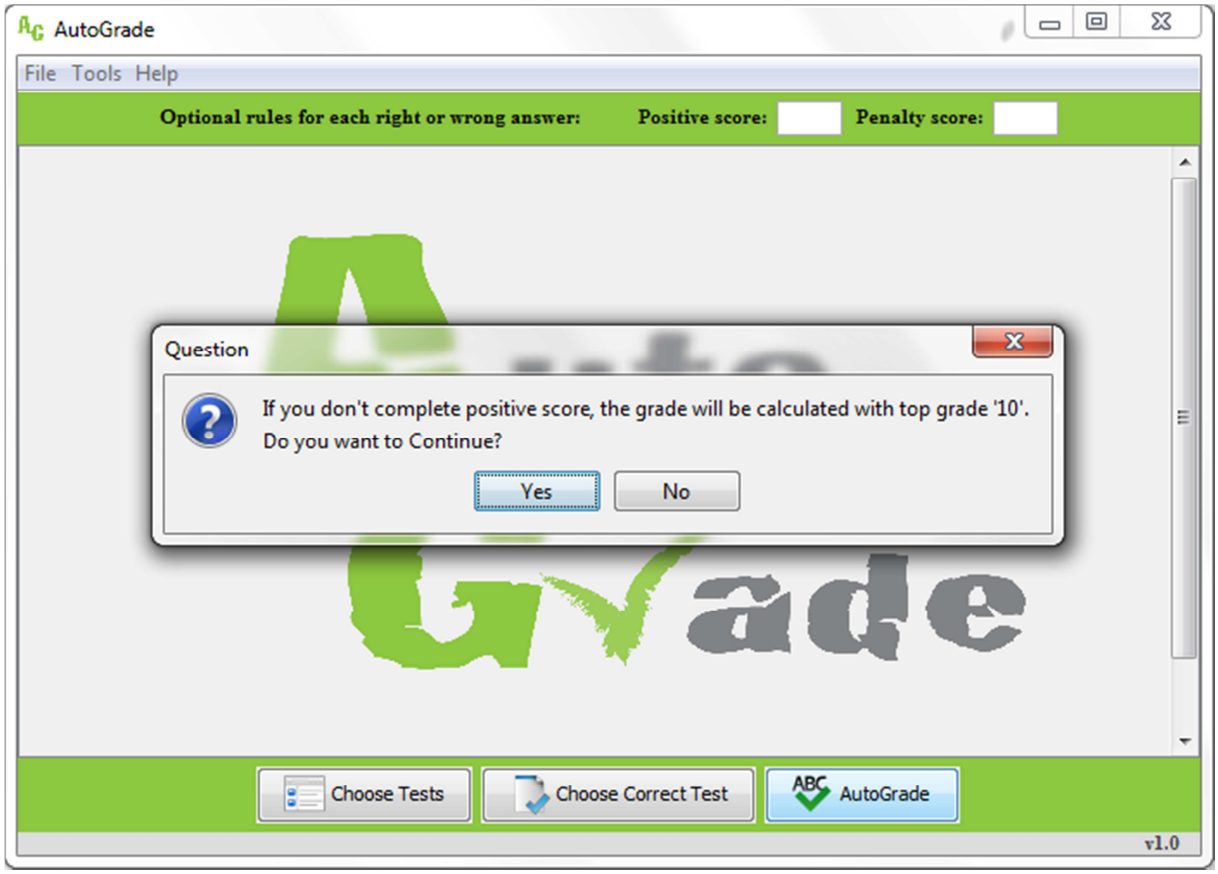


Εικόνα 48: Επιλογή Πρωτότυπου Αρχείου 1

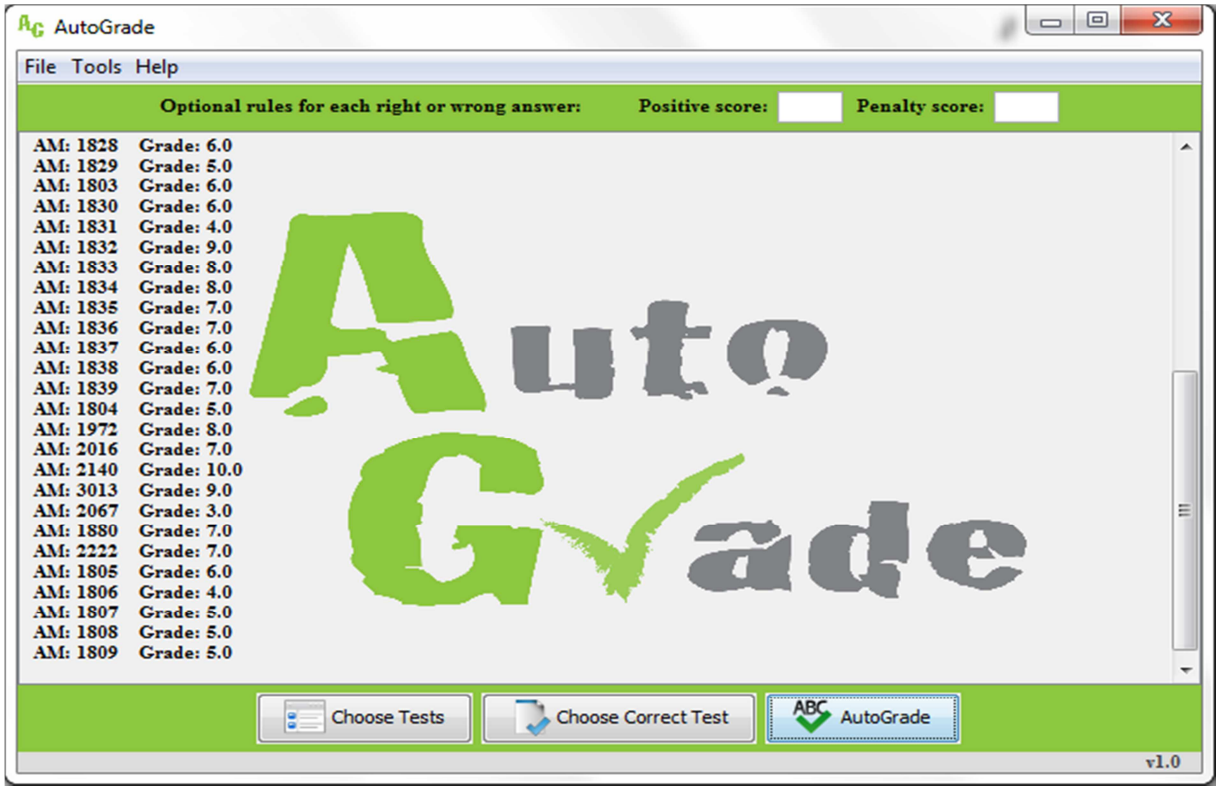


Εικόνα 49: Παράδειγμα γραπτού και πρωτότυπου

Ο χρήστης μπορεί να πατήσει την επιλογή compare και να βαθμολογηθούν τα γραπτά που διάλεξε με άριστα το 10, αφού πρώτα επιβεβαιώσει την επιλογή του.

