ΗΥ-252 Αντικειμενοστρεφής Προγραμματισμός

Προγραμματιστική Εργασία Χειμερινού Εξαμήνου 2014

Mille Bornes

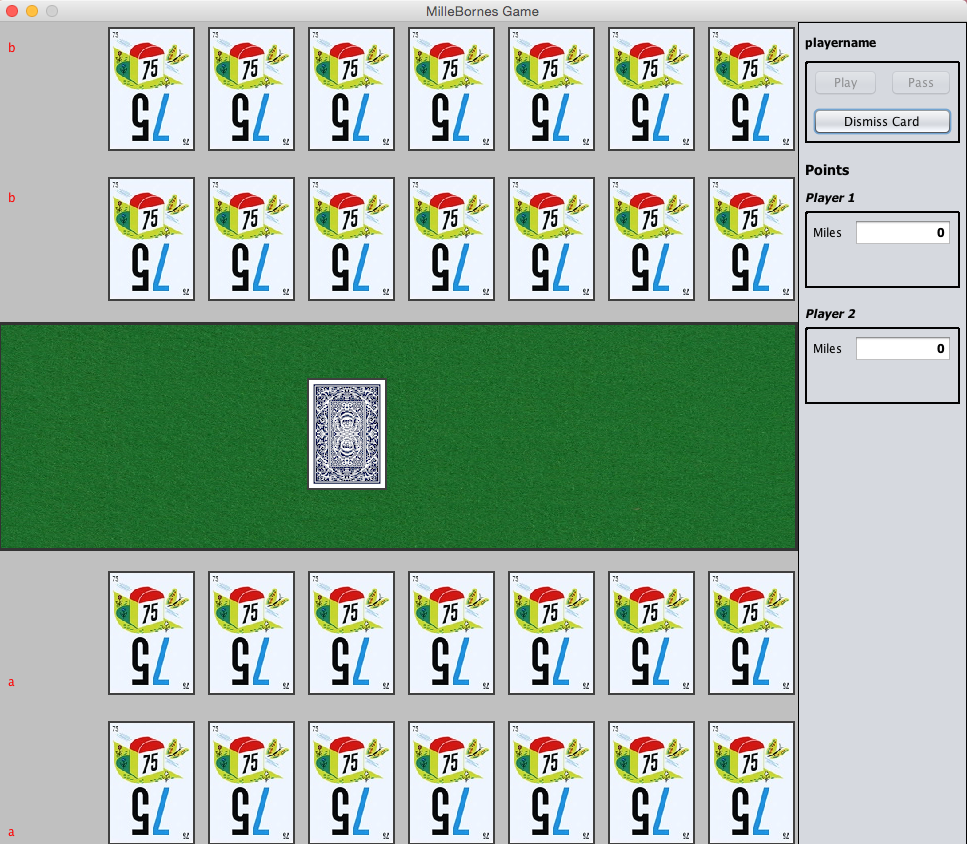


Figure : Game View

Παράδοση 1ης Φάσης

Το Mille Bornes παίζεται με μια ειδική τράπουλα η οποία αποτελείται από φύλλα που έχουν να κάνουν με τους κανόνες οδικής κυκλοφορίας. Για να την αναπαράσταση των φύλλων στον αντικειμενοστραφή προγραμματισμό αλλα και για την ευκολη διαχείριση τους υλοποιήσαμε την αφηρημένη κλάση Card. Με αυτό τον τρόπο δίνουμε την δυνατότητα εξειδίκευσης σε απλη κάρτα όπως τα Distance φύλλα μιας τράπουλας αλλα και σε ειδικες κάρτες (Safety, Hazard, Remedy) από όπου στην συνέχεια εξειδικεύουμε σε κάρτες όπως το AceDriver, Tyre και Gas. Αυτο φαίνεται και στο αντίστοιχο διάγραμμα UML.

