

**ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ**

**ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ**

**ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΓΕΝΝΗΤΡΙΑ ΕΠΙΤΡΑΠΕΖΙΩΝ ΠΑΙΧΝΙΔΙΩΝ**

**Δεύτερη Εβδομάδα**

**Ομαδική Εργασία:**

Αναστασιάδης Αλκίνοος (20003)

Ζήνα Ελένη (20046)

Λαγιόκαπας Δημήτριος (20079)

Μακρή Στυλιανή (20060)

Επιβλέποντες:

Κουρέας Αργύριος

Λάντζος Θεόδωρος

**ΣΕΡΡΕΣ****, 3 έως 7ΑΠΡΙΛΙΟΥ 2023**

Περιεχόμενα

[Περίληψη 3](#_Toc131700936)

[Εισαγωγή 4](#_Toc131700937)

[1. Μεθοδολογία 5](#_Toc131700938)

[2. Υλοποίηση 6](#_Toc131700939)

[3. Χρονοδιάγραμμα 14](#_Toc131700940)

[4.Αποτελέσματα-Επίλογος 16](#_Toc131700941)

[5.Βιβλιογραφία 16](#_Toc131700942)

# Περίληψη

Σκοπός της εργασίας ήταν η ομαδική υλοποίηση μιας γεννήτριας «επιτραπέζιων» παιχνιδιών με χρήση της γλώσσας προγραμματισμού Java. Η εφαρμογή θα πρέπει να δίνει τη δυνατότητα να διαβάζει μια περιγραφή του παιχνιδιού σε μορφή JSON και στη συνέχεια να δημιουργεί το παιχνίδι. Για αρχή δημιουργήσαμε ένα πολύ απλό

παιχνίδι (τύπου φιδάκι) με ταμπλό 40 τετραγώνων (tiles) και 4 παίκτες όπου ένας από τους οποίους είναι νικητής του παιχνιδιού, όταν φτάσει πρώτος στο τερματισμό με βάση κάποιους κανόνες. Έπειτα, εμπλουτίσαμε τον κώδικα, και επεκτείναμε την αρχιτεκτονική του προγράμματος έτσι ώστε τα στοιχεία του παιχνιδιού να προέρχονται από αρχεία json.

# Εισαγωγή

Με την αύξηση της δημοτικότητας των βιντεοπαιχνιδιών την τελευταία περίοδο, το game development είναι μια περιζήτητη δεξιότητα. Παρ’ όλ’ αυτά, η ανάπτυξη βιντεοπαιχνιδιών είναι μια περίπλοκη και χρονοβόρα διαδικασία που απαιτεί γνώσεις προγραμματισμού, graphic design κλπ. Αυτό συχνά το κάνει δύσκολο για τους μη-έμπειρους ή τους πρωτάρηδες να φτιάξουν τα δικά τους παιχνίδια.

Για να λυθεί αυτό το πρόβλημα, αναπτύχθηκαν οι γεννήτριες παιχνιδιών έτσι ώστε να αυτοματοποιηθεί κάποιο κομμάτι της ανάπτυξης παιχνιδιών. Η γεννήτρια που θα αναπτύξουμε στη διάρκεια αυτού του εξαμήνου είναι ένα τέτοιο εργαλείο. Στόχος του προγράμματος μας είναι η γεννήτρια να διαβάζει την περιγραφή του παιχνιδιού μέσα από ένα json αρχείο και να δημιουργεί ένα επιτραπέζιο παιχνίδι.