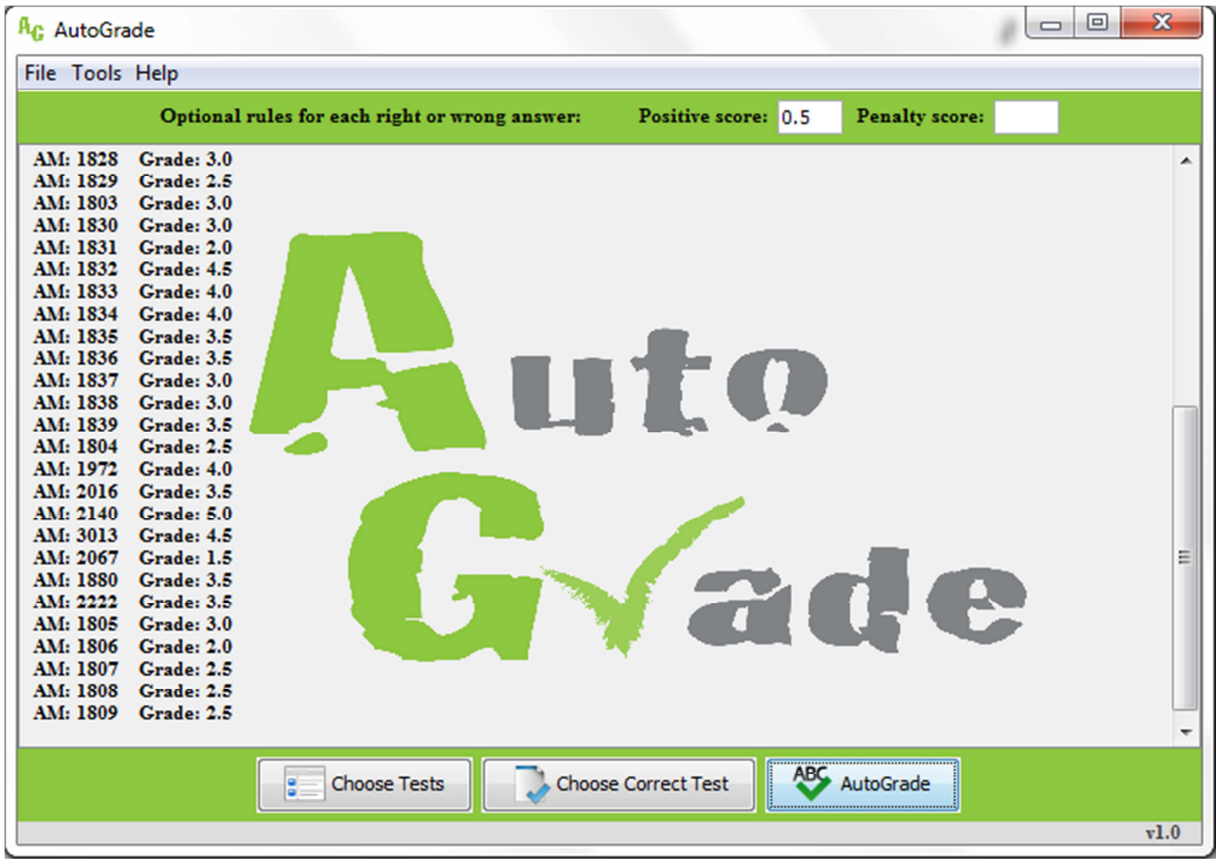


Εικόνα 50: Παράδειγμα σύγκρισης αρχείων με αυτόματη βαθμολόγηση

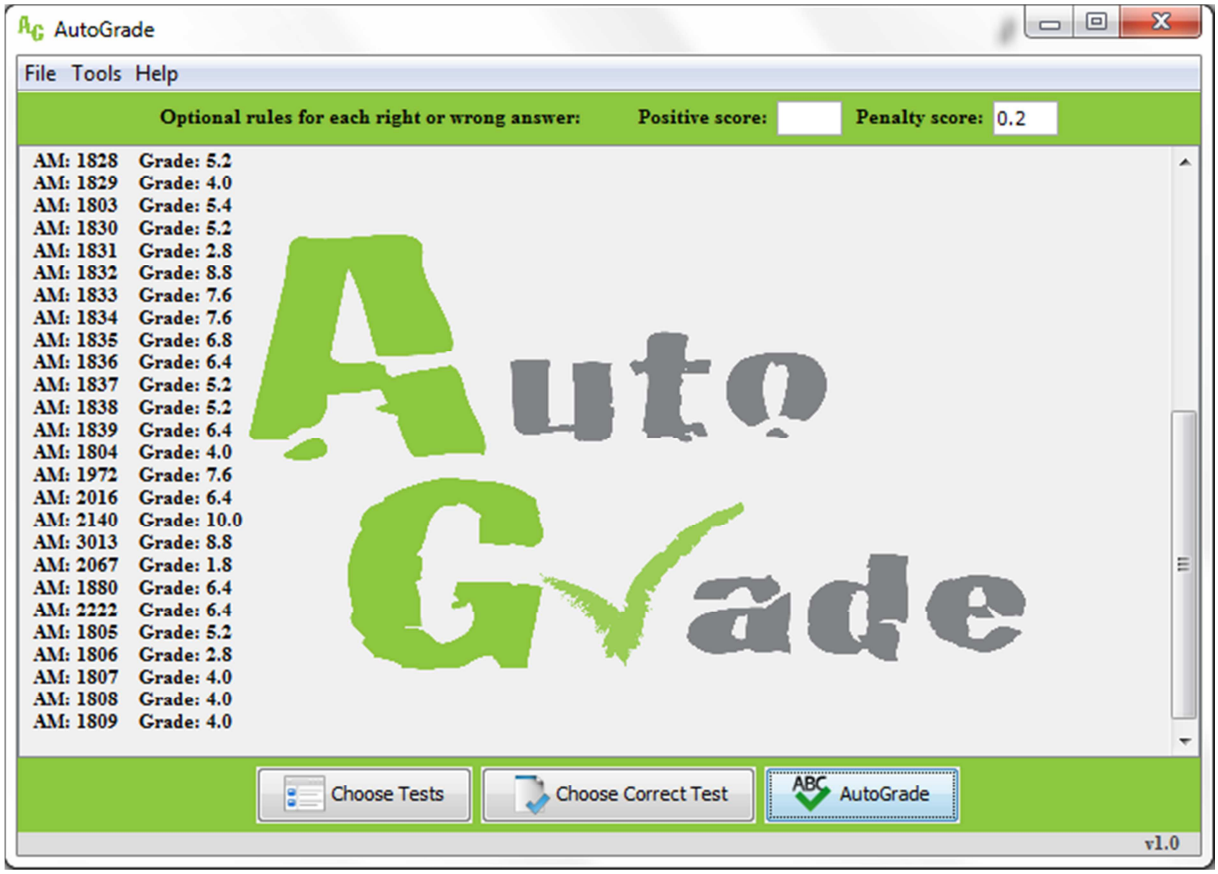


Εικόνα 51: Παράδειγμα σύγκρισης αρχείων με αυτόματη βαθμολόγηση 1

Εναλλακτικά, μπορεί να επιλέξει το score της επιθυμίας του και να πάρει τα ανάλογα αποτελέσματα.