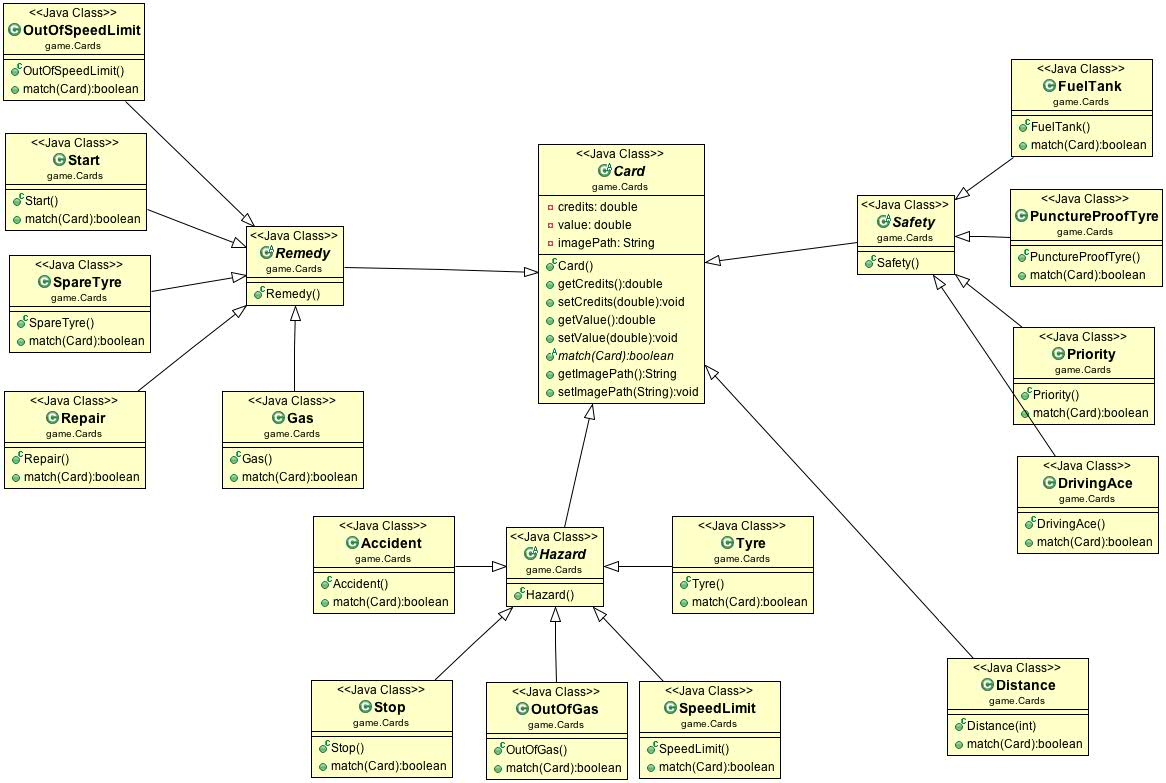


Figure 2: Card

Από τα κοινα χαραστηριστικά των καρτών είναι η αξία της κάρτας (value) που χρειάζεται και για την υλοποίηση της κάρτας Distance. Για παράδειγμα, όταν ο ενας παίκτης έχει οριο ταχύτητας, η τιμή αυτή θα μας βοηθήσει να καταλάβουμε αν ενας παίχτης μπορεί να ρίξει την συγκεκριμένη Distance card η οχι. Να σημειωθεί ότι η κλάσεις Safety, Remedy και Hazard είναι αφηρημένες διότι δεν υπάρχει λόγος αυτόνομης ύπαρξης επιτρέποντας με αυτό τον τρόπο την ευκολότερη διαχείριση και αναγνώριση οποιασδήποτε ειδικής κάρτας από τον πυρήνα του παιχνίδιού.

Άλλο ένα κοινό χαρακτηριστικό όλων των καρτών είναι το image path το οποίο ορίζει που βρίσκεται αποθηκευμένη η εικόνα που αντιπροσωπεύει την κάθε κράτα και θα μας βοηθήσει στην αποτύπωση τους από το View component.