Για να πετύχουμε τους στόχους αυτής της εργασίας, πρέπει να απαντήσουμε στα παρακάτω ερωτήματα:

* Ποια είναι τα κύρια χαρακτηριστικά ενός επιτραπέζιου παιχνιδιού;
* Πόσο αποτελεσματική είναι η γεννήτρια στο να δημιουργεί playable παιχνίδια από τις περιγραφές των json αρχείων;
* Πως θα επαναχρησιμοποιήσουμε τον κώδικα για άλλα projects;
* Πώς συγκρίνεται η γεννήτρια που φτιάξαμε με την παραδοσιακή μέθοδο ανάπτυξης παιχνιδιών με βάση την ευκολία χρήσης, την ποιότητα των παιχνιδιών που δημιουργούνται, και την ικανοποίηση του χρήστη;

# 1. Μεθοδολογία

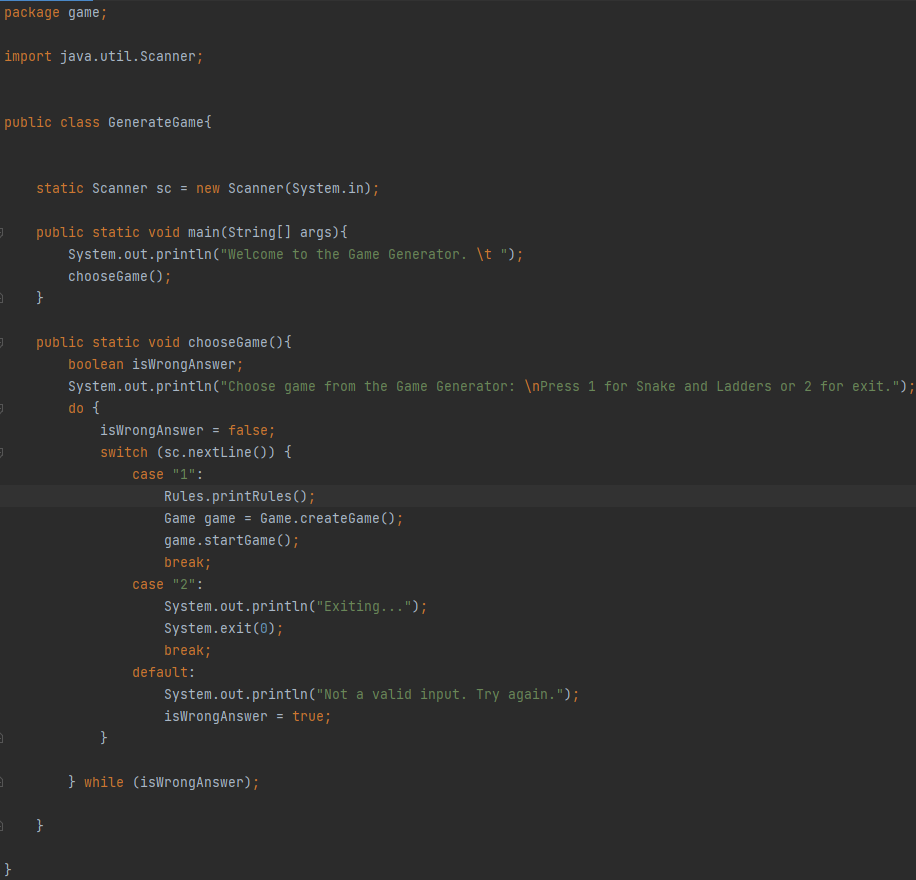
Για να δημιουργήσουμε την αρχική μορφή της γεννήτριας, χρησιμοποιήσαμε ένα συνδυασμό εργαλείων και λογισμικού, συμπεριλαμβανομένου του IntelliJ IDEA ως το περιβάλλον ανάπτυξης του προγράμματος και της βιβλιοθήκης JSON.

Αρχικά, χρησιμοποιήσαμε το IntelliJ IDEA ως το ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment – IDE) για να γράψουμε τον Java κώδικα που σχηματίζει τον κορμό του προγράμματος. Επιλέξαμε το IntelliJ IDEA λόγω των εύκολων στη χρήση χαρακτηριστικών του που μας επέτρεψε να γράψουμε έναν αποδοτικό καλογραμμένο κώδικα. Το JSON είναι ένας τρόπος αναπαράστασης δεδομένων δομημένος σε ένα αρχείο, έτσι ώστε με κατάλληλη μέθοδο να παίρνουμε τα δεδομένα που εμπεριέχονται και να τα κάνουμε επαναχρησιμοποίηση σε πολλά προγράμματα. Γι’ αυτό το λόγο, τα αποθηκεύσαμε όλα τα χαρακτηριστικά σε ένα αρχείο για να έχουμε πρόσβαση στην περιγραφή των παιχνιδιών με ευκολία.