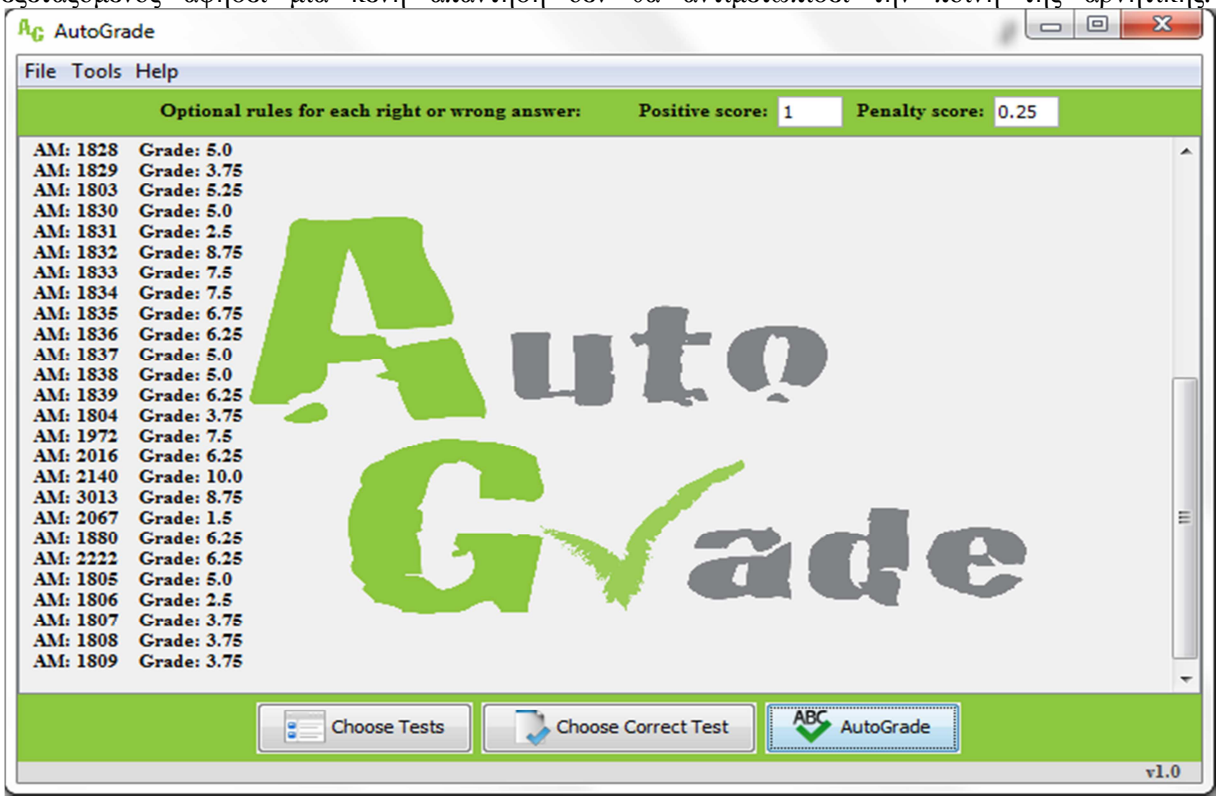


Εικόνα 52: Παράδειγμα σύγκρισης αρχείων με θετική βαθμολόγηση



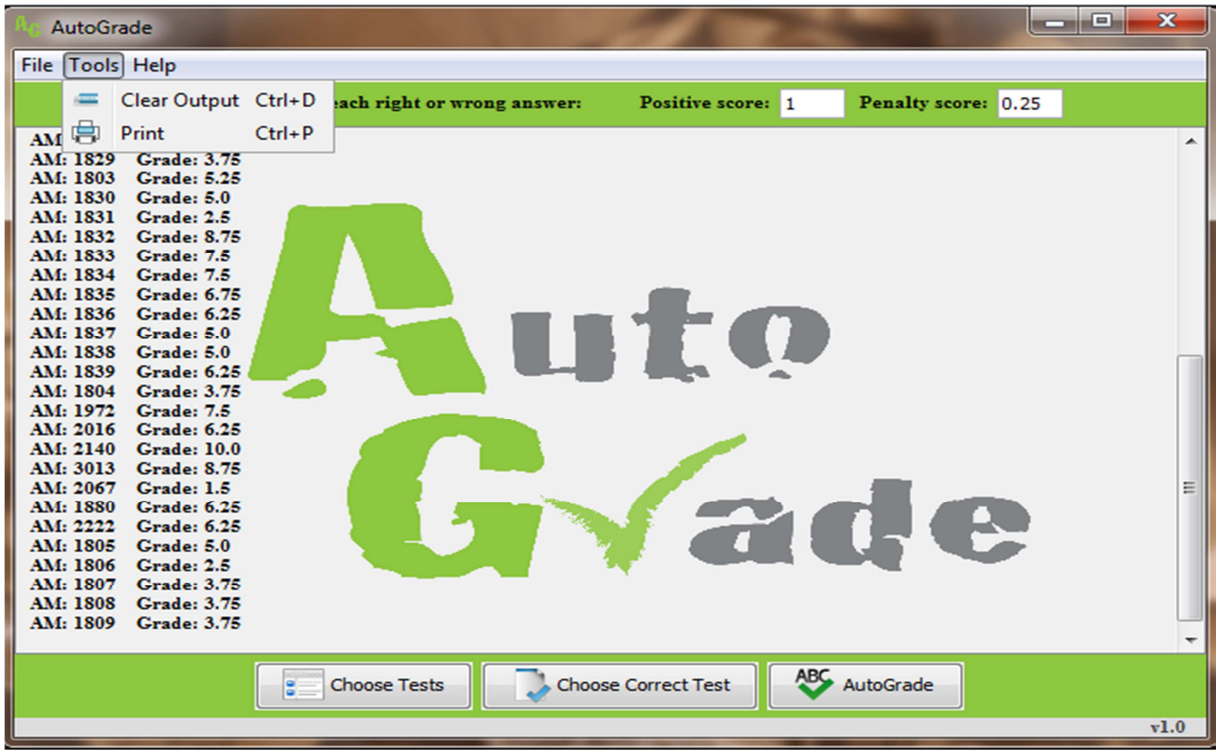
Εικόνα 53: Παράδειγμα σύγκρισης αρχείων με αρνητική βαθμολόγηση

Επίσης υπάρχει η επιλογή της αρνητικής βαθμολόγησης(πάνω δεξιά), με τον κανόνα ότι αν ο εξεταζόμενος αφήσει μία κενή απάντηση δεν θα αντιμετωπίσει την ποινή της αρνητικής.



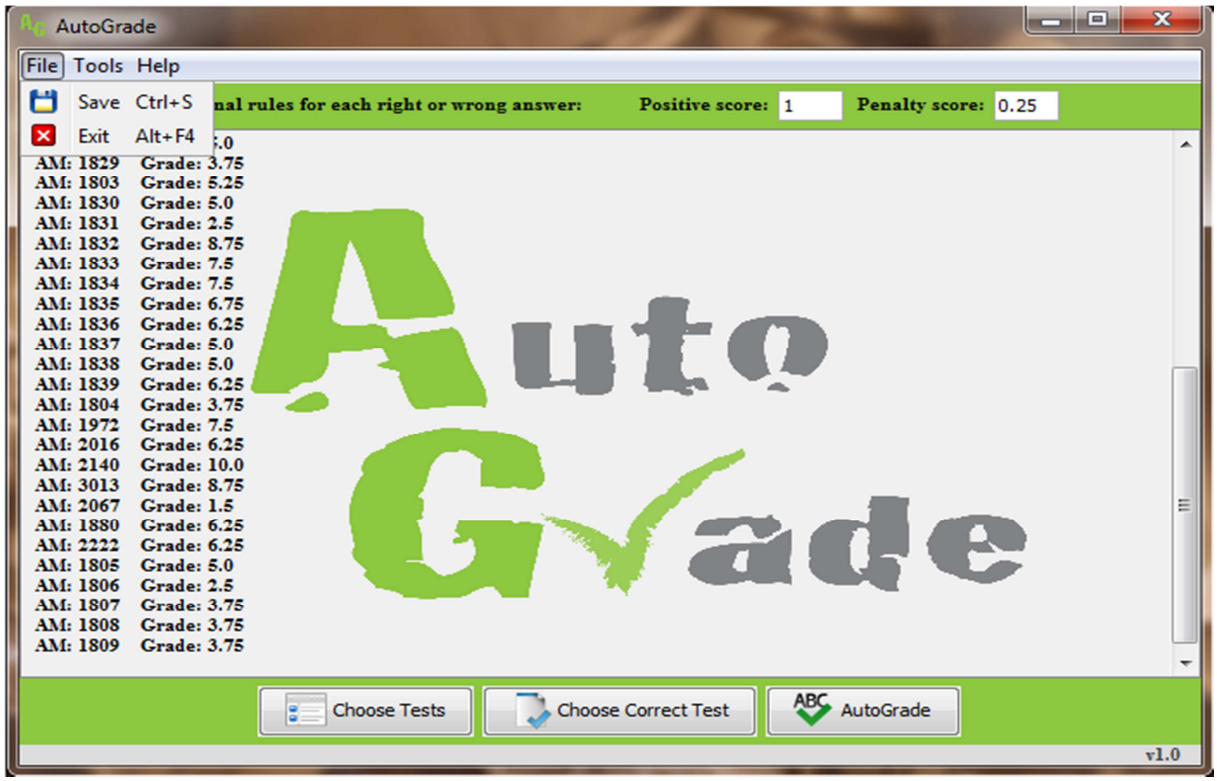
Εικόνα 54: Παράδειγμα σύγκρισης αρχείων με θετική & αρνητική βαθμολόγηση

Αν επιθυμούμε να ξανά συγκρίνουμε τα γραπτά μας για οποιοδήποτε λόγο, αλλάζοντας τους κανόνες και καθαρίζοντας το πρόγραμμά μας, μπορούμε να επιλέξουμε το clear output.

