Αναφορά Τελικής Εργασίας στο μάθημα Τεχνολογία Πολυμέσων

Ανδρεάδης Δημήτρης 03119403

Η ανάπτυξη του project Minesweeper έγινε με τη χρήση του IntelliJ IDE.

Περιγραφή Λογικής Παιχνιδιού και Γραφικής Διεπαφής Χρήστη:

Ο πηγαίος κώδικας περιέχει τα παρακάτω αρχεία/κλάσεις:

- **Minesweeper:** Περιέχει την main class η οποία διευκρινίζεται και στο pom.xml αρχείο
- **MainHandler:** Αποτελεί τον controller του fxml αρχείου main. fxml, διαχειρίζεται δηλαδή το παράθυρο της κύριας οθόνης. Η κύρια οθόνη περιέχει το menu Application (Create/Load/Start/Exit) και το menu Details (Rounds/Solution), τα πεδία Total Mines,Flags left, Time left και το pane στο οποίο ανοίγει το grid του παιχνιδιού (grid από panes με γκρι χρώμα) όταν ξεκινήσει ένα νέο παιχνίδι με ήδη φορτωμένο σενάριο όταν επιλέξουμε Start. Η επιλογή των menu items Create,Load, Rounds,Solution ανοίγει νέα παράθυρα πάνω στο κυρίως παράθυρο τα οποία ελέγχουν οι συναρτήσεις controllers που περιγράφονται στη συνέχεια και ορίζονται στον MainHandler.
  Η λειτουργία αντίστροφου μετρητή (Time left) υλοποιείται με την χρήση του package Animation όπου με τη χρήση ενός timeline ανανεώνεται η ένδειξη του πεδίου του χρόνου της εφαρμογής και μειώνεται κατά 1 με την παρέλευση 1 sec.
  Με την χρήση της onSquareClick ο χρήστης μπορεί είτε να επιλέξει ένα κουτί που θα αποκαλυφθεί (αριστερό κλικ) είτε να μαρκάρει ένα κουτί ως πιθανή θέση νάρκης (δεξί κλικ). Κάθε φορά ελέγχεται αν η θέση περιέχει νάρκη και αποκαλύπτονται τα γειτονικά κουτιά σύμφωνα με τη λογική του παιχνιδιού Minesweeper.
  Το αποτέλεσμα της επιλογής του χρηστή που οδηγεί στην φανέρωση νέων κουτιών ανανεώνεται στο grid με χρήση της refreshBoardTiles η οποία χρωματίζει με άσπρο τα κουτιά που φανερώνονται.
  Ο παίκτης μπορεί είτε να επιλέξει νάρκη και να χάσει είτε να προλάβει να φανερώσει όλα τα tiles είτε να εκπνεύσει ο χρόνος. Το αποτέλεσμα κάθε αγώνα αποθηκεύει η updateRounds στο αρχείο rounds.txt το οποίο χρησιμοποιεί ο controller του RoundHandler προκειμένου να εμφανίσει τα αποτελέσματα των τελευταίων 5 αγώνων.

- **CreateHandler:** Αποτελεί τον controller του fxml αρχείου create.fxml, διαχειρίζεται δηλαδή το παράθυρο δημιουργίας νέου σεναρίου. Συγκεκριμένα διαβάζει τις τιμές τόσο των φορμών TextFields όσο και του Toggle group των radio buttons 1 και 2 (μπορεί να επιλεγεί μόνο ένα από τα 2). Στη συνέχεια ελέγχει αν οι τιμές είναι έγκυρες για την δημιουργία σεναρίου με την validValueScenario. Στα TextFields φαίνονται ποιες είναι οι έγκυρες τιμές κάθε φορά προκειμένου να διευκολύνεται ο

χρήστης να τις εισάγει ορθά. Το σενάριο αποθηκεύεται στον default φάκελο medialab.