Φέραμε το πρόγραμμα σε επεκτάσιμη μορφή έτσι ώστε να είναι έτοιμο για τις επόμενες υλοποιήσεις. Δημιουργήθηκε το πρότυπο για διαχείριση διαφορετικών ταμπλό, ανάγνωση όλων των χαρακτηριστικών (όπως παίκτες, μέγεθος ταμπλό, κανόνες κλπ) μέσα από το json αρχείο. Προσπαθήσαμε να είναι ξεκάθαρο το πως υλοποιείται και το πως συνδέεται με το υπόλοιπο πρόγραμμα το κάθε κομμάτι και ο κάθε παράγοντας που συντελούν το επιτραπέζιο παιχνίδι με τη χρήση διαφορετικών κλάσεων, μεταβλητών κλπ.

# 2. Υλοποίηση

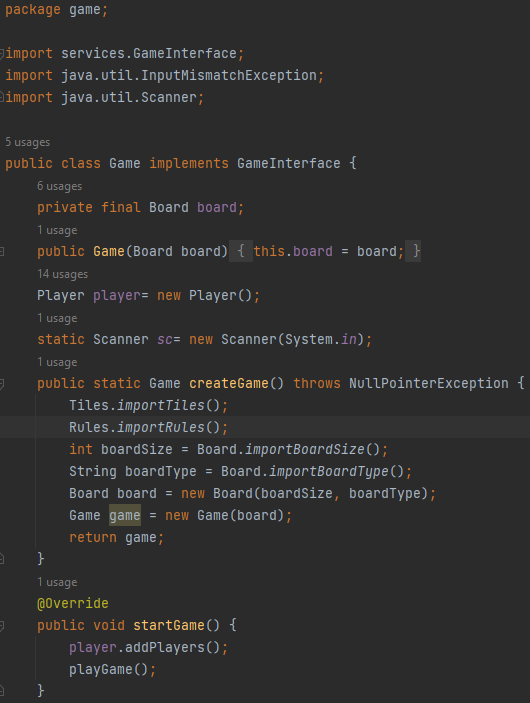
Το πρόγραμμα περιέχει πλέον, πέρα από την κύρια κλάση GenerateGame που εκτελεί το πρόγραμμα, τις κλάσεις: Game, Board, Player, Specifications, Rules και Cards (που δεν χρησιμοποιείται ακόμα σε αυτό το στάδιο).

Συνοπτικά, η GenerateGame είναι η κύρια κλάση του προγράμματος, υπεύθυνη για την επιλογή του παιχνιδιού από τον χρήστη και την δημιουργία του παιχνιδιού χρησιμοποιώντας την κλάση Game με βάση την επιλογή που έκανε.

Παρατηρείται στην αρχή του κώδικα ένα «package game». Έχουμε δημιουργήσει ένα custom πακέτο game για το πρόγραμμα μας, το οποίο μας δίνει την δυνατότητα να κάνουμε «import» διάφορες οντότητες από όλες τις κλάσεις, με σκοπό να τις χρησιμοποιούμε όσες φορές θέλουμε χωρίς να δηλώνονται δεύτερη φορά σε κάποια άλλη κλάση.

Η κλάση Game αντιπροσωπεύει την δημιουργία, την έναρξη, την λειτουργία και τον τερματισμό του παιχνιδιού. Αυτή δηλαδή επικοινωνεί με όλες τις υπόλοιπες κλάσεις και αποκτά πρόσβαση σε όλες τις οντότητες του προγράμματος.

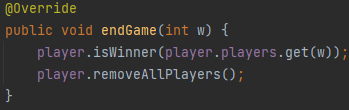
Η συνάρτηση createGame() δημιουργεί το παιχνίδι εισάγοντας τα δεδομένα που έχουν διαβαστεί από το αρχείο json. Χρησιμοποιεί constructors για να δημιουργήσει αντικείμενο παιχνιδιού με βάση αυτά τα δεδομένα που δέχεται. Η συνάρτηση startGame() καλεί την προσθήκη των παικτών και ξεκινάει το παιχνίδι.



