Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2025-2026

ΑΝΑΦΟΡΆ ΑΜΦΊΠΟΛΗΣ (ΦΆΣΗ Β)

ΧΡΙΣΤΟΦΟΡΆΚΗΣ ΧΡΙΣΤΌΦΟΡΟΣ

CSD 5809

29/11/2025 (Phase A)

2/1/2026 (Phase B)

Jar αρχείο

Περιεχόμενα

1. Εισαγωγή

Για την υλοποίηση του project χρησιμοποιήσαμε το μοντέλο MVC (Model-View-Controller). Η επιλογή αυτή έγινε για να μπορέσουμε να διαχωρίσουμε τη λογική και τα δεδομένα του παιχνιδιού (Model) από τα γραφικά (View) και το ν έλεγχο ροής (Controller), διευκολύνοντας έτσι την ανάπτυξη και συντήρηση του κώδικα.

Στις επόμενες ενότητες περιγράφονται οι κλάσεις που δημιουργήθηκαν για κάθε πακέτο του MVC και αναλύεται ο ρόλος της καθεμίας. Τέλος παρουσιάζονται τα : Διάγραμμα κλάσεων (UML) το οποίο αποτυπώνει την ιεραρχία και το πως αλληλεπιδρούν οι κλάσεις μεταξύ τους καθώς και τα συμπεράσματα που προέκυψαν μετά την εκπόνηση της εργασίας αυτής.

2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Στο πακέτο `model` περιλαμβάνονται όλες οι κλάσεις που αναπαριστούν τα δεδομένα και την κατάσταση του παιχνιδιού.

1. Ιεραρχία Κλάσεων Πλακιδίων (Tiles): Για την αναπαράσταση των πλακιδίων του παιχνιδιού, δημιουργήθηκε μια ιεραρχία κλάσεων με ρίζα την abstract κλάση `Tile`.

- **`Tile` (Abstract):** Η βασική κλάση που περιέχει τα κοινά χαρακτηριστικά όλων των πλακιδίων, δηλαδή το μοναδικό αναγνωριστικό (`id`) και τη διαδρομή της εικόνας (`imagePath`). Ορίστηκε ως abstract καθώς δεν χρειάζεται/απαιτέιται η δημιουργία γενικού αντικειμένου Tile στο παιχνίδι. Περιέχει επίσης την abstract μέθοδο `getColor()`.
- **`LandslideTile`:** Κληρονομεί από την `Tile` και αναπαριστά τις πέτρες κατολίσθησης.
- **`FindingTile` (Abstract):** Μια ενδιάμεση abstract κλάση που ομαδοποιεί όλα τα ευρήματα (Findings). Δημιουργήθηκε για να διαχωριστούν τα αντικείμενα που δίνουν πόντους από τις κατολισθήσεις.
    - **`MosaicTile` και `AmphoraTile`:** Κληρονομούν από τη `FindingTile` και διαθέτουν επιπλέον πεδίο για το χρώμα (`color`), το οποίο είναι απαραίτητο για τον υπολογισμό των βαθμών.
    - **`SkeletonTile`:** Διαθέτει τα boolean πεδία `isBig` και `isUpper` για να προσδιοριστεί αν ο σκελετός είναι ενήλικας/παιδί και πάνω/κάτω μέρος, σύμφωνα με τους κανόνες του παιχνιδιού.
    - **`StatueTile` (Abstract):** Ομαδοποιεί τα αγάλματα και κληρονομείται από τις κλάσεις `SphinxTile` και `CaryatidTile`, οι οποίες επιστρέφουν τον τύπο τους μέσω της μεθόδου `getColor()`.

2. Ιεραρχία Κλάσεων Χαρακτήρων (Characters): Για τις κάρτες χαρακτήρων, δημιουργήθηκε η abstract κλάση `Character`, η οποία περιέχει το όνομα του χαρακτήρα και το πεδίο `isUsed` για να ελέγχεται αν έχει χρησιμοποιηθεί η κάρτα.

- Κάθε χαρακτήρας (`Assistant`, `Archaeologist`, `Digger`, `Professor`, `TheCoder`) υλοποιείται ως ξεχωριστή υποκλάση που κληρονομεί από την `Character`.
- Όλες οι υποκλάσεις υλοποιούν την abstract μέθοδο `ability()`, η οποία θα περιέχει στη Φάση Β τη λογική της ειδικής ικανότητας του κάθε χαρακτήρα (π.χ. τράβηγμα επιπλέον πλακιδίων).