Το παιχνίδι βασίζεται σε συλλογές από κάρτες. Οι ίδιες οι συλλογές αυτές χωρίζονται σε διάφορα είδη. Οι συλλογές αυτές εξειδικεύουν την AbastractPile που είναι αφηρημένη κλάση και βοηθάει να ομαδοποιήσουμε τα κοινά χαρακτηριστικά των Piles. Για τον ίδιο τον παίχτη έχουμε τέσσερις διαφορετικές συλλογές από κάρτες που έχει μπροστά του ανά παστά στιγμή. Ατές είναι οι Battle Pile, SpeedPile, SafetyPile και DistancePile. Επίσης στο τραπέζι (η αλλιώς Deck ο πως θα δούμε μετα) υπάρχουν αλλα δυο είδη καρτών, το Draw Pile από οπίου ο παίχτης μπορεί να τραβήξει κάρτες, και το DistardPile οπου ο χρήστης μπορεί να ῾ξεφορτωθεί᾽ κάρτες! Να σημειωθεί οτι η κλαση AbstractPile είναι αφηρημένη μιας δεν έχει νοημά η αυτόνομη ύπαρξη της αλλα διευκολύνει τον πηρύνα του παιχνιδιού με τον χειρισμό ειδικών συλλογών από καρτες .

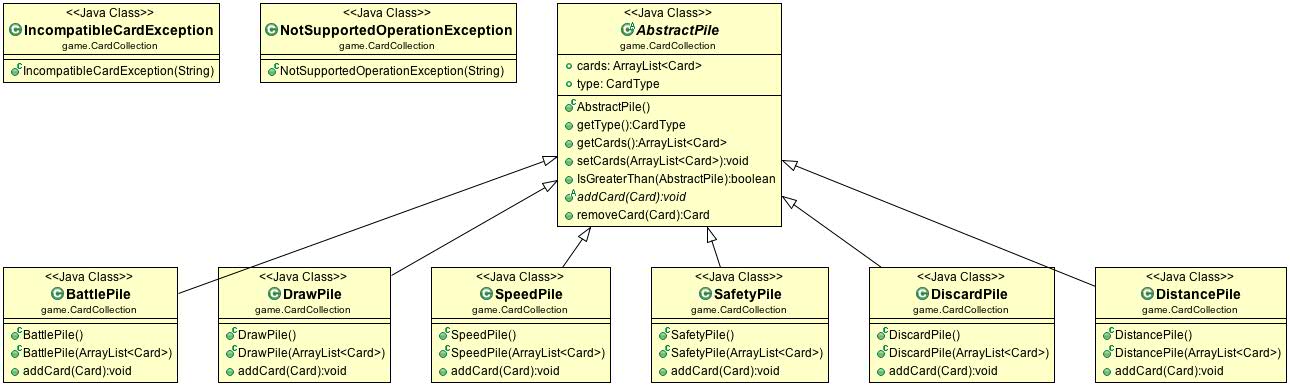


Figure 3: Card Pile

Με αυτή την προσέγγιση γίνεται ευκολότερο για τον πυρήνα του παιχνιδιού να καθορίσει το είδος της συλλογής που χρησιμοποιεί ένας παίκτης όπως επίσης και να ελέγξει αν είναι δυνατό να προστεθεί μια κἀρτα σε μια συγκεκριμένη συλλογή. Η κλάση AbstractPile περιλαμβάνει τα εξής γνωρίσματα:

|  |
| --- |
| private ArrayList<Card> cards;  private CardType type; |

Δηλαδή ένα σύνολο από κάρτες όπως επίσης και το είδος της συλλογής. Το είδος της συλλογής με βάση τους κανόνες του παιχνιδιού μπορεί να είναι:

|  |
| --- |
| public enum CardType{ BATTLE, SPEED, DISTANCE, SAFETY, UNGATEGORIZED, DISCARD, DRAW }; |

Αξίζει να σημειωθεί οτι οι συναρτήσεις addCard και isValidMove είναι πολυμορφικές αφού δεν είχε νόημα να υλοποιηθούν για την abstract κλάση CardPile. Επίσης ἐχουμε ορίσει δυο δικά μας Exceptions το NotSupportedOperationException και IncompatibleCardException για τις δυο αυτἐς μεθόδους σε περίπτωση που κατα την διάρκεια του παιχνιδιού προστεθούν καρτες που δεν θα επρεπε στην συγκεκριμενη συλλογη καρτων.