Η συνάρτηση playGame() μαζί με την movePosition() της Board, που πλέον δέχεται και τον τύπο του ταμπλό σαν όρισμα, υλοποιούν το gameplay και τις μετακινήσεις των παικτών παράλληλα.

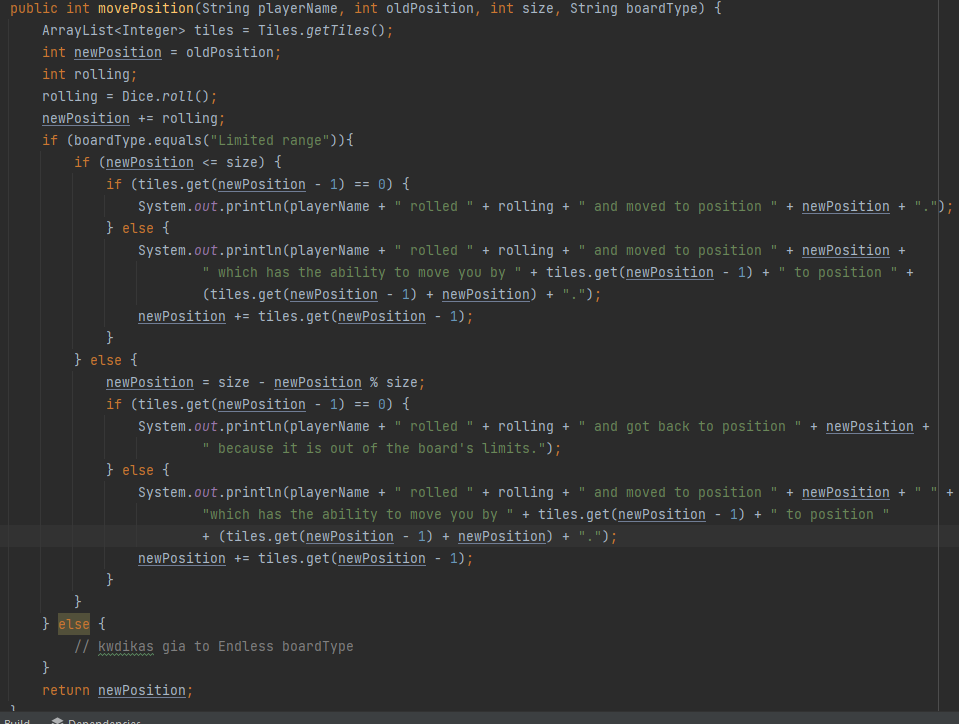


Η συνάρτηση endGame() καλείται μέσα στην playGame(), για να πάρει τον νικητή (την θέση του στην λίστα των παικτών) και τερματίζει το παιχνίδι, διαγράφοντας όλους τους παίκτες για απελευθέρωση πόρων.



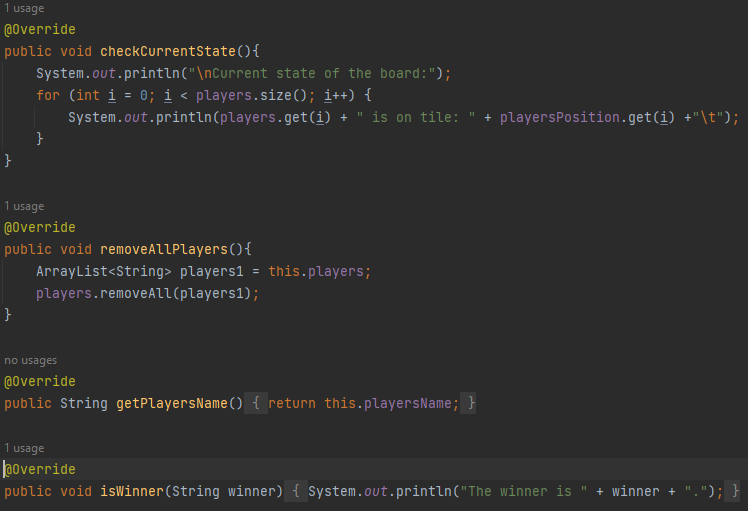
Η κλάση Rules έχει τον ρόλο της εμφάνισης των κανόνων του παιχνιδιού που επέλεξε ο παίκτης και τους εμφανίζει με την συνάρτηση printRules(), η οποία δέχεται αυτούς τους κανόνες από το αρχείο json.

