

ΑΝΑΦΟΡΑ ΕΞΑΜΗΝΙΑΙΑΣ ΕΡΓΑΣΙΑΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ «ΤΕΧΝΟΛΟΓΙΑ ΠΟΛΥΜΕΣΩΝ» ΤΟΥ 7^{ΟΥ} ΕΞΑΜΗΝΟΥ

Δημήτριος Βασιλείου, 03119830, ΣΗΜΜΥ

Στο παρόν έγγραφο περιγράφεται με συντομία ο σχεδιασμός της υλοποίησης της εφαρμογής, και λοιπά σχόλια για την εφαρμογή.

Σχεδιασμός Υλοποίησης

- Η κλάση `GameConfiguration` αναλαμβάνει να θέσει τις προδιαγραφές του παιχνιδιού μέσω του αρχείου που φορτώνεται από το μενού. Συγκεκριμένα καθορίζει το ύψος (`height`) και το πλάτος (`width`) του `board`, τον αριθμό των ναρκών (`minesNumber`), το διαθέσιμο χρόνο σε `seconds` (`time`), το αν υπάρχει ή όχι υπερ-νάρκη (`superMine`) και τη δυσκολία (`difficulty`) μέσω των αντίστοιχων **στατικών** μεθόδων.
- Η `public static` μέθοδος `gameRules()` της κλάσης δέχεται ως όρισμα το αρχικό `stage` της εναρκτήριας κλάσης (`HelloApplication`) και δημιουργεί ένα άδειο αρχικά πλαίσιο μαζί με τη μπάρα του μενού, μέσω ενός `BorderPane`. Οι λειτουργίες που επιτελούνται με το πάτημα σε κάθε `menu item` υλοποιούνται στην τρέχουσα μέθοδο, δηλαδή η δημιουργία του αρχείου περιγραφής του παιχνιδιού, το φόρτωμα του επιθυμητού αρχείου περιγραφής, η έναρξη του παιχνιδιού, καθώς και η αποκάλυψη των θέσεων όπου βρίσκονται οι νάρκες. Για την εκτέλεση κάποια λειτουργίας με το πάτημα σε κάποιο `menu item`, χρησιμοποιήθηκαν οι συναρτήσεις `setOnAction()` και `setOnMouseClicked()`.
- Σημειώνεται πως όταν πατηθεί το κουμπί `Start`, η μέθοδος `gameRules()` φτιάχνει ένα νέο αντικείμενο της κλάσης `Game` το οποίο δέχεται ως παραμέτρους μέσω κατάλληλων `getters` στον κατασκευαστή του την δυσκολία, τον αριθμό των ναρκών, το διαθέσιμο χρόνο την ύπαρξη ή όχι υπερ-νάρκης, το ύψος και το πλάτος, όπως αυτά έχουν καθοριστεί από την κλάση `GameConfiguration`. Επίσης, δέχεται το `stage` της αρχικής κλάσης, καθώς και το αρχικό `BorderPane` που έχει σχηματιστεί.
- Για το νέο παιχνίδι που δημιουργήθηκε, η `gameRules()` καλεί στη συνέχεια τη μέθοδο `initBoard()`, η οποία δημιουργεί ένα `GridPane` το οποίο θα αποτελέσει το «πλέγμα» με τα τετράγωνα. Η κλάση `Game` περιλαμβάνει ένα `public` πεδίο `squares` που περιέχει όλα τα τετράγωνα. Μέσω ενός `for-loop`, το `GridPane` και το πεδίο `squares` γεμίζουν με νέα αντικείμενα της κλάσης `Square`. Κάθε αντικείμενο της κλάσης `Square`, περιέχει ως πεδίο έναν πίνακα από `Square` που περιέχει όλα τα γειτονικά τετράγωνα. Ο πίνακας αυτός γεμίζει στην μέθοδο `initBoard()` μέσω `for-loops`. Η ίδια μέθοδος έπειτα, δημιουργεί ένα πλαίσιο κάτω από το μενού, με τον αριθμό των ναρκών, τον αριθμό των μαρκαρισμένων τετραγώνων και το χρονόμετρο.
- Η κλάση `Game` περιλαμβάνει ακόμα μία `public` μέθοδο `win()`, η οποία καλείται όταν ο παίκτης νικάει, και απεικονίζει στην οθόνη ένα `popup window` που αναγράφει ότι ο παίκτης νίκησε. Η μέθοδος `lose()`, καλείται αντίστοιχα όταν ο παίκτης χάνει και εμφανίζει κατάλληλο μήνυμα στην οθόνη μέσω `popup window`. Σημειώνεται πως η κλήση των στατικών μεθόδων `win()` και `lose()` πραγματοποιείται μέσα στην κλάση `Square`, καθώς αυτή αναλαμβάνει την αποκάλυψη των τετραγώνων, όπως θα αναφερθεί στη συνέχεια. Τέλος, η μέθοδος `initMines()` της κλάσης `Game`, τοποθετεί τυχαία σε αντικείμενα `Square` που δημιουργήθηκαν προηγουμένως, νάρκη μέσω του πίνακα `squares`, και γεμίζει το αρχείο `"mines.txt"` με τις συντεταγμένες των ναρκών. Η μέθοδος αυτή, καλείται στην μέθοδο `gameRules()` της `GameConfiguration`, μετά την `initBoard()`.
- Η κλάση `Square` που αναφέρθηκε προηγουμένως, κληρονομεί από την κλάση `StackPane` και περιλαμβάνει τις `x` και `y` συντεταγμένες (`xCord`, `yCord`) ένα `Vector` από `Square` με τα γειτονικά `Squares` (`neighbors`), το αν είναι αποκαλυμμένο ή όχι (`isRevealed`), το αν περιέχει νάρκη (`hasBomb`), το αν περιέχει υπερ-νάρκη (`hasSuperMine`) και ένα `Text` που θα γράφει είτε τους γείτονες, είτε `"X"` για τη