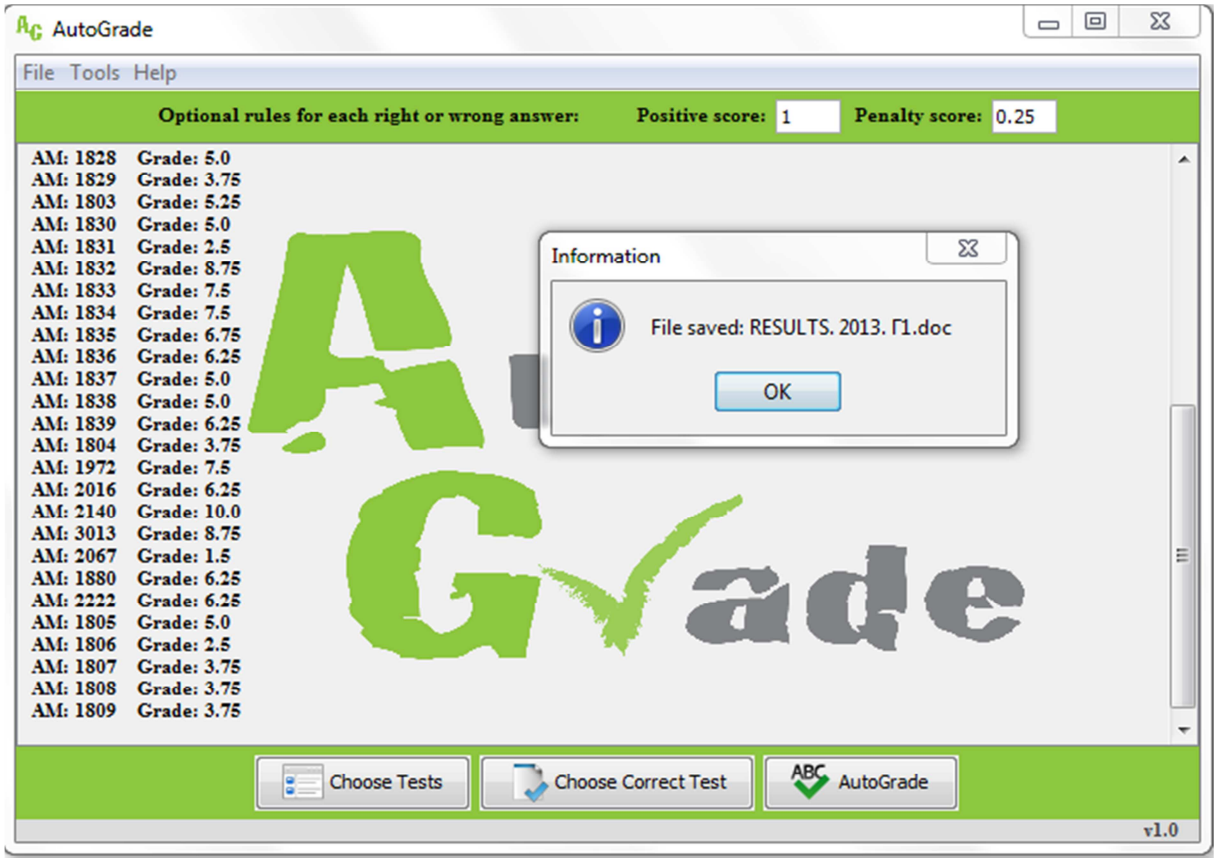


Εικόνα 55: Παράδειγμα tools-clear output

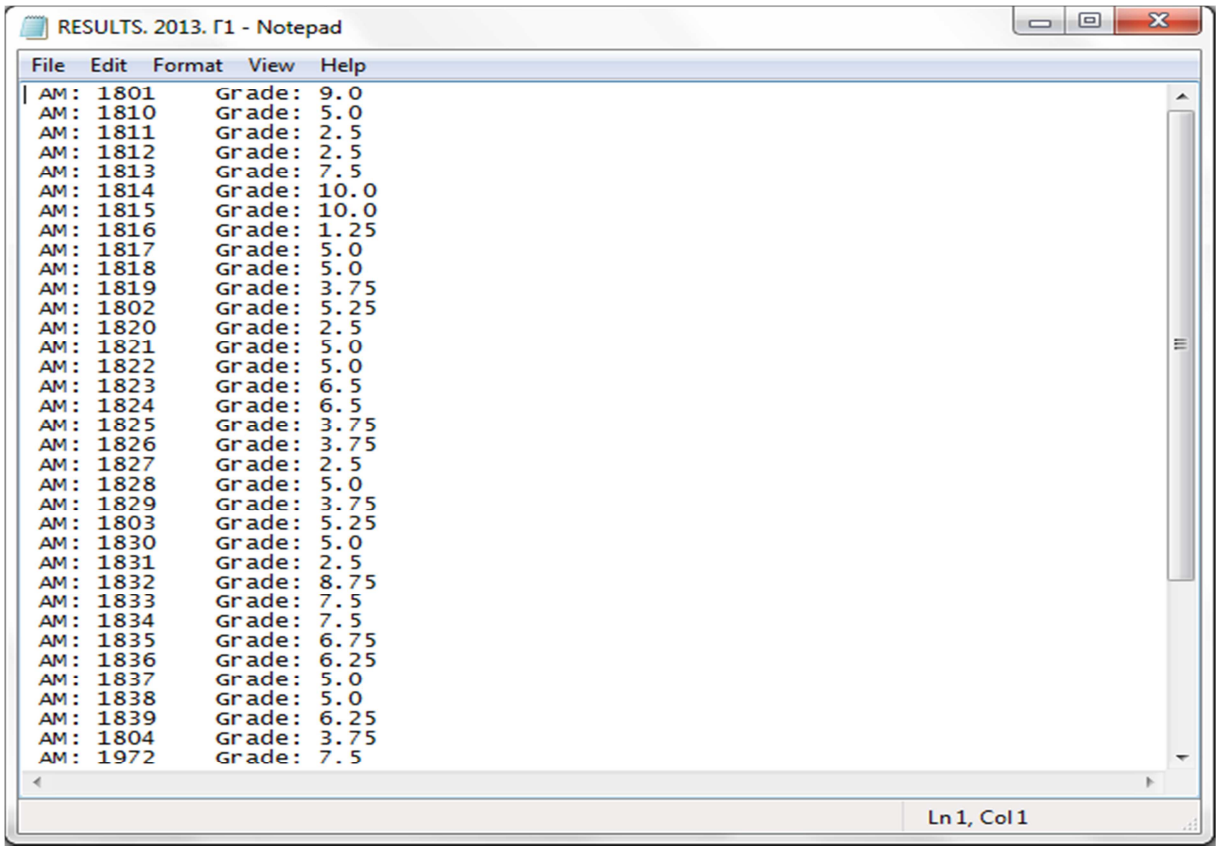
Επίσης έχουμε την επιλογή των exit & save όπου μπορούμε να αποθηκεύσουμε τα αποτελέσματα μας σε ένα ξεχωριστό αρχείο txt.



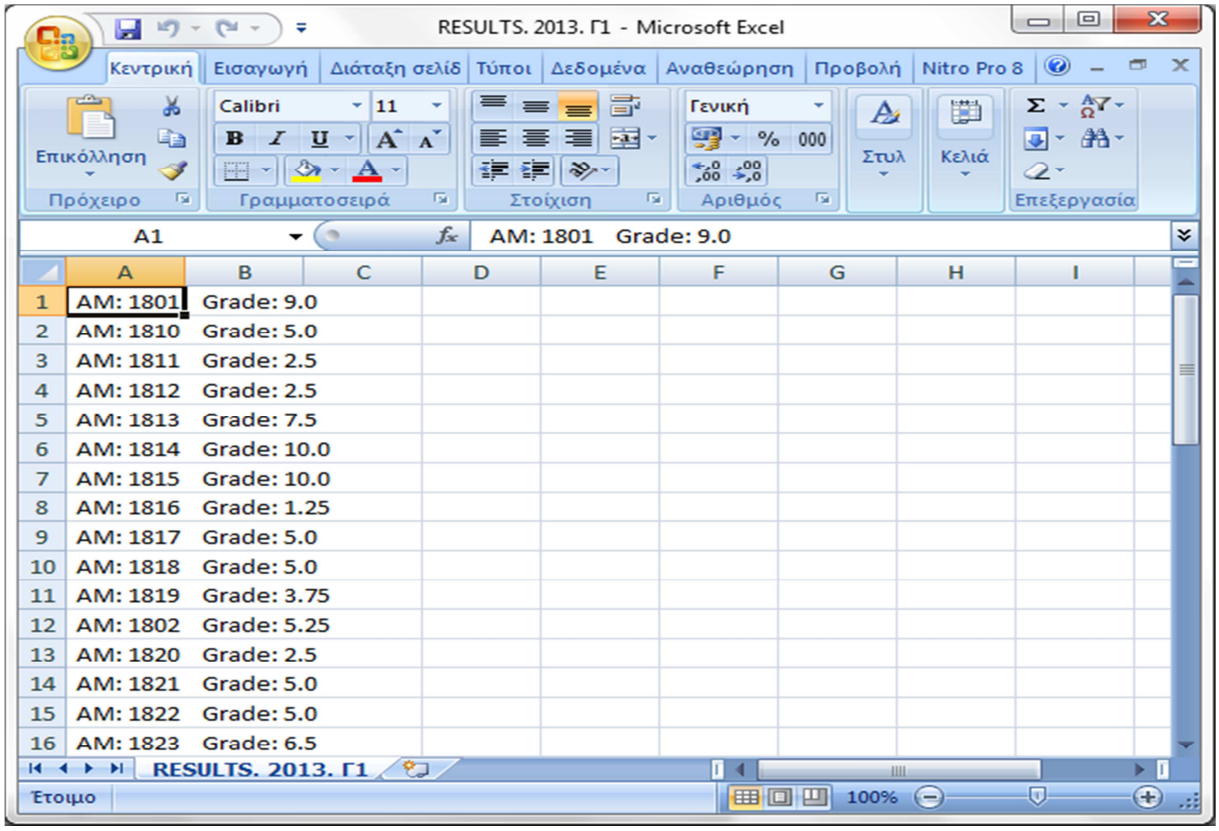
Εικόνα 56: Παράδειγμα tools-exit & save



Εικόνα 57: Παράδειγμα tools-save

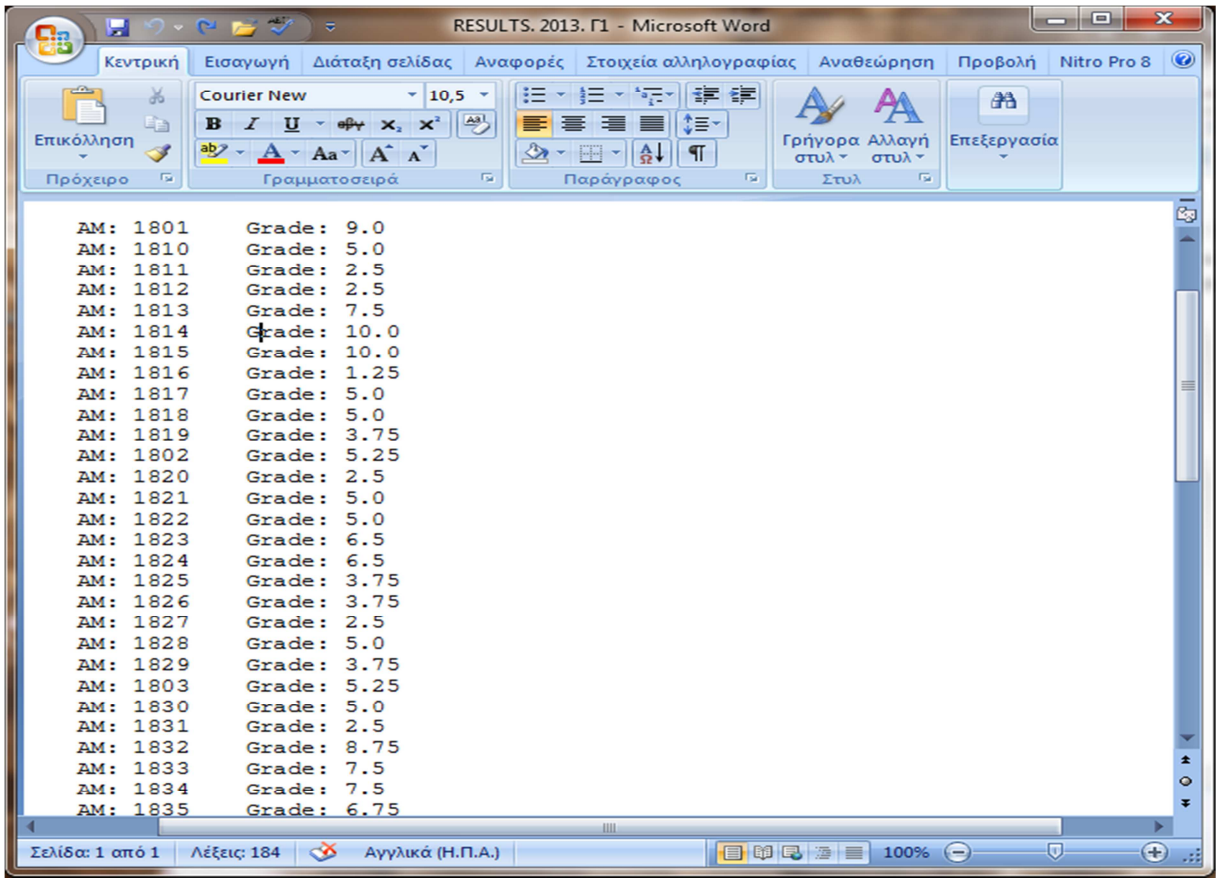


Εικόνα 58: Παράδειγμα αποτελεσμάτων σε txt



	A	B
1	AM: 1801	Grade: 9.0
2	AM: 1810	Grade: 5.0
3	AM: 1811	Grade: 2.5
4	AM: 1812	Grade: 2.5
5	AM: 1813	Grade: 7.5
6	AM: 1814	Grade: 10.0
7	AM: 1815	Grade: 10.0
8	AM: 1816	Grade: 1.25
9	AM: 1817	Grade: 5.0
10	AM: 1818	Grade: 5.0
11	AM: 1819	Grade: 3.75
12	AM: 1802	Grade: 5.25
13	AM: 1820	Grade: 2.5
14	AM: 1821	Grade: 5.0
15	AM: 1822	Grade: 5.0
16	AM: 1823	Grade: 6.5