Η κλάση Board αντιπροσωπεύει το ταμπλό του παιχνιδιού και υλοποιεί τη λογική της κίνησης των παικτών στο ταμπλό με την συνάρτηση movePosition(), η οποία δέχεται το όνομα του παίκτη, τη θέση του στο ταμπλό, το μέγεθος και τον τύπο του ταμπλό και υπολογίζει τη καινούρια θέση του παίκτη με βάση τους κανόνες που θέσαμε εν συναρτήσει με την τιμή του ζαριού.



Η κλάση Player αντιπροσωπεύει τους παίκτες του παιχνιδιού και διαχειρίζεται πληροφορίες που αφορούν τους παίκτες. Προσθέτει (addPlayers()), αφαιρεί (removeAllPlayers()) όλους τους παίκτες από το ταμπλό στο τέλος του παιχνιδιού, ελέγχει την θέση τους σε κάθε γύρο (checkCurrentState()) και εμφανίζει τον νικητή (isWinner()) ανάλογα με τον κανόνα που τον ορίζει.





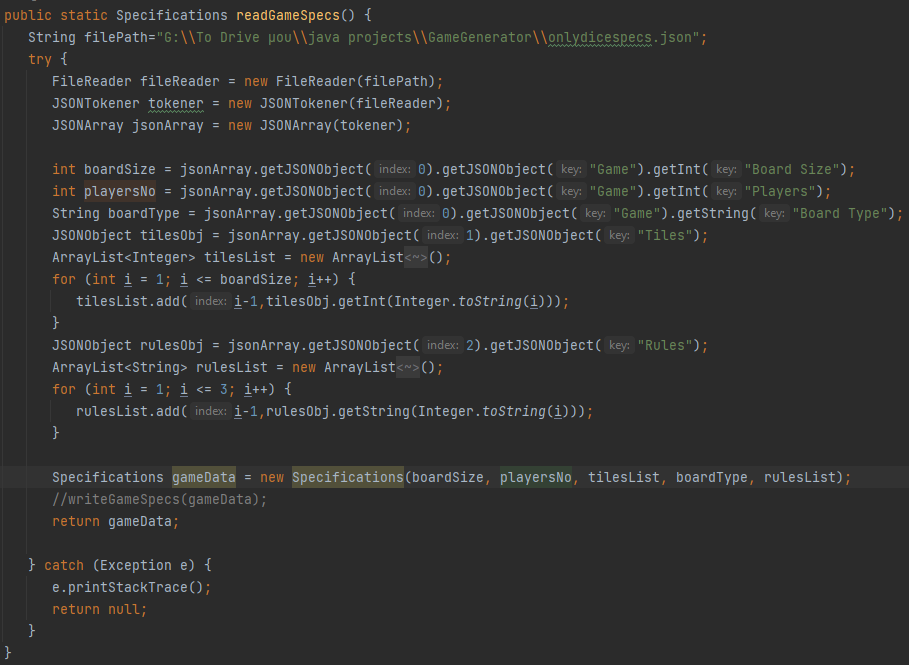
H κλάση Dice εκπροσωπεύει το ζάρι ή τα ζάρια του παιχνιδιού, ανάλογα τον τύπο του ταμπλό και επιστρέφει την τιμή του με την συνάρτηση roll(), όταν δηλαδή είναι η σειρά ενός παίκτη για να παίξει.



Η κλάση Tiles δέχεται τα δεδομένα για τα τετράγωνα ενός ταμπλό, δηλαδή τις ιδιότητες αυτών από το αρχείο json με την κλήση της συνάρτησης readGameSpecs() της κλάσης Specifications.

