1η εργασία

Μέλη:

Ηρακλής Χρύσανθος 3170183

Γιώργος Λεωνιδόπουλος 3170091

Σταύρος Σοφρώνης 3170150

Format αρχείων εισόδου:

Τα αρχεία τα οποία δέχεται το πρόγραμμά μας είναι ένα αρχείο με τα μαθήματα που διδάσκονται ανά τάξη και ένα αρχείο με τα ονόματα των διαθέσιμων καθηγητών. Συγκεκριμένα, το αρχείο των μαθημάτων (Lessons.txt) περιέχει με την σειρά τον αναγνωριστικό κωδικό του κάθε μαθήματος ο οποίος είναι ένας τριψήφιος αριθμός, έπειτα ακολουθεί το όνομα του μαθήματος με λατινικούς χαρακτήρες, η τάξη στην οποία ανήκει το μάθημα συμβολιζόμενη με “a” για την πρώτη, με “b” για την δευτεύρα και “c” για την τρίτη γυμνασίου αντίστοιχα. Τέλος ακολουθεί ένας αριθμός ο οποίος υποδηλώνει πόσες ώρες πρέπει να διδάσκεται το μάθημα εβδομαδιαία σε κάθε τμήμα της συγκεκριμένης τάξης. Το αρχείο των καθηγητών (Teachers.txt) περιέχει με την σειρά τον αναγνωριστικό κωδικό του κάθε καθηγητή ο οποίος είναι και πάλι ένας τριψήφιος αριθμός, ακολουθεί το ονοματεπώνυμο του κάθε καθηγητή με λατινικούς χαρακτήρες, έπειτα υπάρχουν ένας ή περισσότεροι τριψήφιοι αριθμοί οι οποίοι είναι οι αναγνωριστικοί κωδικοί των μαθημάτων που μπορεί να διδάξει ο καθένας. Τέλος, υπάρχουν 2 ακόμα αριθμοί οι οποίοι έχουν το πρόθεμα «$» και από τους οποίος ο πρώτος αντιστοιχεί στον μέγιστο αριθμό ωρών την ημέρα και ο δεύτερος στον μέγιστο αριθμό ωρών την εβδομάδα που ο κάθε καθηγητής δύναται να διδάξει.

Teacher.java – Lessons.java:

Για την διευκόλυνση μας στην διαχείριση των καθηγητών και των μαθημάτων, δημιουργήσαμε τις κλάσεις Teacher και Lessons αντίστοιχα για να τα αντιμετωπίζουμε ως αντικείμενα. Η κλάση Teacher δέχεται στον κατασκευαστή της ως ορίσματα τον αναγνωριστικό κωδικό του καθηγητή ως string, το όνομα του καθηγητή ως string, έναν πίνακα τύπου string ο οποίος περιέχει τους αναγνωριστικούς κωδικούς των μαθημάτων που μπορεί να διδάξει ο κάθε καθηγητής, τον αριθμό ωρών που μπορεί ο καθηγητής να διδάξει την ημέρα ως int και τέλος τον αριθμό ωρών που μπορεί ο καθηγητής να διδάξει την εβδομάδα και πάλι ως int. Ακόμα, η κλάση πλαισιώνεται από μεθόδους get και set για κάθε πεδίο του κατασκευαστή. Η κλάση Lessons δέχεται στον κατασκευαστή της ως ορίσματα τον αναγνωριστικό κωδικό του μαθήματος ως string, το όνομα του μαθήματος ως string, την τάξη στην οποία πρέπει να διδάσκεται το μάθημα ως string (πχ «a», «b», «c») και τέλος τον

αριθμό ωρών που πρέπει να διδάσκεται ως int. Τέλος πλαισιώνεται και αυτή με τις κατάλληλες μεθόδους get και set.