- **LoadHandler:** Αποτελεί τον controller του fxml αρχείου load.fxml, διαχειρίζεται δηλαδή το παράθυρο δημιουργίας νέου σεναρίου. Συγκεκριμένα, μέσω της findScenario αναζητά κάποιο αρχείο σεναρίου με το δοθέν όνομα. Εφόσον βρεθεί, ελέγχεται όπως και με τον CreateHandler εάν οι τιμές που περιέχει είναι έγκυρες τιμές σεναρίου με την validValueScenario. Εφόσον φορτωθεί επιτυχώς, κλείνει το παράθυρο και ενημερώνεται το πεδίο του menu με το όνομα του αρχείου που περιέχει το σενάριο. Έτσι, ο χρήστης ξέρει πως μπορεί να εκκινήσει το παιχνίδι εφόσον έχει φορτωθεί το αρχείο με τις παραμέτρους του παιχνιδιού.
- **InvalidDescriptionException:** Κατά την φόρτωση ενός αρχείου σεναρίου σε περίπτωση που έχουμε NoSuchElementException (σενάριο με λιγότερες από 4 παραμέτρους) τότε την κάνουμε catch και κάνουμε throw μια InvalidDescriptionException. Η τελευταία εκτυπώνει ένα μήνυμα λάθους.
- **InvalidValueException:** Κατά το διάβασμα των παραμέτρων ενός σεναρίου είτε από create είτε από load καλείται η validValueScenario. Αυτή σε περίπτωση που εντοπίσει invalid παραμέτρους κάνει throw InvalidValueException, η οποία εξαίρεση εκτυπώνει πάλι ένα μήνυμα λάθους.
- **Roundhandler:** Αποτελεί τον controller του fxml αρχείου rounds.fxml, διαβάζει και φορτώνει δηλαδή μέσω της initialize το αρχείο rounds.txt που περιέχει τα τελευταία 5 αποτελέσματα.
- **Game:** Περιέχει το πεδίο mineBoard το οποίο περιέχει τις θέσεις των ναρκών καθώς και το revealedBoard που περιέχει τις θέσεις των κουτιών που έχουν αποκαλυφθεί στο χρήστη. Ο κατασκευαστής Game κατασκευάζει με βάση τις παραμέτρους του σεναρίου ένα instance Game. Επιλέγει τυχαία κάθε φορά τις θέσεις των ναρκών και τις αποθηκεύει σε ένα πίνακα. Επιλέγεται επίσης τυχαία η θέση της υπερνάρκης εάν χρειάζεται. Η checkClickedBox ελέγχει αν το επιλεγμένο box με την onSquareClick περιέχει νάρκη. Εάν περιέχει ενημερώνει το status σε loss. Εάν δεν περιέχει και εάν αποκαλυφθούν όλα τα τετράγωνα ενημερώνει το status σε win. Η αναδρομική αποκάλυψη γειτονικών άδειων boxes γίνεται με την revealBox. Μια θέση σημαδεύεται ως πιθανή θέση νάρκης με την mark (που αντιστοιχεί σε δεξί κλικ) ενώ στην περίπτωση ύπαρξης υπερνάρκης εάν στις πρώτες 4 προσπάθειες σημαδέψουμε την υπερνάρκη γίνεται αποκάλυψη στον παίκτη όλων των κουτιών στην ίδια σειρά και στήλη με την υπερνάρκη μέσω της superReveal.

  Η endGameReveal φανερώνει στο board όλες τις θέσεις με νάρκες τις οποίες σημαδεύει με M. Καλείται όταν φανερώνεται η λύση στον παίκτη μέσω της endgame.

  Στο revealedBoard συμβολίζουμε με '\u0000' τα κουτιά που δεν έχουν επιλεγεί ούτε για αποκάλυψη ούτε ως πιθανές θέσεις νάρκες. Οι πιθανές θέσεις συμβολίζονται με ‘F’, τα κουτιά που δεν έχουν γύρω νάρκες με ‘E’ και τέλος οι νάρκες μετά την αποκάλυψη τους (είτε μετά από ήττα του παίκτη είτε λόγω υπερνάρκης) με ‘M’.

Λοιπές απαιτήσεις:

Οι μέθοδοι και τα πεδία όλων των handlers/controllers είναι private σύμφωνα με τις αρχές OOP. Τα πεδία των controllers που χρειάζονται σε άλλες κλάσεις/controllers δημιουργούν

την ανάγκη δημιουργίας public μεθόδων getters σε αυτούς. Η κλάση Game επιλέγεται ως αυτή που θα πλαισιωθεί με σχόλια για τις public μεθόδους της σύμφωνα με το Javadoc documentation αφού περιέχει public μεθόδους που χρησιμοποιούν οι controllers.