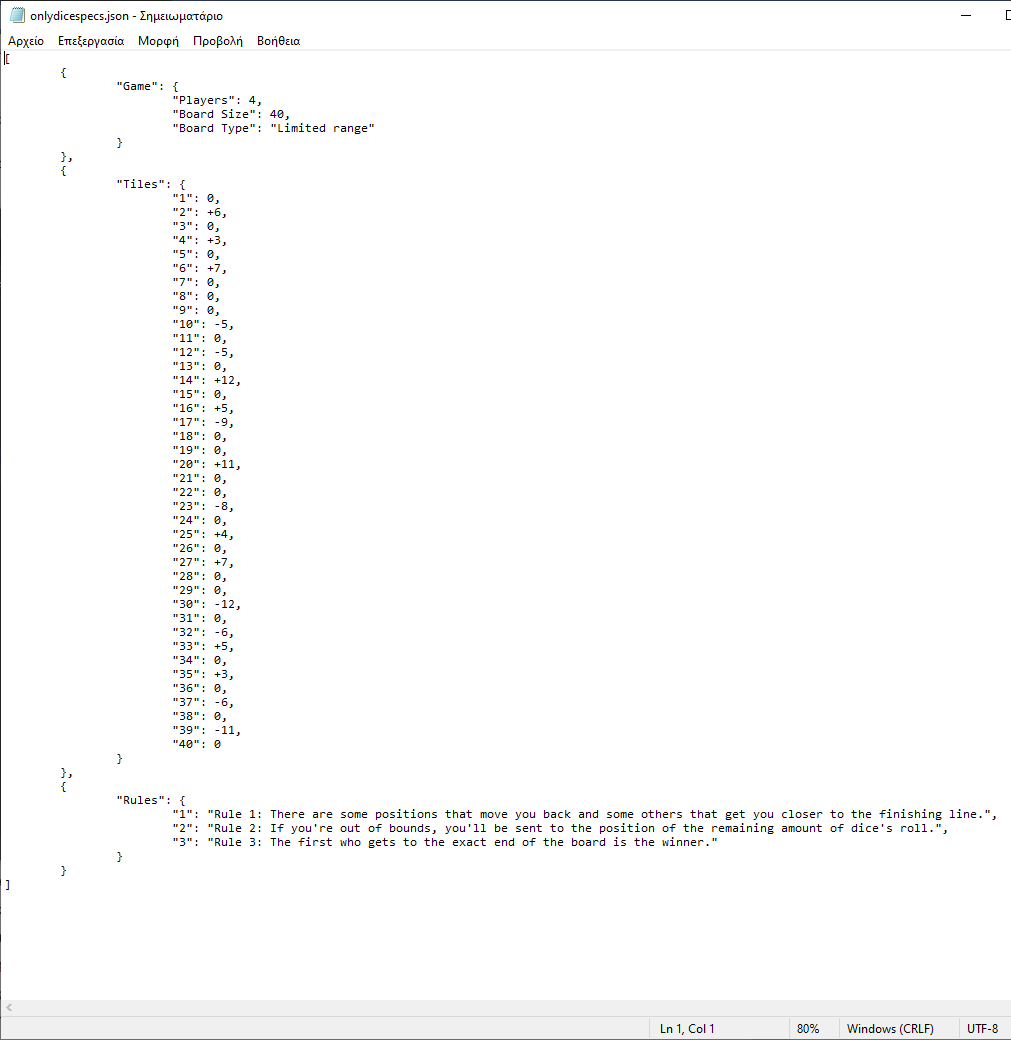


Η κλάση Specifications είναι η κλάση, μέσα στην οποία γίνεται το «διάβασμα» των δεδομένων μέσω της συνάρτησης readGameSpecs(), η οποία επιστρέφει αντικείμενο, ώστε να πάρουμε μεμονωμένα τα χαρακτηριστικά που θέλουμε με getters. Στην readGameSpecs(), το αρχείο json διαβάζεται με τη βοήθεια του File Reader και της JSONTokener κλάσης από το πακέτο org.json, η οποία δέχεται string και μετατρέπει την πληροφορία σε χαρακτήρες.