Με αυτόν τον τρόπο ο πυρήνας του παιχνιδιού μπορεί εύκολα να διαχειριστεί τις συλλογές των καρτών που έχει ενας παικτης με τις ειδικές συλλογές δοκιμάζοντας απλά να δημιουργήσει ενα στιγμιοτυπό της. Αν κατα τη πρόσθεση κάρτας δεν παρουσιαστεί κάποια εξαίρεση (exception) τότε αυτό σημαίνει οτι οι δοθείσες κάρτες μπορούσαν να προστεθούν μια τέτοια ειδική συλλογή. Στην συνέχεια ο πυρήνας του παιχνιδιού μπορεί να ελέγξει αν μια κίνηση ειναι επιτρεπτή χρησιμοποιώντας την εξης μέθοδο:

|  |
| --- |
| public boolean isValidMove(Card tmp) throws IncompatibleCardException{  return false;  } |

Η οποία εξ όρισμου επιστρέφει false αλλα στην υποκλάσεις γίνεται overridden εσωκλειόντας με αυτόν τον τρόπο την λογική σύγκρισης που ορίζουν οι κανόνες του παιχνιδιού. Π.χ. Αν ενας παίχτης εχει στην συλλογή ασφαλείας το άφθαρτο λάστιχο δεν θα μπορεί ο αντίπαλος να του ρίξει την καρτα επίθεσης λάστιχο! Αντίστοιχα αν ο παίχτης έχει δεχθεί ήδη οριο ταχύτητας δεν θα μπορεί να ρίξει καρτας απόστασης πανω απο αυτο το οριο!

Το παιχνίδι παίζεται από δυο παίκτες. Τα βασικά γνωρίσματα ενός παίκτη (πέρα απο το όνομα του) είναι το φύλλο που τράβηξε, τα μίλια που εχει διανύσει και οι τέσσερις στοίβες που τον αναγνωρίζουν (battle, speed, distance, safety) και κάποια επίσης γνώρισματα που χρησιμεύουν στον πυρήνα του παιχνιδιού οπως αν εχει ρίξει κάρτα εκκίνησης.

|  |
| --- |
| private String name;  private int MilesRun;  private AbstractPile hand;    /\*\*  \* Players Piles!  \*/  private AbstractPile battle;  private AbstractPile speed;  private AbstractPile distance;  private AbstractPile satefy;    private Card tmp;  private boolean hasStarted; |

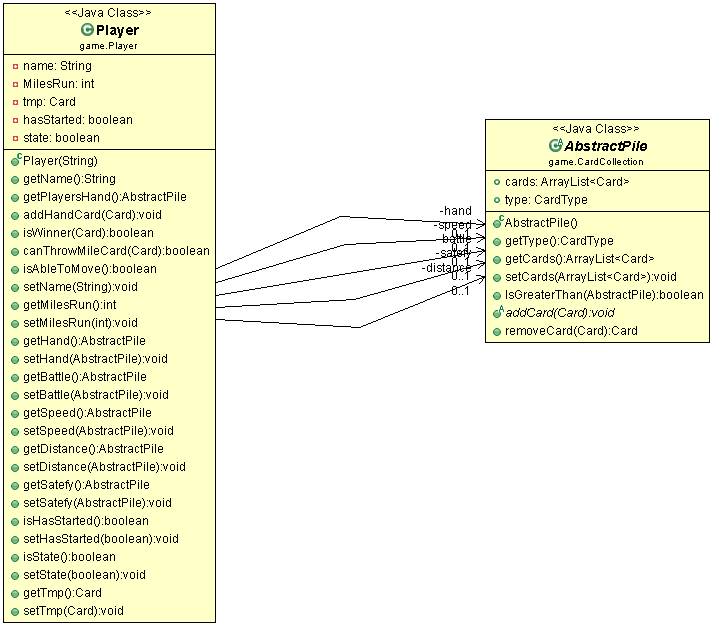


Figure 4: Player

Μια ακόμα σημαντική οντότητα του παιχνιδιού είναι η κλάση Deck η οποία προσομοιώνει το ταμπλό του παιχνιδιού. Το ταμπλό αυτο περιέχει την στοίβα για να τραβάνε κάρτες οι παίχτες (DrawPile) και την στοίβα για να απορρίπτουν κάρτες οι παίχτες (DiscardPile). Επίσης περιέχει και στιγμιότυπα των δυο παιχτών (Player) και μέσω αυτων μπορεί να εχει πρόσβαση σε ολα τα πεδία της κλάσης του κάθε παίχτη. Το διάγραμμα UML αυτής της κλάσης φαίνεται παρακάτω.