νάρκη, είτε “F” για τη σημαία. Επίσης, περιέχει ένα στατικό πεδίο `flagCounter` που μετράει το πλήθος των σημαιών που έχουν χρησιμοποιηθεί και ένα στατικό πεδίο `movesCounter` για τον αριθμό των προσπαθειών (όπως αναφέρεται στην εκφώνηση για την υπερ-νάρκη). Η κλάση `Square`, αναλαμβάνει την αποκάλυψη των τετραγώνων, το μαρκάρισμα με σημαία, και την αποκάλυψη της νάρκης, αν ο παίκτης πατήσει πάνω σε νάρκη. Για την αποκάλυψη των τετραγώνων υπάρχει η `public` μέθοδος `reveal()`, και συγκεκριμένα μέσα σε αυτή, γίνονται τα εξής:

- ➔ Αν το τετράγωνο που πατιέται δεν έχει νάρκη, καλείται η μέθοδος `squareReveal()`, αλλιώς ο παίκτης χάνει και καλείται η μέθοδος `lose()` της κλάσης `Game`.
- ➔ Αν πατιέται το τετράγωνο με δεξί κλικ αντί για αριστερό, καλείται η μέθοδος `squareRevealWithFlag()`. Αν ωστόσο η το τετράγωνο που πατιέται έχει υπερ-νάρκη, και δεν έχουν ξεπεραστεί οι 4 κινήσεις, καλείται η μέθοδος `flagSuperMine()` για κάθε τετράγωνο, και αποκαλύπτονται όσα τετράγωνα βρίσκονται στην ίδια γραμμή και στήλη με αυτό που περιέχει την υπερ-νάρκη.

Τέλος, οι κλάσεις `InvalidDescriptionException` και `InvalidValueException` κληρονομούν από την κλάση `Exception` και αναλαμβάνουν την δημιουργία `popup windows` με κατάλληλα μηνύματα όταν προκύπτουν τα αντίστοιχα σφάλματα. Η κλάση `GameConfiguration`, στην μέθοδο `gameRules`, κατά το πάτημα του κουμπιού `Load`, μπορεί να «πετάξει» αυτές τις εξαιρέσεις.

Λοιπά Σχόλια

- Για το `project`, χρησιμοποιήθηκε ο IDE `IntelliJ`.
- Το γραφικό περιβάλλον υλοποιήθηκε εξ ολοκλήρου στις κλάσεις και δεν χρησιμοποιήθηκε καθόλου το πρόγραμμα `SceneBuilder`.
- Η κλάση `Mine` δεν χρειάστηκε.
- Η τεκμηρίωση με το εργαλείο `Javadoc`, πραγματοποιήθηκε για όλες τις `public` μεθόδους της κλάσης `Square`.
- Το αρχείο “`rounds.txt`” στο φάκελο `medialab` αποθηκεύει τα δεδομένα για όλα τα παιχνίδια και χρησιμοποιείται για την λειτουργικότητα του `button` “`Rounds`” στο μενού “`Details`”.