Οι τιμές των boardSize, playersNo, boardType είναι απλές μεταβλητές και οι tilesList και rulesList είναι λίστες που διαβάζονται από το onlydicespecs.json. Έχει αναφερθεί παραπάνω η κάθε λειτουργία που αντιπροσωπεύουν, το καθένα για την δική του κλάση. Ο τρόπος με τον οποίο εισάγονται τα δεδομένα στο πρόγραμμα είναι πολύ εύκολα κατανοήσιμος και μπορεί να επαναλαμβάνεται για πολλές περιπτώσεις, πχ για την προσθήκη νέων λειτουργιών ή νέου παιχνιδιού. Όπως φαίνεται και από τις φωτογραφίες, οι παράμετροι που δημιουργήσαμε, εισάγονται στις αντίστοιχες κλάσεις σε συναρτήσεις που ξεκινάνε με την λέξη «*import*». Ο λόγος ύπαρξης αυτών των συναρτήσεων είναι για να «κρατούνται» οι τιμές των παραμέτρων στις αντίστοιχες κλάσεις τους, ώστε να μην καλείται το αρχείο κάθε φορά.

Μορφή αρχείου json:



# 3. Χρονοδιάγραμμα

Διαχωρισμός εργασιών:

Αλκίνοος: Συνδέθηκε το BoardType (ArrayList<String>) στο onlydicespecs.json με τη readGameSpecs() που επιστρέφει αντικείμενο των χαρακτηριστικών τα οποία μπορούμε να έχουμε πρόσβαση το καθένα ξεχωριστά με getters (σύνδεση όπως την tilesList με παρόμοιο τρόπο, λίγες αλλαγές) και στέλνεται στη κλάση Board που θα παίρνει το type με παρόμοιο τρόπο όπως στην Tiles (importTiles()) με getter της Specifications (importBoardType()). Όταν ο παίκτης επιλέξει να παίξει τότε οι συναρτήσεις: createGame(), startGame(), playGame(), movePosition() θα παίρνουν αυτό ως ένα επιπλέον όρισμα (String type) στη δήλωση τους (και όταν καλούνται με ανάλογο τρόπο καλώντας μέσα στις παρενθέσεις ένα gameData.getBoardType() ) το οποίο με μια δομή if else θα καθορίζει τις ενέργειες (π.χ. κίνηση). Προς το παρόν το else θα είναι κενό.

Δημήτρης: Αλλαγή στις τιμές των tiles βάζοντας περισσότερες σκάλες και φίδια με την προϋπόθεση ότι όταν ο παίκτης μετακινηθεί θα αλλάξει η θέση του μια φορά είτε πατήσει σε φίδι είτε σε σκάλα. Με βάση το BoardType, στέλνεται στη roll() της Dice το if Board.getBoardType().equals(...) έτσι ώστε να αλλάζει ο αριθμός ζαριών ανάλογα τον τύπο του παιχνιδιού.

Στέλλα: Δημιουργήθηκε καινούρια κλάση Rules η οποία παίρνει τους κανόνες από το json αρχείο με getters η οποία επιστρέφει αντικείμενο των χαρακτηριστικών τα οποία μπορούμε να έχουμε πρόσβαση το καθένα ξεχωριστά (σύνδεση όπως την tilesList με παρόμοιο τρόπο). Οι κανόνες θα τυπώνονται πριν το κάλεσμα της createGame στην κύρια κλάση GenerateGame.

Ελένη: Μοίρασμα εργασιών στα μέλη της ομάδας. Δημιουργία services εν μέσω interfaces και packages.

Αναλυτικά:

2/4/2023:

Δόθηκαν εργασίες.

3/4/2023:

Αλκίνοος : προσθήκη της παραμέτρου του είδους του ταμπλό του παιχνιδιού (Board Type) στο αρχείο json, δημιουργία λίστας (ArrayList) για αποθήκευση των τιμών της παραμέτρου Board Type από το αρχείο json και ενημέρωση των συναρτήσεων createGame(), playGame(), movePosition() ετσι ώστε να λειτουργούν με τάδε Board Type.