Εικόνα 59: Παράδειγμα αποτελεσμάτων σε Excel




```
AM: 1801    Grade: 9.0
AM: 1810    Grade: 5.0
AM: 1811    Grade: 2.5
AM: 1812    Grade: 2.5
AM: 1813    Grade: 7.5
AM: 1814    Grade: 10.0
AM: 1815    Grade: 10.0
AM: 1816    Grade: 1.25
AM: 1817    Grade: 5.0
AM: 1818    Grade: 5.0
AM: 1819    Grade: 3.75
AM: 1802    Grade: 5.25
AM: 1820    Grade: 2.5
AM: 1821    Grade: 5.0
AM: 1822    Grade: 5.0
AM: 1823    Grade: 6.5
AM: 1824    Grade: 6.5
AM: 1825    Grade: 3.75
AM: 1826    Grade: 3.75
AM: 1827    Grade: 2.5
AM: 1828    Grade: 5.0
AM: 1829    Grade: 3.75
AM: 1803    Grade: 5.25
AM: 1830    Grade: 5.0
AM: 1831    Grade: 2.5
AM: 1832    Grade: 8.75
AM: 1833    Grade: 7.5
AM: 1834    Grade: 7.5
AM: 1835    Grade: 6.75
```

Εικόνα 60: Παράδειγμα αποτελεσμάτων σε Word

5 ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Με την ολοκλήρωση της Πτυχιακής Εργασίας, δημιουργήθηκε μία εφαρμογή που απευθύνεται κυρίως σε εκπαιδευτικούς. Το πρόγραμμα είναι ικανό να μειώσει το χρόνο διόρθωσης γραπτών πολλαπλής επιλογής και ταυτόχρονα να αυξήσει το χρόνο των χρηστών. Πρόκειται για μία λειτουργική εφαρμογή γεγονός που φαίνεται από την ευκολία στην εγκατάσταση και τη χρήση. Ο χρήστης για να είναι σε θέση να χρησιμοποιήσει την εφαρμογή απαιτείται μόνο ο Η/Υ του να διαθέτει περιβάλλον Java. Ενώ για να την επεξεργαστεί δεν απαιτείται ιδιαίτερη εκπαίδευση καθώς περιλαμβάνει απλές επιλογές, κατανοητές για όλους.

Σε προσωπικό επίπεδο, η εκπόνηση της συγκεκριμένης Πτυχιακής Εργασίας βελτίωσε σε μεγάλο βαθμό τις γνώσεις μας στον Αντικειμενοστραφή Προγραμματισμό και την ικανότητά μας να επιλύουμε προβλήματα τέτοιου τύπου. Επίσης, αναπτύξαμε χαρακτηριστικά όπως η ευρηματικότητα, η συνεργασία, κ.α.

5.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

Δημιουργήθηκε η εφαρμογή όπως είχε προσχεδιαστεί, με μια μικρή απόκλιση από το αρχικό πλάνο λόγω της μεγάλης πιθανότητας λάθους που παρουσιάζουν τη δεδομένη χρονική περίοδο οι OCR εφαρμογές. Σχεδιάστηκε το πρόγραμμα με γνώμονα το καθηγητή για την εύκολη χρήση του προγράμματος και την ταχύτερη διόρθωση των γραπτών. Έτσι, ο καθηγητής έχει τη δυνατότητα να πραγματοποιήσει τα εξής:

- υπολογισμός των βαθμών των εξεταζόμενων και αντιστοίχηση αυτών με τα ΑΜ τους
- προαιρετική επιλογή θετικής ή/και αρνητικής βαθμολόγησης
- προαιρετική εκτύπωση των αποτελεσμάτων

προαιρετική εξαγωγή των αποτελεσμάτων σε ξεχωριστό αρχείο (π.χ. txt).

5.2 ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ ΚΑΙ ΕΠΕΚΤΑΣΕΙΣ

Όπως έχει ήδη αναφερθεί η εφαρμογή «Αυτόματη Διόρθωση» δεν είναι ακόμη έτοιμη για χρήση καθώς στην αγορά δεν υπάρχει OCR κατάλληλη για να ικανοποιήσει τις απαιτήσεις του συστήματος, αν και η Google βρίσκεται στο σωστό δρόμο. Συνεπώς, μελλοντικά θα πρέπει να δημιουργηθεί μία OCR που θα είναι σε θέση να μετατρέψει πλήρως την εικόνα σε text χωρίς να παρουσιάζει κανένα λάθος.

Επίσης, μία μελλοντική επέκταση του προγράμματος θα μπορούσε να είναι η ανάπτυξή του σε ένα λογισμικό κινητής εφαρμογής, όπως είναι το λογισμικό android, όπου θα μπορεί ο φοιτητής μετά το τέλος της εξέτασης, επισυνάπτοντας τη φωτογραφία του γραπτού του σε μία βάση δεδομένων του καθηγητή (online), να λαμβάνει άμεσα τα αποτελέσματα. Στην πράξη, για να πραγματοποιηθεί αυτό το update δε χρειάζεται εκ νέου δημιουργία του προγράμματος, αλλά κάποιες μικρές αλλαγές στον κώδικα ώστε να γίνει συμβατό με κινητή συσκευή.

ΠΗΓΕΣ

el.wiki-pedia.org/wiki/Java

en.wikipedia.org/wiki.NetBeans

el.wikipedia.org/wiki/Οπτική_Αναγνώριση_Χαρακτήρων

el.wikipedia.org/wiki/Αντικειμενοστραφής_προγραμματισμός

en.wikipedia.org/wiki/File_comparison

http://docs.oracle.com/javase/tutorial/uiswing/components/index.html

http://stackoverflow.com/

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Ιακωβίδης Δ. (χ.χ.). [Online]. Available: www.inf.teilam.gr/OLD/java/Java_Lecture1_OVR.pdf

[2] Βεσκούκης Β. και Κουτουμάνος Α. (2000). Εισαγωγή στη γλώσσα Java. Πανεπιστήμιο Πειραιώς Τμήμα Τεχνολογικής Εκπαίδευσης. [Online]. Available: users.sof/lab.ece.ntua.gr/~bxb/courses/unipi2001_te/00-CourseNotes/031-OO&Java/TE031-2.pdf

[3] Εθνικό Μετσόβειο Πολυτεχνείο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ. (χ.χ.) Εισαγωγή στη γλώσσα προγραμματισμού Java. [Online]. Available: www.ebooks4greeks.gr/down/oads/Pliroforiki/Glosses.program./Java_Downloaded_from_eBooks4Greeks.gr.pdf

[4] Σιακαβέλλα Ε. Η. (2006). Συστήματα Επιχειρησιακής Μοντελοποίησης και Αναπαράστασης Αξιολόγηση και Εφαρμογές. Εθνικό Μετσόβειο Πολυτεχνείο Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων. Αθήνα.

[5] UML notes (n.d.). [Online]. Available: www.icsd.aegean.gr/kotis/softTech06/UMLnotes.pdf

[6] Βιδάκης Ν. (2010). Αντικειμενοστρεφής Προγραμματισμός. Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων. Ηράκλειο, Κρήτης.

[7] Deitel P. and Deteil H. (2010). JAVA Προγραμματισμός. 8η έκδοση. Γκιούρδας Μ

ΠΑΡΑΡΤΗΜΑ Α'

Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

Τίτλος

Σχεδίαση και ανάπτυξη εφαρμογής υπολογιστικής και συγκριτικής ικανότητας για αυτόματη διόρθωση γραπτών πολλαπλής επιλογής.

Σπουδαστές: Καλαμουδάκος Κωνσταντίνος (ΑΜ: 1880)
Μανωλαράκης Γεώργιος (ΑΜ: 2049)

Εισηγητής: Παπαδάκης Νικόλαος

Εισαγωγή

- **Σκοπός**: ανάπτυξη λογισμικού, που θα μπορεί να χρησιμοποιηθεί για την ταχύτερη και ευκολότερη διόρθωση πολλαπλών διαγωνισμάτων (πολλαπλής επιλογής).