Cell.java:

Με την χρήση της κλάσης Cell η καθεμία μαθητική ώρα του εβδομαδιαίου προγράμματος του σχολείου αντιμετωπίζεται ως ένα ξεχωριστό αντικείμενο. Ο κατασκευαστής της Cell δέχεται την ημέρα στην οποία αναφερόμαστε ως int, την ώρα ως int, το τμήμα ως int, το μάθημα ως αντικείμενο τύπου Lesson και τον καθηγητή ως αντικείμενο τύπου Teacher. Τέλος η κλάση πλαισιώνεται από μεθόδους get και set για κάθε πεδίο του κατασκευαστή και επίσης υπάρχει και η μέθοδος toStringCell που βοηθάει στην καλύτερη εμφάνιση των δεδομένων.

Lists.java:

Για τον χειρισμό και την αποθήκευση των καθηγητών και των μαθημάτων ως δεδομένα χρησιμοποιούμε arraylists και για την καλύτερη εφαρμογή λειτουργιών πάνω τους δημιουργήσαμε την κλάση Lists μέσα στην οποία ορίζουμε μία λίστα για τους καθηγητές και μία για τα μαθήματα όλων των τάξεων. Ακόμα, υπάρχουν οι μέθοδοι addLesson και addTeacher οι οποίες δέχονται ένα μάθημα και έναν καθηγητή αντίστοιχα και τα προσθέτουν στην αντίστοιχη λίστα. Επίσης έχουμε τις μεθόδους ListLessons και ListTeachers οι οποίες τυπώνουν την λίστα των μαθημάτων και των καθηγητών αντίστοιχα.

Schedule.java:

Η κλάση Schedule είναι υπεύθυνη για την εξαγωγή του αρχείου που περιέχει το τελικό εβδομαδιαίο πρόγραμμα. Συγκεκριμένα, η μέθοδος createFile δέχεται το όνομα του αρχείου ως string και ένα arraylist με αντικείμενα τύπου Cell. Στη συνέχεια ορίζουμε έναν δισδιάστατο πίνακα τύπου Cell 63 γραμμών και 5 στηλών με όνομα schedule. Με ένα εξωτερικό for που διατρέχει την λίστα με τα cell και ένα εσωτερικό που διατρέχει τις ημέρες από Δευτέρα εώς Παρασκευή καθώς και τους κατάλληλους ελέγχους, αποθηκεύουμε το cell στην κατάλληλη θέση στον schedule με βάση το τμήμα της εκάστοτε τάξης συμβολιζόμενο ως int από το 0 ως το 8 (0=α1, 1=α2, ..., 7=γ2, 8=γ3). Με αυτόν τον τρόπο καταφέρνουμε να περάσουμε όλα τα cells σε έναν πίνακα ανά τμήμα (κάθε 7 γραμμές αλλάζουμε τμήμα). Ύστερα με την βοήθεια ενός BufferedWriter, γράφουμε ένα .txt αρχείο όπου πρώτα αναγράφεται το τμήμα και μετά με κατάλληλα for παραθέτουμε για κάθε τμήμα το ημερήσιο πρόγραμμα για όλες τις ημέρες της εβδομάδας(πχ α1 *πρόγραμμα από Δευτέρα ως Παρασκευή*, α2 * πρόγραμμα από Δευτέρα ως Παρασκευή*,κλπ).

StoreFiles.txt:

Η StoreFiles είναι υπεύθυνη για το διάβασμα και την αποθήκευση των .txt αρχείων των μαθημάτων και των καθηγητών. Συγκεκριμένα, μέσω της μεθόδου LoadFile, η οποία δέχεται ως ορίσματα το όνομα του αρχείου ως string και μία λίστα τύπου Lists, με την βοήθεια ενός BufferedReader ξεκινάμε να διαβάζουμε το .txt αρχείο γραμμή-γραμμή. Αρχικά ελέγχουμε το πρώτο ψηφίο του τριψήφιου αριθμού της πρώτης γραμμής και αν αυτό είναι ένα, δύο ή τρία τότε έχουμε αρχείο μαθημάτων και αν είναι τέσσερα τότε έχουμε αρχείο καθηγητών. Αν είμαστε στην πρώτη περίπτωση, διατρέχουμε όλη την γραμμή με ένα while συλλέγοντας ανά χαρακτήρα τα τέσσερα στοιχεία που χρειαζόμαστε για την δημιουργία ενός αντικειμένου τύπου Lesson. Μόλις ολοκληρωθεί η καταγραφή αυτών των στοιχείων τότε δημιουργούμε το αντικείμενο τύπου Lesson και το προσθέτουμε στην λίστα των μαθημάτων μέσω ενός αντικειμένου τύπου Lists. Αν το αρχείο που διαβάζουμε είναι αρχείο καθηγητών τότε και πάλι ακολουθεί μία παρόμοια διαδικασία για την καταγραφή των στοιχείων που χρειαζόμαστε για να δημιουργήσουμε ένα αντικείμενο τύπου Teacher. Η μόνη διαφορετική επεξεργασία είναι ο εντοπισμός των θέσεων των κωδικών με το πρόθεμα «$», η αποθήκευση αυτών με την βοήθεια της μεθόδου split και η μετατροπή αυτών από string σε int για λόγους συμβατότητας με την κλάση μας. Έπειτα από αυτή την επεξεργασία δημιουργείται ένα αντικείμενο τύπου Teacher και προσθέτεται στην λίστα των καθηγητών μέσω ενός αντικειμένου τύπου Lists.