Ελένη: Ολοκλήρωση δημιουργίας services εν μέσω interfaces τα οποία θα μας βοηθάνε να καλούμε συναρτήσεις ιδίων ονομάτων με διαφορετική συμπεριφορά ανάλογα το αντικείμενο της κλάσης που καλείται κάνοντας τες επαναχρησιμοποιήσιμες.

4/4/2023:

Αλκίνοος : βελτίωση του κώδικα της 3/4/2023 και παροχή βοήθειας στον Δημήτρη.

Δημήτρης: προσαρμογή της συνάρτησης roll() ώστε να έχει ένα ζάρι στο παιχνίδι με αρχή-τέλος και 2 ζάρια στο κυκλικό.

5/4/2023:

Στέλλα: Υλοποίηση της κλάσης Rules, και προσαρμογή του κώδικα έτσι ώστε να τυπώνει τους κανόνες που υπάρχουν μέσα στο onlydicespecs.json.

Ελένη: Διορθώσεις και προσθήκες στο πρόγραμμα για να δουλέψει σωστά με services (όταν δημιουργηθεί κι άλλο παιχνίδι με παρόμοιες λειτουργίες) και δημιουργία πακέτου (package game).

Αλκίνοος : υλοποίηση του κώδικα τις 3/4/2023 και 4/4/2023 αλλά έγινε αντικατάσταση της ArrayList με μια απλή μεταβλητή τύπου String (boardType) (γιατί θα δημιουργηθεί άλλο αρχείο json για άλλο παιχνίδι) και υλοποιήθηκαν οι αναγκαίες αλλαγές στον υπόλοιπο σχετικό κώδικα για την υποστήριξη και καλή λειτουργία της αλλαγής.

Στέλλα: Σύνταξη αρχικής μορφής του report αφού ολοκληρώσαμε το πρόγραμμα αυτής της εβδομάδας.

6/4/2023:   
Ελένη, Στέλλα: Ολοκλήρωση του report.

Ελένη: Ολοκλήρωση του παρόντος Project.

# 4.Αποτελέσματα-Επίλογος

Επιλογικά, την δεύτερη εβδομάδα εμπλουτίσαμε τον κώδικα του παιχνιδιού έτσι ώστε να είναι περισσότερο επεκτάσιμο. Πλέον, το παιχνίδι μας τύπου «Φιδάκι» είναι έτοιμο και είναι δομημένο σύμφωνα με τις οδηγίες της εκφώνησης του project. Τις επόμενες εβδομάδες το πρόγραμμά μας θα επεκταθεί με τον δεύτερο τύπου παιχνιδιού, τύπου «Μονόπολυ», και ίσως προστεθούν κάποιες έξτρα λειτουργίες, όπως προσωρινή αποθήκευση, ένα αρχικό μενού και καλύτερη και πιο ομαλή διαχείριση των διαφόρων σφαλμάτων που μπορεί να προκύψουν κατά τη διάρκεια της εκτέλεσης του προγράμματος.

# 5.Βιβλιογραφία

|  |  |  |
| --- | --- | --- |
| [5] | JSONTokener Examples Guide  <https://www.tabnine.com/code/java/classes/org.json.JSONTokener> | |
| [6] | Java Iterator – w3schools tutorial  <https://www.w3schools.com/java/java_iterator.asp> | |
| [7] | Java packages tutorial  <https://www.youtube.com/watch?v=NZ7NfZD8T2Y&ab_channel=BroCode> | |
| [8] | Java Interface tutorial - example  <https://www.youtube.com/watch?v=GhslBwrRsnw&ab_channel=BroCode> | |
| [9] | Package java.io summary and explanation  <https://docs.oracle.com/javase/7/docs/api/java/io/package-summary.html> | |
| [10] | | Roll two dice java program  <https://stackoverflow.com/questions/19482283/roll-two-dice-java-program> | |
|  |  | |