- **Στόχος**: χρήση της εφαρμογής από το ΑΤΕΙ Ηρακλείου Κρήτης και συνεπώς από το εκπαιδευτικό προσωπικό του. Επιθυμούμε μακροπρόθεσμα να γίνει ένα καθημερινό λειτουργικό εργαλείο στα χέρια του μειωμένου και ελλιπές προσωπικού του ΑΤΕΙ, ώστε να επιτύχουμε τελικά τη μείωση του επί του παρόντος υπέρογκου φόρτου εργασίας τους και την πιο γρήγορη πληροφόρηση των αποτελεσμάτων των φοιτητών.

Βασικές λειτουργίες εφαρμογής

ΟΙ ΔΥΝΑΤΟΤΗΤΕΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ AUTOGRADE (AG) ΕΙΝΑΙ:

- ΣΥΓΚΡΙΣΗ & ΔΙΟΡΘΩΣΗ ΓΡΑΠΤΩΝ
- ΕΞΑΓΩΓΗ ΒΑΘΜΟΛΟΓΙΩΝ
- ΑΠΟΘΗΚΕΥΣΗ & ΕΚΤΥΠΩΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Μέθοδος ανάλυσης & ανάπτυξης

- ΜΟΡΦΟΠΟΙΗΣΗ ΓΡΑΠΤΩΝ ΣΕ ΕΙΚΟΝΕΣ (SCANNER)
- ΜΟΡΦΟΠΟΙΗΣΗ ΕΙΚΟΝΩΝ ΣΕ ΑΡΧΕΙΟ ΚΕΙΜΕΝΟΥ (OCR)
- ΕΙΣΑΓΩΓΗ ΑΡΧΕΙΩΝ (ΤΕΛΙΚΗΣ ΜΟΡΦΗΣ) ΣΤΟ ΠΡΟΓΡΑΜΜΑ
- ΣΥΓΚΡΙΣΗ & ΕΞΑΓΩΓΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ (ΧΡΗΣΗ OPP)

Αυτόματη Διόρθωση

- *Σαν γενικός όρος είναι μια αυτόματη σύγκριση μεταξύ διαφόρων ειδών αρχείων.*
- *Συγκρίνει το περιεχόμενο των αρχείων.*
- *Εντοπίζει τα κοινά περιεχόμενά τους και τις διαφορές τους.*
- *Το αποτέλεσμα της σύγκρισης, που συχνά αποκαλείται «diff», είναι δυνατό να παρουσιάζεται σε ένα γραφικό περιβάλλον χρήστη ή να χρησιμοποιηθεί ως μέρος διεργασιών σε networks, file systems, ή revision control.*

Τεχνολογίες και εργαλεία ανάπτυξης 1/6

- OCR
- Unified Modeling Language
- Αντικειμενοστραφής Προγραμματισμός
- NetBeans Platform

Τεχνολογίες και εργαλεία ανάπτυξης 2/6

OCR

Η **Οπτική Αναγνώριση Χαρακτήρων** (**O**ptical **C**haracter **R**ecognition) ή αλλιώς Αυτόματη Αναγνώριση Χαρακτήρων Κειμένου, ονομάζεται η διαδικασία μετατροπής σαρωμένων εικόνων χειρογράφων ή έντυπων κειμένων σε κείμενο αναγνώσιμο από Η/Υ.

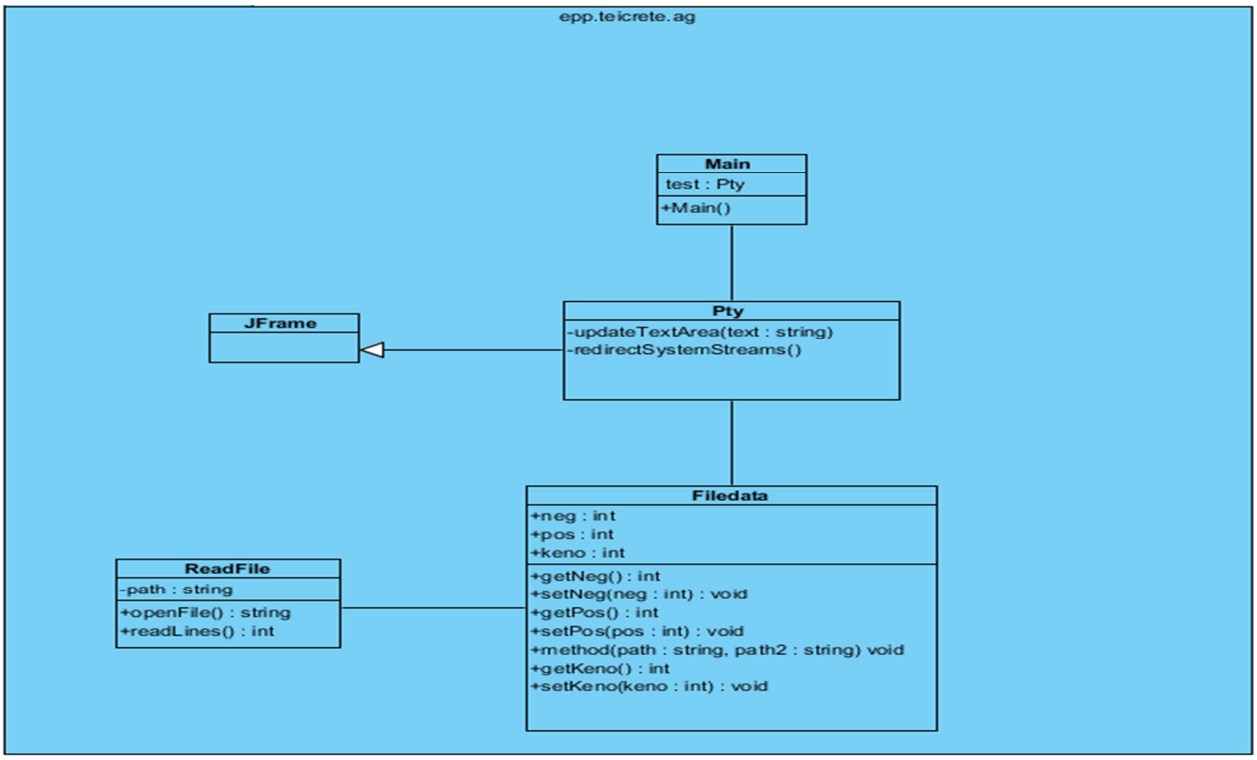
Η Οπτική Αναγνώριση Χαρακτήρων καθιστά εφικτή την εκ νέου επεξεργασία του κειμένου, αποφεύγοντας την δακτυλογράφηση του από την αρχή.

Τεχνολογίες και εργαλεία ανάπτυξης 3/6

Unified Modeling Language (UML)

Η **Unified Modeling Language** (UML) είναι μία γλώσσα που χρησιμοποιείται:

- για προδιαγραφές, αναπαράσταση με οπτικό τρόπο (visualizing), δημιουργία και τεκμηρίωση των τμημάτων των συστημάτων λογισμικού,
- καθώς και για μοντελοποίηση εταιρικών και άλλων συστημάτων που δεν αφορούν λογισμικό.



Τεχνολογίες και εργαλεία ανάπτυξης 5/6

Αντικειμενοστραφής Προγραμματισμός (OPP)

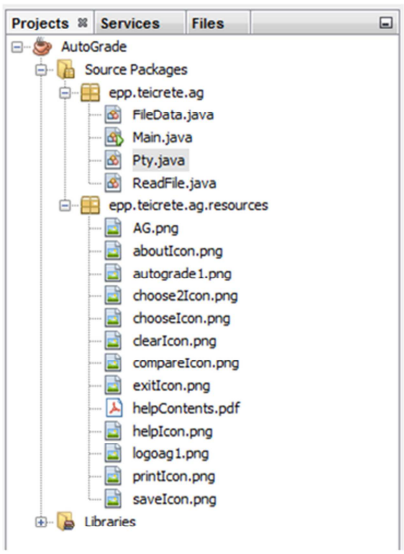
- Ο ***αντικειμενοστραφής προγραμματισμός*** (OOP-Object Oriented Programming), είναι μια προγραμματιστική φιλοσοφία όπως και ο προστακτικός ή ο λογικός προγραμματισμός. Σύμφωνα με τον OPP ένα πρόγραμμα δεν αποτελείται από τα δεδομένα και τον κώδικα που τα επεξεργάζεται αλλά από ***αντικείμενα (objects)*** τα οποία εμπεριέχουν τα δεδομένα και τα οποία ανταλλάσουν μεταξύ τους πληροφορίες και μηνύματα προκειμένου να επιτευχθεί ο στόχος του προγράμματος.
- Κεντρική ιδέα στον αντικειμενοστραφή προγραμματισμό είναι η ***κλάση (class)***, μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων, είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών, εννοιολογικών αντικειμένων, σε ένα περιβάλλον προγραμματισμού.

Τεχνολογίες και εργαλεία ανάπτυξης 6/6

NetBeans Platform

Το NetBeans είναι ένα ολοκληρωμένο, εξελιγμένο περιβάλλον στο οποίο αναπτύσσεται κυρίως Java γλώσσα, αλλά και άλλες γλώσσες, πιο συγκεκριμένα οι PHP, C/C++ και HTML5.

Ανάπτυξη Εφαρμογής «AutoGrade»



Περιγραφή χρήσης της εφαρμογής

- ΕΠΙΛΟΓΗ ΤΩΝ ΑΡΧΕΙΩΝ
- ΠΡΟΑΙΡΕΤΙΚΟΙ ΚΑΝΟΝΕΣ ΒΑΘΜΟΛΟΓΗΣΗΣ
- ΣΥΓΚΡΙΣΗ & ΑΠΟΘΗΚΕΥΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Τέλος Παρουσίασης

Σας Ευχαριστώ!