State.java:

Η κλάση State είναι υπεύθυνη για την αρχικοποίηση ενός εβδομαδιαίου προγράμματος το οποίο δημιουργείται εν μέρει από εμάς και εν μέρει τυχαία. Για τον λόγο αυτό πρέπει να ακολουθεί ορισμένους περιορισμούς. Πιο αναλυτικά, στην State ορίζουμε δύο είδη κατασκευαστών, εκ των οποίων ο πρώτος δέχεται ως όρισμα ένα αντικείμενο τύπου Lists αποκλειστικά για να έχει πρόσβαση στην λίστα των μαθημάτων και των καθηγητών που βρίσκονται εκεί. Με τους κατάλληλους ελέγχους για την τάξη και μέσω μιας μεθόδου που θα αναλυθεί παρακάτω (InititializeSchedule) δημιουργούμε ένα τυχαίο εβδομαδιαίο πρόγραμμα πέντε ημερών και εφτά ωρών. Το δεύτερο είδος κατασκευαστή είναι ένας copy constructor ο οποίος δέχεται ως ορίσματα μία λίστα τύπου Cell, δηλαδή μια λίστα με ήδη έτοιμα κελιά εβδομαδιαίου προγράμματος των οποίων τα στοιχεία παίρνει και με αυτά δημιουργεί μία λίστα αντίγραφο. Στη συνέχεια, με την μέθοδο InititializeSchedule δημιουργούμε την **αρχική μας κατάσταση** δηλαδή ένα εβδομαδιαίο πρόγραμμα. Η μέθοδος αυτή δέχεται ως ορίσματα την τάξη ως string, ένα προσωρινό αντικείμενο τύπου Cell, έναν πίνακα με τις συνολικές εβδομαδιαίες ώρες κάθε τμήματος και ένα αντικείμενο τύπου Lists. Στη συνέχεια βρίσκουμε τυχαία ένα μάθημα και ελέγχουμε με την βοήθεια ενός παράλληλου πίνακα αν ανήκει στην τάξη και αν έχει συμπληρώσει τις εβδομαδιαίες ώρες που πρέπει να διδαχθεί. Τέλος ψάχνουμε έναν τυχαίο καθηγητή μέχρι να βρούμε κάποιον ο οποίος μπορεί να πάρει το μάθημα που βρήκαμε από πάνω. Έπειτα ακολουθεί η μέθοδος calculateScore η οποία παίρνει ως όρισμα ένα αντικείμενο τύπου Lists. Η συγκεκριμένη μέθοδος ελέγχουμε αν παραβιάζεται κάποιος από τους παρακάτω περιορισμούς:

- Ο κάθε καθηγητής να μην έχει δύο συνεχόμενες ώρες μαθήματος.
- Ο κάθε καθηγητής να μην κάνει παραπάνω ώρες την ημέρα από όσες μπορεί.
- Ο κάθε καθηγητής να μην κάνει παραπάνω ώρες την εβδομάδα από όσες μπορεί.
- Να μην μπαίνουν σε μία μέρα όλες οι εβδομαδιαίες του μαθήματος.
- Να κάνει **ένας** καθηγητής το κάθε μάθημα στο εκάστοτε τμήμα.
- Να μην κάνει καθηγητής σε μία ώρα μάθημα σε παραπάνω από ένα τμήμα.