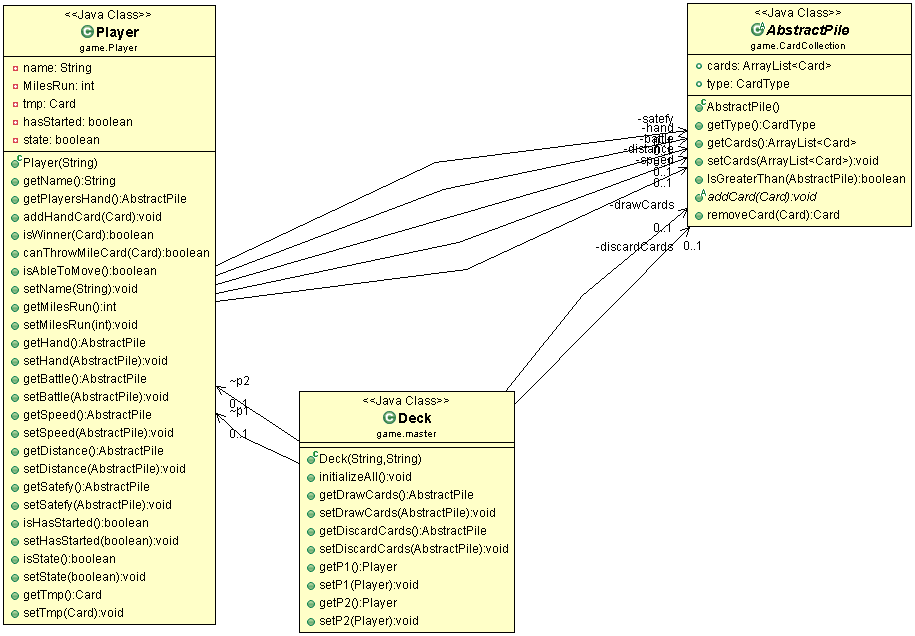


Figure : Deck

Ο πυρήνας του παιχνιδιού (GameMaster) είναι υπεύθυνος για την διεξαγωγή του παιχνιδιού από την έναρξη μέχρι και την λήξη του. Είναι επίσης ο controller αναλαμβάνοντας τον ρόλο ενδιάμεσης επικοινωνίας μεταξύ του View και του Model υιοθετόντας με αυτό τον τρόπο το design pattern MVC. Κάποιες από τις κύριες λειτουργίες του πυρήνα του παιχνιδιού είναι η αποδοχή και ο έλεγχος της κάρτας που ρίχνει ο κάθε παίκτης, τον καθορισμὀ του είδους του (RoundType) καθώς και τον ελέγχο ποτε πρέπει να ξαναπαίξει κάθε παίχτης αλλά και ποτε πρέπει να τερματίσει το παιχνίδι. Βασικά γνωρίσματα της κλάσης αυτής φαίνονται παρακάτω.

|  |
| --- |
| private Player currPlayer;  private GameState state;  private Player Winner;  private Player lastPlayed;  private Deck deck;  JFrame view; |

Η κλάση GameMaster υλοποιεί την διεπαφή Game στην οποία ορίζονται οι βασικές λειτουργίες που επιτελεί σχεδόν κάθε παιχνίδι:

|  |
| --- |
| public interface Game {  public void Start();  public void Stop();  public void Pause();  public void Resume();  public void Restart();  } |

Επίσης ο constructor της GameMaster δεχεται σαν παράμετρο το στιγμιοτυπο του πελάτη (View -Jframe) ο οποίος θα πρέπει να υλοποιεί την ακόλουθη διεπαφή IGameClient προκειμένου να λαμβάνει γνώση της εκαστοτε καταστασης του παιχνιδιού, της κατάστασης του κάθε παίκτη, ποιος παίκτης έχει σειρά να παίξει.

|  |
| --- |
| public interface IGameClient {    public Player showCardsWindow();  public void skipPlayer();  public void changePlayer();  public void newRound();  public void showWinnerPanel(String \_winner);    } |