3. Κλάσεις Διαχείρισης Παιχνιδιού (Board, Player, Bag)

- **`Board`:** Αναπαριστά το ταμπλό και περιέχει 5 λίστες (`ArrayList<Tile>`), μία για κάθε περιοχή (Μωσαϊκά, Αγάλματα, Αμφορείς, Σκελετοί, Είσοδος). Διαθέτει μεθόδους όπως η `addTileToArea()` για την τοποθέτηση πλακιδίων και η `isEntranceFull()` για τον έλεγχο τερματισμού του παιχνιδιού.
- **`Player`:** Κρατάει την κατάσταση του παίκτη (όνομα, σκορ, συλλογή πλακιδίων, διαθέσιμοι χαρακτήρες). Περιλαμβάνει τη μέθοδο `calculateScore()`, η οποία στη Φάση Β θα υλοποιήσει τον αλγόριθμο βαθμολόγησης. Επίσης, έχουν προστεθεί πεδία για την υποστήριξη του χαρακτήρα "The Coder" (flag ενεργοποίησης για τον επόμενο γύρο).
- **`Bag`:** Αναπαριστά τη σακούλα με τα πλακίδια. Περιέχει τη μέθοδο `drawTiles(int count)`, η οποία θα υλοποιηθεί ώστε να επιστρέφει τυχαία πλακίδια και να τα αφαιρεί από τη σακούλα.

3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Το πακέτο `controller` περιέχει την λογική της εφαρμογής και είναι υπεύθυνο για τον συντονισμό της ροής του παιχνιδιού. Η μοναδική κλάση που σχεδιάστηκε σε αυτό το πακέτο είναι η `Controller`, η οποία λειτουργεί ως ο ενδιάμεσος κρίκος μεταξύ του Model και του View.

1. Μέθοδοι και Λειτουργικότητα (Σχεδιασμός για Φάση Β): Στη Φάση Α ορίστηκε ο σκελετός της κλάσης `Controller`. Οι βασικές μέθοδοι που σχεδιάζονται να υλοποιηθούν πλήρως κατά τη Β Φάση είναι οι εξής:

- **`Controller()` (Constructor):** Θα είναι υπεύθυνος για την αρχικοποίηση του παιχνιδιού (`initGame`). Συγκεκριμένα, θα δημιουργεί τα αντικείμενα `Board`, `Bag` και `Player`, θα μοιράζει τις αρχικές κάρτες χαρακτήρων και θα καθορίζει τυχαία τη σειρά των παικτών.
- **`nextTurn()`:** Αυτή είναι η κεντρική μέθοδος διαχείρισης της ροής.
    - Θα ελέγχει ποιος παίκτης έχει σειρά.
    - Θα επαναφέρει τα flags του παίκτη (π.χ. `hasPlayed`, `isUsed` για κάρτες που ισχύουν για έναν γύρο).
    - Θα διαχειρίζεται το ability του χαρακτήρα The Coder (δίνοντας τα επιπλέον πλακίδια στον παίκτη αν είχε ενεργοποιηθεί στον προηγούμενο γύρο).
- **`checkWinCondition()` (helper function):** Θα καλείται στο τέλος κάθε γύρου για να ελέγξει αν η περιοχή εισόδου του ταμπλό έχει γεμίσει με κατολισθήσεις (μέσω της μεθόδου `isEntranceFull()` του Model), σηματοδοτώντας το τέλος του παιχνιδιού.

2. Αλληλεπίδραση Model-View-Controller Η επικοινωνία μεταξύ των τριών μερών θα γίνεται ως εξής:

1. **View με Controller:** Ο χρήστης αλληλεπιδρά με το γραφικό περιβάλλον (π.χ. πατάει το κουμπί Draw Tiles). Το View ειδοποιεί τον Controller για το συμβάν (Event).
2. **Controller με Model:** Ο Controller λαμβάνει το αίτημα και καλεί την αντίστοιχη μέθοδο στο Model (π.χ. `bag.drawTiles()`, `board.addTileToArea()`). Το Model ενημερώνει την κατάστασή του (π.χ. μειώνονται τα πλακίδια στη σακούλα).
3. **Controller με View:** Αφού ενημερωθεί το Model, ο Controller ζητά από το View να σχεδιάσει ξανά το ταμπλό ή να εμφανίσει τα νέα πλακίδια του παίκτη, ώστε ο χρήστης να βλέπει την τρέχουσα κατάσταση του παιχνιδιού.