Επίσης από τη συνολική δομή του προγράμματος επιτυγχάνονται οι εξής περιορισμοί:

- Να μην υπάρχουν κενά στο πρόγραμμα κανενός τμήματος αφού όλες οι μέρες εχουν εφτά ώρες μαθήματος.
- Ο ημερήσιος αριθμός ωρών διδασκαλίας κάθε τμήματος είναι ομοιόμορφος για όλες τις ημέρες.
- Ο αριθμός ωρών διδασκαλίας ανά εβδομάδα είναι ομοιόμορφος για όλους τους καθηγητές.

Ο τρόπος που υπολογίζουμε το σκορ είναι αντίθετος, δηλαδή κάθε φορά που παραβιάζεται κάποιος από τους παραπάνω περιορισμούς το σκορ αυξάνεται. Έτσι αυτό που προσπαθούμε να επιτύχουμε είναι ένα πρόγραμμα με το ελάχιστο δυνατό σκορ. Στη συνέχεια υπάρχει η μέθοδος SwapHours η οποία παίρνει ως ορίσματα δύο ώρες ως int, δύο μέρες ως int, ένα τμήμα ως int και ένα αντικείμενο τύπου Lists και αλλάζει τις ώρες και τις μέρες ενός μαθήματος για ένα συγκεκριμένο τμήμα. Τέλος, υπάρχει η μέθοδος ChangeTeacher η οποία για ένα τμήμα και για ένα μάθημα, βρίσκει ικανό αντικαταστάτη κάποιου καθηγητή και θέτει αυτόν ώστε να διδάξει το μάθημα.

HillClimb.java:

Για την επίλυση της εργασίας επιλέξαμε ως αλγόριθμο τεχνητής νοημοσύνης τον Hill Climbing με πρώτη επιλογή. Αυτός υλοποιείται στην κλάση HillClimb. Αρχικά υλοποιούμε μία μέθοδο HillClimbAlgorithm η οποία δέχεται ως ορίσματα τα βήματα (ως int) που θα ψάξει ο αλγόριθμος και ένα αντικείμενο τύπου Lists. Αυτό που κάνει είναι να φτιάχνει αρχικά μια τυχαία αρχική κατάσταση. Με μία πιθανότητα 50% αν θα αλλάζουμε ώρες ή καθηγητή και ελέγχουμε αν η καινούργια κατάσταση έχει χαμηλότερο σκορ από την παλιά κατάσταση. Αν αυτό ισχύει τότε η νέα κατάσταση θα γίνει ο πατέρας και θα συνεχιστεί η ίδια διαδικασία για άλλες τόσες φορές όσες τα βήματα της εισόδου.

MainApp.java:

Στην MainApp γίνονται οι κλήσεις κάποιον μεθόδων κλάσεων. Αρχικά περνάμε τα αρχεία στο πρόγραμμά μας και καλούμε την μέθοδο HillClimbAlgorithm και δημιουργούμε το τελικό μας αρχείο με το εβδομαδιαίο πρόγραμμα με την μέθοδο createFile της κλάσης Schedule και τέλος τυπώνουμε τον χρόνο εκτέλεσης του προγράμματός μας.

Στην παρακάτω εικόνα έχουμε ένα παράδειγμα εκτέλεσης του προγράμματος με το τελικό σκορ να εμφανίζεται δίπλα στο τρέχων σκορ και ο αριθμός είναι ο χρόνος εκτέλεσης σε nanosecond.

```
new score: 36
new score: 35
new score: 34
new score: 33
new score: 32
new score: 31
new score: 30
new score: 29
new score: 28
new score: 26
new score: 25
new score: 24
new score: 23
new score: 22
new score: 21
new score: 20
new score: 16
new score: 14
new score: 13
new score: 12
new score: 11
new score: 10
new score: 8
new score: 7
new score: 6
new score: 2
Trexon score: 2
7448247814
```