Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2020-2021

Μαστοράκης Εμμανουήλ AM csd5255

Παρουσίαση για την πρώτη φάση του project.

Εισαγωγή

Think and describe what you plan to do and why it will be useful.

Περιεχόμενα

- Εισαγωγή

Η εργασία υλοποιήθηκε με βάση το μοντέλο MVC. Χωρίστηκε σε 3 πακέτα (Model, View, Controller) όπου κάθε ένα έχει και από έναν ξεχωριστό ρόλο στην οργάνωση του κώδικα.
Στο Model έχουμε τα δεδομένα του παιχνιδιού καθώς και την λογική του.
Στο View έχουμε την αναπαράσταση των γραφικών του παιχνιδιού και την υλοποίηση αυτων.
Και τέλος το Controller είναι αυτό που επικοινωνεί με τα δύο παραπάνω πακέτα και ανανεώνει τα γραφικά του παιχνιδιού με βάση τα δεδομένα.

Παρακάτω αρχίζουμε αναλύοντας περισσότερο το τι περιέχει το πακέτο model και επειτα το πακέτο view ωστε να μπορεί να δημιουργηθεί στην Β φάση το επιτραπέζιο παιχνίδι Sorry! .
Τέλος παρουσιάζουμε και το πακέτο Controller μαζι με κάποια UML διαγράμματα.

- Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Στο πακέτο Model υπάρχουν οι κλάσεις που αφορούν την λογική του παιχνιδιού.
έχουμε την κλάση Player, Pawn, Deck, Color, Board, και μετά έχουμε και την αφηρημένη κλάση Square και την αφηρημένη κλάση Card που αυτές οι δύο χωρίζονται σε άλλες υποκλάσεις.

Η αφηρημένη κλάση Card και οι υποκλάσεις της.

- public abstract Card class:

  είναι η κλάση για τις κάρτες του παιχνιδιού.
  Περιέχει μία μεταβλητή string description όπου είναι η περιγραφή του τι κάνει η κάρτα και μία string note που είναι κάποια τυχόν σημείωση που μπορεί να έχει.

  Constructor: public Card(String description, String note)
  όπου κατασκευάζει την κάρτα, με βάση την περιγραφή και σημείωση που έχει δωθεί

  έχει τις παρακάτω μεθόδους:
  public String getDescription() // Getter για την περιγραφή.

  public String getNote() // Getter για την σημείωση.

  public void setDescription(String desc) // Setter για την περιγραφή.

  public void setNote(String note) // Setter για την σημείωση.

  public void execute(Player player, Board board)
  Αυτή την συνάρτηση την έχει κάθε υποκλάση της Card και την κάνει OverRide απο αυτή την κλάση. Είναι η συνάρτηση που παίρνει ως όρισμα τον παίχτη που την τράβηξε και το ταμπλό και απλά εκτελεί την περιγραφή της κάρτας.

- public class NumberCard extend Card:

αυτή η κλάση κάνει extend την κλάση Card και δηλώνει ότι η κάρτα είναι κάρτα που έχει πάνω της αριθμό.

άρα θα έχει σαν attribute έναν ακέραιο αριθμό που δηλώνει τον αριθμό της.

Constructor: public NumberCard(String description, String note, int number), αυτή η κατασκευάστρια μέθοδος καλεί τον γονέα της για την δημιουργία μίας κάρτας μαζί με τον αριθμό που δίνεται.

έχει τις παρακάτω έξτρα μεθόδους:
public int getNumber() // Getter για τον αριθμό της κάρτας

public void execute(Player player, Board board) // Override της Card.

- public class SimpleNumberCard extends NumberCard:
  αυτή η κλάση κάνει extend την NumberCard και δηλώνει ότι η κάρτα έχει αριθμό που έχει απλό κανόνα και δεν αφήνει κάποια επιλογή στον παίχτη για το τί θα κάνει στο παιχνίδι. Έτσι διαχωρίζονται οι κάρτες σε αυτές που αφήνουν επιλογές στους παίχτες και σε αυτές που όχι.

  Constructor: public SimpleNumberCard(String description, String note, int number) καλεί την κατασκευάστρια της υπερκλάσης της.

  έχει τις παρακάτω έξτρα μεθόδους:
  public void execute(Player player, Board board) // Override της Card.

Στην συνέχεια έχουμε τις κλάσεις

- NumberElevenCard extends NumberCard.
- NumberTenCard extends NumberCard.
- NumberSevenCard extends NumberCard.
- NumberFourCard extends NumberCard.
- NumberTwoCard extends NumberCard.

όλες έχουν την δική τους κατασκευάστρια μέθοδο με όρισμα κάποια περιγραφή, κάποια σημείωση και έναν ακέραιο αριθμό.
Κάνουν επίσης όλες override την μέθοδο execute όπου εκτελεί τον κανόνα της κάθε κάρτας. Καθώς οι κάρτες δεν ανοίκουν στην SimpleNumberCard, αυτό σημαίνει ότι κάθε μία αφήνει και κάποια επιλογή στον παίχτη για το πού θα πάει ή μετακινεί παραπάνω από ένα πιόνι.