3. **Υλοποίηση Λογικής (Φάση Β):** Η κλάση `Controller` αποτελεί τον συνδετικό κρίκο μεταξύ του Μοντέλου και της Γραφικής Διεπαφής. Στη Β' Φάση υλοποιήθηκαν όλες οι απαραίτητες μέθοδοι για την τήρηση των κανόνων και τη ροή του παιχνιδιού.

Βασικές Μέθοδοι και Λειτουργίες:

- **`Controller()` & `initListeners()`:** Στον constructor γίνεται η αρχικοποίηση των δομών (`Board`, `Bag`, `Players`) και η δημιουργία του παραθύρου. Η `initListeners()` αναθέτει τους `ActionListeners` στα κουμπιά του View, συνδέοντας τα γραφικά γεγονότα (clicks) με τη λογική του παιχνιδιού.
- **`startTurn()` / `endTurn(boolean force)`:** Ελέγχουν την εναλλαγή των γύρων, ενεργοποιούν/απενεργοποιούν τα κουμπιά ανάλογα με τη σειρά του παίκτη και διαχειρίζονται τον χρόνο (Timer).

- **`drawTilesForCurrentPlayer()`:** Διαχειρίζεται την άντληση πλακιδίων από τη σακούλα, την τοποθέτησή τους στο ταμπλό και τον έλεγχο για το αν τραβήχτηκε πλακίδιο κατολίσθησης.
- **`selectFromArea(int areaIndex)`:** Υλοποιεί τον περιορισμό "μέχρι 2 πλακίδια από μία περιοχή", ελέγχοντας την εγκυρότητα της κίνησης πριν ενημερώσει το Model.
- **`handleThiefTurn()` (Solo Mode):** Ειδική μέθοδος που καλείται όταν εμφανιστεί κατολίσθηση στο παιχνίδι ενός παίκτη. Αφαιρεί αυτόματα όλα τα ευρήματα από το ταμπλό και ενημερώνει το σκορ του Κλέφτη.
- **`useCharacter(int index)`:** Ενεργοποιεί την ειδική ικανότητα του επιλεγμένου χαρακτήρα (π.χ. Assistant, Archaeologist), καλώντας την αντίστοιχη μέθοδο στο Model.
- **`handleGameOver()`:** Ελέγχει τις συνθήκες τερματισμού (γεμάτη είσοδος), υπολογίζει τα τελικά σκορ και ανακηρύσσει τον νικητή.

4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Το πακέτο `view` είναι υπεύθυνο για τη γραφική διεπαφή χρήστη (GUI) και την αναπαράσταση παιχνιδιού. Σχεδιάστηκε ώστε να λειτουργεί αποκλειστικά ως επίπεδο παρουσίασης, χωρίς να περιέχει λογική κανόνων, σύμφωνα με το MVC.

Κλάσεις και GUI (Σχεδιασμός για Φάση Β): Στη Φάση Α δημιουργήθηκε η κεντρική κλάση `GameWindow`, η οποία θα αποτελέσει τον βασικό μέρος του παραθύρου της εφαρμογής.

- **Λειτουργία:** Στη Φάση Β, η κλάση αυτή θα υλοποιηθεί χρησιμοποιώντας τη βιβλιοθήκη Java Swing και θα κληρονομεί από την κλάση `JFrame`.

- **Δομή Παραθύρου:** Το γραφικό περιβάλλον σχεδιάζεται να περιλαμβάνει τρία κύρια τμήματα:
    - **Το Ταμπλό (Board Panel):** Θα παίρνει το μεγαλύτερο μέρος του παραθύρου. Για την υλοποίησή του θα χρησιμοποιηθεί η κλάση `JLayeredPane`, ώστε να είναι δυνατή η τοποθέτηση των εικόνων των Tiles πάνω ακριβώς από τις αντίστοιχες περιοχές της εικόνας του background (Μωσαϊκά, Αγάλματα, κ.λπ.).