Ένα άλλο χαρακτηριστικό γνώρισμα της κλάσης GameMaster είναι το State του παιχνιδιού (State) το οποίο εσωτερικά χρησιμοποιείται από το μοιρασμα των φύλλων κατα της έναρξη του παιχνιδιού, αλλα και τον τερματισμο του.

|  |
| --- |
| public enum GameState { UNSPECIFIED, PLAYING, WAITFORNAMES, PAUSED, STOPPED, FINISHED}; |

Η κλάση GameMaster, στον ρόλο του Controller, ενημερώνεται για τα UI events μέσω του public API της όπως για την περίπτωση που 1) ο παίκτης επέλεξε να πει πάσο, ή 2) επέλεξε να κατεβάσει μια καρτα .

|  |
| --- |
| public boolean playerSelectsToPass(Player p)  public boolean playerSubmitsCard (Player p, Card c) |

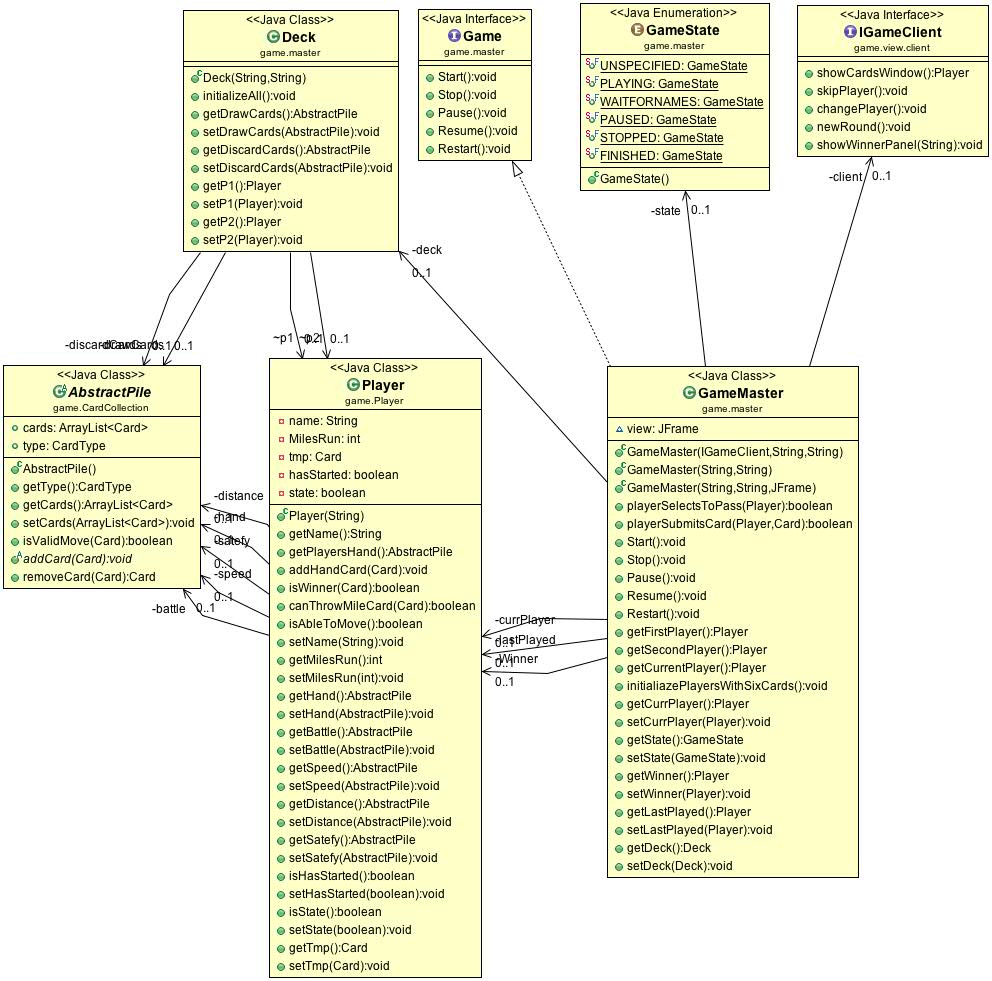


Figure 6: GameMaster