ΠΑΡΑΡΤΗΜΑ Β'

ΠΕΡΙΛΗΨΗ ΔΗΜΟΣΙΕΥΣΗΣ

Σχεδίαση και ανάπτυξη εφαρμογής υπολογιστικής και συγκριτικής ικανότητας για αυτόματη διόρθωση γραπτών πολλαπλής επιλογής.

Καλαμουδάκος Κ. , Μανωλαράκης Γ. & Παπαδάκης Ν.

Ανώτατο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων, Ηράκλειο Κρήτης, Ελλάδα.

Σύνοψη

Σκοπός της παρούσας Πτυχιακής είναι η δημιουργία μιας αξιόπιστης εφαρμογής η οποία με αποτελεσματικότητα, ευκολία και ταχύτητα είναι σε θέση να διορθώνει αυτόματα ένα μεγάλο αριθμό γραπτών πολλαπλής επιλογής, να εξάγει τους βαθμούς και να τους αντιστοιχεί με τους Αριθμούς Μητρώου των εξεταζόμενων.

Η βασική τεχνολογία που χρησιμοποιήθηκε, είναι η πλατφόρμα NetBeans. Η εφαρμογή αναπτύχθηκε και υλοποιήθηκε με τη χρήση Αντικειμενοστραφούς Προγραμματισμού.

Λέξεις κλειδιά: Αντικειμενοστραφής Προγραμματισμός (OPP), OCR, Αυτόματη βαθμολόγηση (AutoGrade), Java.

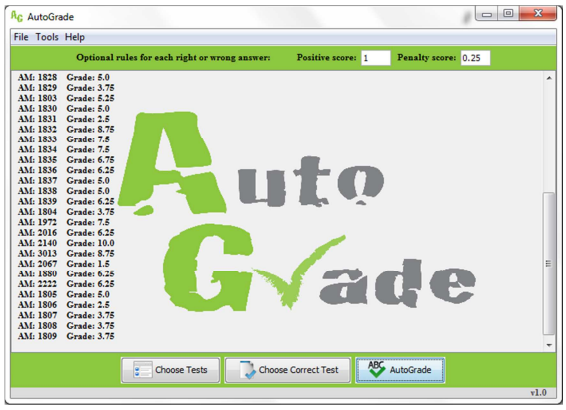
Εισαγωγή

Η πτυχιακή εργασία, όσον αφορά τον εκπαιδευτικό της χαρακτήρα, είναι ένα σημαντικό κομμάτι των σπουδών, διότι παρέχει τη δυνατότητα στον σπουδαστή να μελετήσει, να αναλύσει και να αναπτύξει διάφορα θέματα υπό την καθοδήγηση του εισηγητή της πτυχιακής. Ο σπουδαστής από τη διαδικασία αυτήν αποκτά εις βάθος γνώσεις για το αντικείμενο, και αποκομίζει εφόδια που θα του είναι χρήσιμα στην περαιτέρω επιστημονική και επαγγελματική του πορεία.

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάλυση, σχεδίαση και υλοποίηση μιας εφαρμογής αυτόματης διόρθωσης πολλαπλής επιλογής διαγωνισμάτων. Η συγκεκριμένη εφαρμογή αντλεί τα αρχεία που έχουν μετατραπεί από εικόνα σε κείμενο (txt), μέσω της OCR, και τα συγκρίνει με ένα πρωτότυπο αρχείο για να βρει διαφορές. Στη συνέχεια, αφού έχει εντοπίσει τα λάθη εξάγει τη βαθμολογία των γραπτών και τα αντιστοιχεί με τους Αριθμούς Μητρώων (ΑΜ) των εξεταζόμενων. Για την υλοποίηση όλων των παραπάνω χρησιμοποιήθηκε η πλατφόρμα NetBeans. Για την ανάπτυξή τους χρησιμοποιήθηκε ο Αντικειμενοστραφής Προγραμματισμός. Η εκτέλεση της εφαρμογής απαιτεί την ύπαρξη περιβάλλοντος Java.

Η εφαρμογή θα δίνει τη δυνατότητα στο χρήστη (εκπαιδευτικός κυρίως) να εξάγει

βαθμολογίες με λιγότερη προσπάθεια και σε λιγότερο χρόνο, έχοντας σαν αποτέλεσμα να είναι πιο αποδοτικός και αποτελεσματικός στην εργασία του.



Εικόνα 1: Αυτόματη διόρθωση.

Αντικειμενοστραφής Προγραμματισμός

Ο ***αντικειμενοστραφής προγραμματισμός*** (OOP-Object Oriented Programming), είναι μια προγραμματιστική φιλοσοφία όπως και ο προστακτικός ή ο λογικός προγραμματισμός. Σύμφωνα με τον OPP ένα πρόγραμμα δεν αποτελείται από τα δεδομένα και τον κώδικα που τα επεξεργάζεται αλλά από αντικείμενα (objects) τα οποία εμπεριέχουν τα δεδομένα και τα οποία ανταλλάσουν μεταξύ τους πληροφορίες και μηνύματα προκειμένου να επιτευχθεί ο στόχος του προγράμματος.

Κεντρική ιδέα στον αντικειμενοστραφή προγραμματισμό είναι η ***κλάση*** (class), μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων, είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών, εννοιολογικών αντικειμένων, σε ένα περιβάλλον προγραμματισμού.

Μετά την ευρεία διάδοση του ΑΠ κατά τη δεκαετία του '90, το αντικειμενοστραφές μοντέλο σχεδίασης (με κλάσεις, κληρονομικότητα, αντικείμενα και τυποποιημένες αλληλεπιδράσεις μεταξύ τους) επικράτησε ακόμη και για μοντελοποίηση που δεν περιελάμβανε καν προγραμματισμό (π. χ. σχήματα βάσεων δεδομένων). Έτσι αναπτύχθηκαν διάφορες πρότυπες **γλώσσες μοντελοποίησης λογισμικού** οι οποίες τυποποιούσαν οπτικά σύμβολα και συμπεριφορές με στόχο την αφαιρετική περιγραφή της λειτουργίας και της δομής ενός υπολογιστικού συστήματος. Οι γλώσσες αυτές είχαν εξαρχής έναν εμφανή αντικειμενοστραφή προσανατολισμό. Τελικά οι πιο δημοφιλείς από αυτές ενοποιήθηκαν στο κοινό πρότυπο UML που η πρώτη του έκδοση οριστικοποιήθηκε το 1997.

Η UML πλέον είναι η πρότυπη γλώσσα μοντελοποίησης στη μηχανική λογισμικού. Χρησιμοποιείται για τη γραφική απεικόνιση, προσδιορισμό, κατασκευή και τεκμηρίωση των στοιχείων ενός συστήματος λογισμικού. Μπορεί να χρησιμοποιηθεί σε διάφορες φάσεις ανάπτυξης, από την ανάλυση απαιτήσεων ως τον έλεγχο ενός ολοκληρωμένου συστήματος, και αποτελείται από ένα σύνολο προσυμφωνημένων όρων, συμβόλων και διαγραμμάτων.

OCR

Η **Οπτική Αναγνώριση Χαρακτήρων** (Αγγλ. **O**ptical **C**haracter **R**ecogni tion) ή αλλιώς Αυτόματη Αναγνώριση Χαρακτήρων Κειμένου ονομάζεται η διαδικασία μετατροπής σαρωμένων εικόνων χειρογράφων ή έντυπων κειμένων σε κείμενο αναγνώσιμο από ηλεκτρονικό υπολογιστή. Η Οπτική Αναγνώριση Χαρακτήρων καθιστά εφικτή την εκ νέου επεξεργασία του κειμένου, αποφεύγοντας την δακτυλογράφηση του από την αρχή. Τα συστήματα Οπτικής Αναγνώρισης Χαρακτήρων απαιτούν βαθμονόμηση για να διαβάσουν μια συγκεκριμένη γραμματοσειρά. Οι πρώτες εκδόσεις ήταν προγραμματισμένες με εικόνες για κάθε χαρακτήρα και δούλευαν μια γραμματοσειρά την φορά. Τα ευφυή συστήματα με υψηλό δείκτη αναγνώρισης είναι πλέον κοινά. Μερικά συστήματα είναι ικανά να αναπαράγουν ακόμη και τις πληροφορίες που δεν είναι κείμενο σε ένα έγγραφο, όπως εικόνες, στήλες, γραμμές, γωνίες κτλ.

NetBeans

Το NetBeans είναι ένα ολοκληρωμένο, εξελιγμένο περιβάλλον στο οποίο αναπτύσσεται κυρίως Java γλώσσα, αλλά και άλλες γλώσσες, πιο συγκεκριμένα οι PHP, C/C++ και HTML5. Είναι επίσης μια εφαρμογή πλατφόρμας για Java εφαρμογές και άλλα. Το NetBeans είναι γραμμένο σε Java γλώσσα και μπορεί να τρέξει σε Windows, OS X, Linux, Solaris και σε άλλες πλατφόρμες που υποστηρίζουν JVM. Η πλατφόρμα NetBeans επιτρέπει την ανάπτυξη εφαρμογών από ένα σύνολο στοιχείων λογισμικού που ονομάζονται modules. Εφαρμογές βασισμένες στην πλατφόρμα του NetBeans μπορούν να επεκταθούν από τρίτους προγραμματιστές.