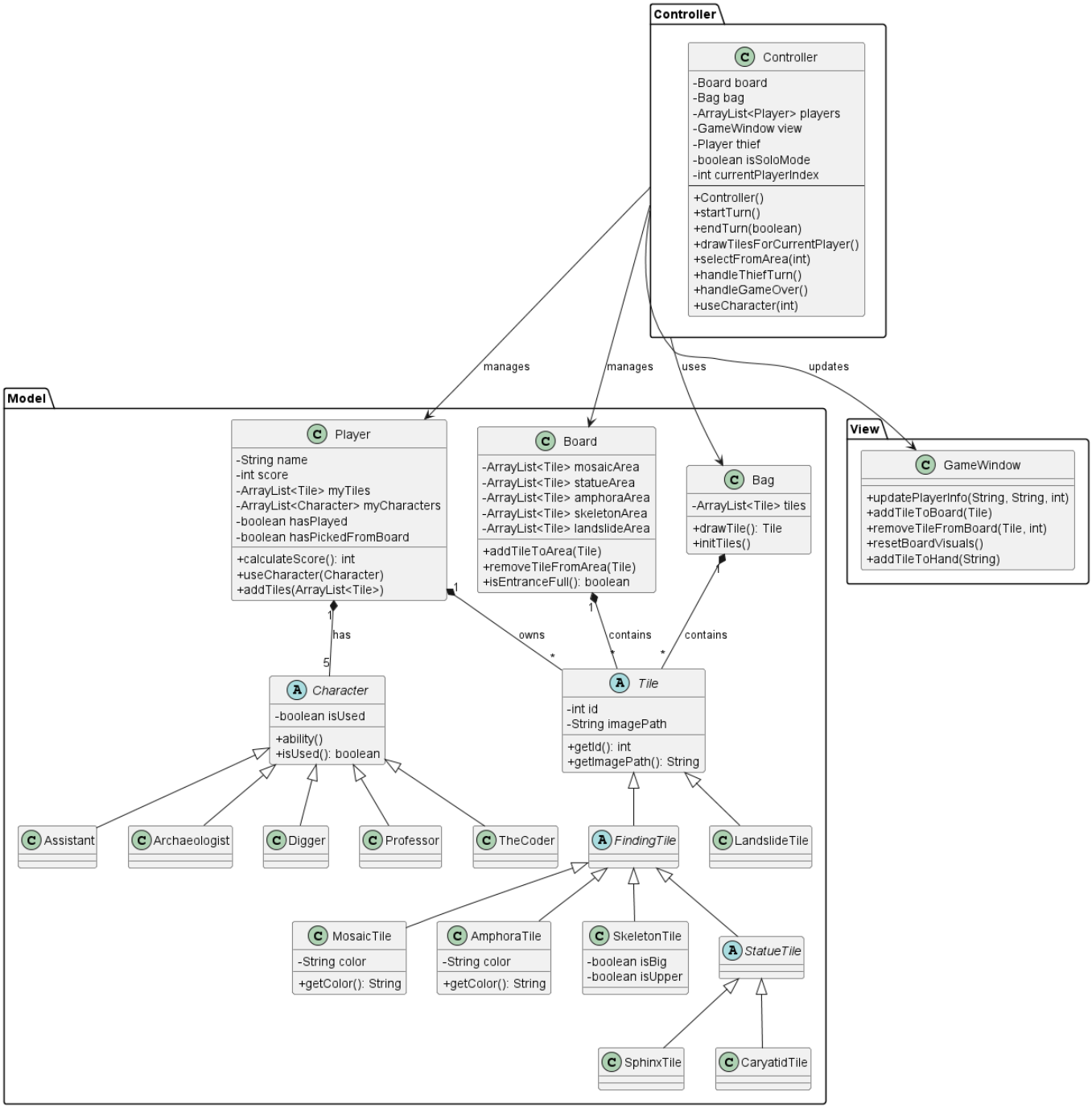
- **Υλοποίηση Λογικής (Φάση Β):** Η κλάση `GameWindow` υλοποιήθηκε με χρήση της βιβλιοθήκης **Java Swing**, παρέχοντας δυναμική οπτική αναπαράσταση της κατάστασης του παιχνιδιού.

  Βασικές Μέθοδοι και Συστατικά:

  - **`GameWindow()` (Constructor):** Ρυθμίζει το βασικό πλαίσιο (`JFrame`), ορίζει τις διαστάσεις και φορτώνει την εικόνα φόντου.
  - **`initComponents()`:** Δημιουργεί και τοποθετεί τα γραφικά στοιχεία (`Buttons`, `Labels`, `Panels`) στις κατάλληλες θέσεις χρησιμοποιώντας `JLayeredPane` για τη σωστή διαστρωμάτωση (Z-order).
  - **`updatePlayerInfo(String name, String color, int score)`:** Ενημερώνει σε πραγματικό χρόνο το πλευρικό πάνελ με τα στοιχεία του τρέχοντος παίκτη και τα πλακίδια που έχει συλλέξει.
  - **`addTileToBoard(Tile t)`:** Δημιουργεί δυναμικά ένα `JLabel` με την εικόνα του πλακιδίου και το προσθέτει στην αντίστοιχη περιοχή του ταμπλό.
  - **`removeTileFromBoard(Tile t, int index)`:** Εντοπίζει και αφαιρεί το γραφικό αντικείμενο ενός συγκεκριμένου πλακιδίου από την οθόνη, κάνοντας `repaint` για να φανεί η αλλαγή.
  - **`resetBoardVisuals()`:** Καθαρίζει μαζικά όλα τα γραφικά πλακίδια από το ταμπλό (χρησιμοποιείται κυρίως στη λειτουργία του Κλέφτη).
  - **`addTileToHand(String imagePath)`:** Εμφανίζει οπτικά τα πλακίδια που μόλις απέκτησε ο παίκτης στην περιοχή του "χεριού" του.

5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διάγραμμα UML

Το παρακάτω διάγραμμα κλάσεων (Class Diagram) απεικονίζει την τελική δομή της εφαρμογής μετά την υλοποίηση της Β' Φάσης.



Το παραπάνω διάγραμμα κλάσεων δείχνει τη συνολική δομή του συστήματος, η οποία είναι οργανωμένη σε τρία πακέτα M-V-C.

Ανάλυση Σχέσεων και Ιεραρχιών :

- **Ιεραρχία Κληρονομικότητας :** Στο πακέτο Model, κυριαρχεί η χρήση της κληρονομικότητας για την οργάνωση των αντικειμένων του παιχνιδιού.
    - Η abstract κλάση `Tile` αποτελεί τη βάση για όλα τα πλακίδια. Από αυτήν προκύπτει η κλάση `LandslideTile` και η ενδιάμεση abstract κλάση `FindingTile`.
    - Η `FindingTile` επεκτείνεται περαιτέρω στις κλάσεις `MosaicTile`, `AmphoraTile`, `SkeletonTile` και στην abstract `StatueTile`, η οποία με τη σειρά της είναι γονέας των `SphinxTile` και `CaryatidTile` . Αυτή η δομή επιτρέπει την πολυμορφική διαχείριση όλων των πλακιδίων μέσω λιστών τύπου `ArrayList<Tile>` στις κλάσεις `Board`, `Player` και `Bag` .
    - Αντίστοιχη ιεραρχία ακολουθείται για τις κάρτες χαρακτήρων, όπου η abstract κλάση `Character` κληρονομείται από τις 5 συγκεκριμένες υλοποιήσεις (Assistant, Archaeologist, Digger, Professor, TheCoder), επιτρέποντας την κοινή διαχείριση της μεθόδου `ability()` .

- **Αλληλεπιδράσεις :** Όπως φαίνεται από τα βέλη αλληλεπίδρασης στο διάγραμμα, η κλάση `Controller` λειτουργεί ως ο κεντρικός κόμβος του συστήματος.
    - Ο Controller έχει άμεση αλληλεπίδραση με το Model (μέσω των κλάσεων `Board` και `Player`), ώστε να μπορεί να καλεί μεθόδους αλλαγής κατάστασης (π.χ. τοποθέτηση πλακιδίου).
    - Ταυτόχρονα, ο Controller συνδέεται με το View (μέσω της κλάσης `GameWindow`), ώστε να ενημερώνει το γραφικό περιβάλλον για τις αλλαγές που συνέβησαν στο Model.

Βασικές Αλλαγές και Σχέσεις (Φάση Β):

1. **Controller:** Παρατηρούμε ότι η κλάση `Controller` έχει εμπλουτιστεί με πεδία όπως `thief` και `isSoloMode`, καθώς και μεθόδους όπως `handleThiefTurn()` και `handleGameOver()`, οι οποίες είναι υπεύθυνες για τη νέα λειτουργικότητα.
2. **View (GameWindow):** Εμφανίζονται πλέον οι μέθοδοι `addTileToBoard` και `removeTileFromBoard`, οι οποίες δείχνουν την αμφίδρομη επικοινωνία με τον Controller για τη δυναμική ενημέρωση των γραφικών.
3. **Σχέσεις (Associations):**
    a. Ο **Controller** έχει κεντρικό ρόλο, κατέχοντας αναφορές (instances) προς το `Board`, τη `Bag`, τους `Players` και το `GameWindow`.
    b. Η σχέση σύνθεσης (Composition) μεταξύ `Player` και `Character` δείχνει ότι κάθε παίκτης έχει το δικό του σετ καρτών.
    c. Η ιεραρχία των `Tiles` και `Characters` φαίνεται ξεκάθαρα μέσω της κληρονομικότητας (extends).