Αυτόματη διόρθωση

Η Αυτόματη Διόρθωση σαν γενικός όρος είναι μια αυτόματη σύγκριση μεταξύ διαφόρων ειδών αρχείων. Επίσης, συγκρίνει το περιεχόμενο των αρχείων, εντοπίζει τα κοινά περιεχόμενά τους και τις διαφορές τους. Το αποτέλεσμα της σύγκρισης, που συχνά αποκαλείται «diff», είναι δυνατό να παρουσιάζεται σε ένα γραφικό περιβάλλον χρήστη ή να χρησιμοποιηθεί ως μέρος διεργασιών σε networks, file systems, ή revision control. Μερικά ευρέως χρησιμοποιούμενα προγράμματα σύγκρισης αρχείων είναι τα diff, cmp, FileMerge, WinMerge, Beyond Compare, και Microsoft File Compare. Πολλά προγράμματα επεξεργασίας και επεξεργαστές κειμένου εκτελούν σύγκριση αρχείων για να τονίσουν τις αλλαγές σε ένα έγγραφο.

Τα περισσότερα εργαλεία σύγκρισης αρχείων βρίσκουν την πιο κοινή υποακολουθία μεταξύ δύο αρχείων. Οποιαδήποτε δεδομένα που δεν είναι στην κοινή υποαλληλουχία, παρουσιάζονται ως «εισαγωγή» ή «διαγραφή». Το 1978, ο Paul Heckel δημοσίευσε έναν αλγόριθμο που προσδιορίζει τα πιο μετακινούμενα τμήματα του κειμένου. Αυτό χρησιμοποιείται στο IBM History Flow tool. Άλλα προγράμματα σύγκρισης αρχείων βρίσκουν block moves. Ορισμένα εξειδικευμένα εργαλεία σύγκρισης αρχείων βρίσκουν τη μεγαλύτερη αυξανόμενη ακολουθία μεταξύ δύο αρχείων. Το rsync πρωτόκολλο χρησιμοποιεί μία κυλιόμενη hash συνάρτηση για να συγκρίνει δύο αρχεία σε δύο μακρινούς υπολογιστές με χαμηλή επιβάρυνση επικοινωνίας. Η σύγκριση αρχείων σε επεξεργαστές κειμένου είναι συνήθως σε επίπεδο λέξης, ενώ η σύγκριση με τα περισσότερα εργαλεία προγραμματισμού σε επίπεδο γραμμής. Η σύγκριση ανά Byte ή ανά χαρακτήρα είναι χρήσιμη σε κάποιες εξειδικευμένες εφαρμογές.

Στην περίπτωση μας, έχουμε δημιουργήσει μία εφαρμογή που μπορεί να συγκρίνει αρχεία κειμένου μεταξύ τους και να υπολογίζει τις διαφορές τους. Η υλοποίησή της στηρίχθηκε σε ένα σωστό αρχείο κειμένου, το οποίο μπορεί να συγκριθεί με ένα ή περισσότερα αρχεία. Στο πρόγραμμά μας θα υπάρχει η δυνατότητα επιλογής θετικής ή/και αρνητικής βαθμολόγησης (εφόσον υπάρχει). Ο χρήστης θα επιλέγει τα αρχεία που επιθυμεί να συγκρίνει και στη συνέχεια η εφαρμογή μας θα υπολογίζει τους βαθμούς των εξεταζόμενων και θα τους αντιστοιχίζει με τα ΑΜ τους. Τέλος, θα παρέχεται η δυνατότητα εκτύπωσης των αποτελεσμάτων ή εξαγωγή αυτών σε ξεχωριστό αρχείο κειμένου.

Αποτελέσματα

Με την ολοκλήρωση της Πτυχιακής Εργασίας, δημιουργήθηκε μία εφαρμογή που απευθύνεται κυρίως σε εκπαιδευτικούς. Το πρόγραμμα είναι ικανό να μειώσει το χρόνο διόρθωσης γραπτών πολλαπλής επιλογής και ταυτόχρονα να αυξήσει το χρόνο των χρηστών. Πρόκειται για μία λειτουργική εφαρμογή γεγονός που φαίνεται από την ευκολία στην εγκατάσταση και τη χρήση. Ο χρήστης για να είναι σε θέση να χρησιμοποιήσει την εφαρμογή απαιτείται μόνο ο Η/Υ του να διαθέτει περιβάλλον Java. Ενώ για να την επεξεργαστεί δεν απαιτείται ιδιαίτερη εκπαίδευση καθώς περιλαμβάνει απλές επιλογές, κατανοητές για όλους.

Σε προσωπικό επίπεδο, η εκπόνηση της συγκεκριμένης Πτυχιακής Εργασίας βελτίωσε σε μεγάλο βαθμό τις γνώσεις μας στον Αντικειμενοστραφή Προγραμματισμό και την ικανότητά μας να επιλύουμε προβλήματα τέτοιου τύπου. Επίσης, αναπτύξαμε χαρακτηριστικά όπως η ευρηματικότητα, η συνεργασία, κ.α.

Συμπεράσματα και Μελλοντικές Επεκτάσεις

Δημιουργήθηκε η εφαρμογή όπως είχε προσχεδιαστεί, με μια μικρή απόκλιση από το αρχικό πλάνο λόγω της μεγάλης πιθανότητας λάθους που παρουσιάζουν τη δεδομένη χρονική περίοδο οι OCR εφαρμογές. Σχεδιάστηκε το πρόγραμμα με γνώμονα το καθηγητή για την εύκολη χρήση του προγράμματος και την ταχύτερη διόρθωση των γραπτών. Έτσι, ο καθηγητής έχει τη δυνατότητα να πραγματοποιήσει τα εξής:

- υπολογισμός των βαθμών των εξεταζόμενων και αντιστοίχηση αυτών με τα ΑΜ τους
- προαιρετική επιλογή θετικής ή/και αρνητικής βαθμολόγησης
- προαιρετική εκτύπωση των αποτελεσμάτων

προαιρετική εξαγωγή των αποτελεσμάτων σε ξεχωριστό αρχείο (π.χ. excel)

Η εφαρμογή «Αυτόματη Διόρθωση» δεν είναι ακόμη έτοιμη για χρήση καθώς στην αγορά δεν υπάρχει OCR κατάλληλη για να ικανοποιήσει τις απαιτήσεις του συστήματος, αν και η Google βρίσκεται στο σωστό δρόμο. Συνεπώς, μελλοντικά θα πρέπει να δημιουργηθεί μία OCR που θα είναι σε θέση να μετατρέψει πλήρως την εικόνα σε text χωρίς να παρουσιάζει κανένα λάθος.

Επίσης, μία μελλοντική επέκταση του προγράμματος θα μπορούσε να είναι η ανάπτυξή του σε ένα λογισμικό κινητής εφαρμογής, όπως είναι το λογισμικό android, όπου θα μπορεί ο φοιτητής μετά το τέλος της εξέτασης, επισυνάπτοντας τη φωτογραφία του γραπτού του σε μία βάση δεδομένων του καθηγητή (online), να λαμβάνει άμεσα τα αποτελέσματα. Στην πράξη, για να πραγματοποιηθεί αυτό το update δε χρειάζεται εκ νέου δημιουργία του προγράμματος, αλλά κάποιες μικρές αλλαγές στον κώδικα ώστε να γίνει συμβατό με κινητή συσκευή.

Βιβλιογραφία

[1] Ιακωβίδης Δ. (χ.χ.). [Online]. Available: www.inf.teilam.gr/OLD/java/Java_Lecture1_OVR.pdf

[2] Βεσκούκης Β. και Κουτουμάνος Α. (2000). Εισαγωγή στη γλώσσα Java. Πανεπιστήμιο Πειραιώς Τμήμα Τεχνολογικής Εκπαίδευσης. [Online]. Available: users.sof/lab.ece.ntua.gr/~bxb/courses/unipi2001_te/00-CourseNotes/031-OO&Java/TE031-2.pdf

[3] Εθνικό Μετσόβειο Πολυτεχνείο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ. (χ.χ.) Εισαγωγή στη γλώσσα προγραμματισμού Java. [Online]. Available: www.ebooks4greeks.gr/down/oads/Pliroforiki/Glosses.program./Java_Downloaded_from_eBooks4Greeks.gr.pdf

[4] Σιακαβέλλα Ε. Η. (2006). Συστήματα Επιχειρησιακής Μοντελοποίησης και Αναπαράστασης Αξιολόγηση και Εφαρμογές. Εθνικό Μετσόβειο Πολυτεχνείο Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων. Αθήνα.

[5] UML notes (n.d.). [Online]. Available: www.icsd.aegean.gr/kotis/softTech06/UMLnotes.pdf

[6] Βιδάκης Ν. (2010). Αντικειμενοστρεφής Προγραμματισμός. Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων. Ηράκλειο, Κρήτης.

[7] Deitel P. and Deteil H. (2010). JAVA Προγραμματισμός. 8η έκδοση. Γκιούρδας Μ.

[8] el.wiki-pedia.org/wiki/Java

[9] en.wikipedia.org/wiki.NetBeans

[10]el.wikipedia.org/wiki/Οπτική_Αναγνώριση_Χαρακτήρων

[11]el.wikipedia.org/wiki/Αντικειμενοστραφής_προγραμματισμός

[12]en.wikipedia.org/wiki/File_comparison