6. Λειτουργικότητα (Β Φάση)

6.1 Ροή του Παιχνιδιού Κατά την εκκίνηση, ο χρήστης επιλέγει μέσω διαλόγου αν επιθυμεί παιχνίδι **4 Παικτών** ή **Solo Mode**.

- Το σύστημα μοιράζει αυτόματα 4 πλακίδια σε κάθε παίκτη και 5 κάρτες χαρακτήρων.
- Ο παίκτης πατάει "Draw Tiles" για να τραβήξει από τη σακούλα. Τα ευρήματα τοποθετούνται στο ταμπλό, ενώ οι κατολισθήσεις στην είσοδο.

6.2 Υλοποίηση Solo Mode & The Thief Σύμφωνα με τις προδιαγραφές, υλοποιήθηκε πλήρως η παραλλαγή για έναν παίκτη:

- **Προετοιμασία:** Τοποθετούνται εξαρχής 8 πλακίδια κατολίσθησης στην είσοδο.
- **Ο Κλέφτης:** Αν ο παίκτης τραβήξει κατολίσθηση, χάνει τη σειρά του και ενεργοποιείται ο "Κλέφτης", ο οποίος αφαιρεί όλα τα διαθέσιμα ευρήματα από το ταμπλό.
- **Νίκη/Ήττα:** Στο τέλος, ο παίκτης κερδίζει μόνο αν το σκορ του ξεπεράσει το σκορ του Κλέφτη.

6.3 Έλεγχος Ορθότητας (Unit Testing) Για την πιστοποίηση της ορθής λειτουργίας, δημιουργήθηκε η κλάση ελέγχου `AmphipolisTest` με χρήση της βιβλιοθήκης **JUnit 4**. Πραγματοποιούνται 5 βασικοί έλεγχοι:

1. **Μωσαϊκά:** Έλεγχος βαθμολογίας για 4άδες ίδιου χρώματος.
2. **Αμφορείς:** Έλεγχος βαθμολογίας για συλλογή διαφορετικών χρωμάτων.
3. **Σκελετοί:** Έλεγχος αναγνώρισης "οικογενειών" (2 Μεγάλοι + 1 Μικρός).
4. **Τερματισμός:** Έλεγχος ότι το παιχνίδι αναγνωρίζει πότε η είσοδος γέμισε (16 πλακίδια).
5. **Χαρακτήρες:** Έλεγχος ότι η χρήση κάρτας ενημερώνει σωστά την κατάσταση (`isUsed`).

6.4 Επιπλέον Χαρακτηριστικά

- Ενσωματώθηκε ηχητική επένδυση (μουσική) κατά τη διάρκεια του παιχνιδιού.
- Όλος ο κώδικας έχει τεκμηριωθεί με σχόλια μορφής **Javadoc** (Pre/Post conditions).

7. Συμπεράσματα

Στα πλαίσια της Α' Φάσης της εργασίας, ολοκληρώθηκε επιτυχώς η σχεδίαση του παιχνιδιού "Αμφίπολη" με βάση τις αρχές του OOP και το MVC επέτρεψε τον διαχωρισμό των αρμοδιοτήτων για την ανάπτυξη του κώδικα.

Ιδιαίτερη έμφαση δόθηκε στη σωστή οργάνωση των κλάσεων σε ιεραρχίες, ειδικά για τα Πλακίδια (Tiles) και τους Χαρακτήρες (Characters), αξιοποιώντας την κληρονομικότητα και τις abstract κλάσεις για να αποφευχθεί η επανάληψη κώδικα. Η λεπτομερής καταγραφή των υπογραφών των μεθόδων και των συνθηκών λειτουργίας τους (pre/post-conditions) αναμένεται να διευκολύνει σημαντικά τη διαδικασία της υλοποίησης κατά τη Β' Φάση του έργου.

Με την ολοκλήρωση της Β' Φάσης, αναπτύχθηκε ένα πλήρως λειτουργικό παιχνίδι που σέβεται την αρχιτεκτονική MVC. Η διαχωρισμένη λογική (Model) από την απεικόνιση (View) επέτρεψε την εύκολη ενσωμάτωση του Solo Mode χωρίς αλλαγές στη δομή των κλάσεων, ενώ τα Unit Tests διασφαλίζουν την ορθότητα των πολύπλοκων κανόνων βαθμολόγησης.