- public class SorryCard extends Card
  Κάνει extend την κλάση Card και δηλώνει ότι ένα αντικείμενό της είναι κάρτα Sorry. Έτσι δεν έχει attribute number και η κατασκευάστριά της έχει μόνο όρισμα κάποια περιγραφή και κάποια σημείωση.

public SorryCard(String descrpition, String note)

και αυτή η κλάση κάνει override την μέθοδο execute όπου είναι η εκτέλεση της περιγραφής της κάρτας.

public void execute(Player player, Board board) // Override της Card

Η αφηρημένη κλάση Square και οι υποκλάσεις της:

- public abstract class Square

  είναι η κλάση για το κάθε τετράγωνο του ταμπλού.

  Περιέχει τα attributes: int value όπου είναι μία τιμή για το τετραγωνάκι που δείχνει το που βρίσκεται στο ταμπλό και ένα Color color που είναι το χρώμα που αυτό έχει.

  Constructor: public Square(Color color, int value)
  δημιουργεί ένα τετραγωνάκι με το χρώμα και την τιμή που δίνεται.

  Έχει τις παρακάτω μεθόδους:
  public Color getColor() // Getter για το χρώμα που έχει το τετραγωνάκι.
  public void setValue(int value) // Setter για την τιμή που θα έχει το τετραγωνάκι με την τιμή που δίνεται.
  public int getValue() // Getter για την τιμή που έχει το τετραγωνάκι.

- public class SimpleSquare extends Square

  αυτή η κλάση κάνει extend την κλάση Square και απλά δηλώνει ότι το τετραγωνάκι είναι ένα απλό άσπρο τετραγωνάκι.

  Constructor: public Square(Color color, int value)
  δημιουργεί ένα τετραγωνάκι με το χρώμα και την τιμή που δίνεται.

- public class PlayersSquares extends Square

  αυτή η κλάση κάνει extend την κλάση Square και είναι μία κλάση η η οποία δηλώνει ότι το τετραγωνάκι της, είναι ένα τετραγωνάκι που "ανήκει" σε έναν παίχτη. Τετραγωνάκια όπως αυτά που είναι το σπίτι κάποιου παίχτη ή η ασφαλής ζώνη του ανήκουν σε αυτήν την κλάση.

  Έχει attribute private Player player που είναι ο παίχτης στον οποίο ανήκει το τετραγωνάκι.

  Constructor: public PlayersSquares(Color color, int value, Player player)
  καλεί τον κατασκευαστή της υπερκλάσης της και ακόμα αναθέτει στο τετραγωνάκι έναν παίχτη.

  Έχει τις παρακάτω μεθόδους:
  public void setPlayer(Player player) // Setter του παίκτη που θα έχει

το τετραγωνάκι σε αυτόν που δίνεται.

public Player getPlayer() // Getter για τον παίκτη που του ανήκει το τετραγωνάκι.

- public class StartSquare extends PlayersSquares

  αυτή η κλάση δηλώνει ότι ένα τετραγωνάκι είναι το τετραγωνάκι από το

  οποίο θα ξεκινήσει κάποιος παίκτης. Για τον λόγο αυτό κάνει extend την κλάση PlayersSquares.

  Έχει δύο επιπλέον attributes το private Pawn pawn1 και το private Pawn pawn2. Είναι τα πιόνια του παίκτη έτσι ώστε να είναι πιο εύκολα διαθέσιμα και να τα παίρνουμε κατευθείαν από το τετραγωνάκι και όχι απο τον παίκτη.

  Constructor: public StartSquare(Color color, int value, Player player) καλεί τον κατασκευαστή της υπερκλάσης της.

  Έχει τις παρακάτω μεθόδους:
  public void setPawn1(Pawn pawn1) // Setter για το πρώτο πιόνι που θα έχει το τετραγωνάκι με βάση αυτό που δίνεται.
  public void setPawn2(Pawn pawn2) // Setter για το δεύτερο πιόνι που θα έχει το τετραγωνάκι με βάση αυτό που δίνεται.
  public Pawn getPawn1() // Getter για το πρώτο πιόνι που έχει το τετραγωνάκι.
  public Pawn getPawn2() // Getter για το δεύτερο πιόνι που έχει το τετραγωνάκι.

- public class SafetyZoneSquare extends PlayersSquare

  αυτή η κλάση δηλώνει ότι ένα τετραγωνάκι είναι SafeZone κάποιου παίκτη. Για τον λόγο αυτό κάνει extend την κλάση PlayersSquare.

  Έχει ένα attribute που είναι private int value. Δηλαδή έχει την δική του ξεχωριστή τιμή απο αυτή της υπερκλάσης της. Έτσι μπορούμε να καταλάβουμε και που βρίσκεται το πιόνι μέσα στο safe zone του και δεν μπερδευόμαστε με τις τιμές που έχουν άλλα τετράγωνα.

  Constructor: public SafetyZoneSquare(Color color, Player player, int value). Ο κατασκευαστής της κλάσης απλά καλεί τον κατασκευαστή της υπερκλάσης της.

- public class HomeSquare extends PlayersSquares

  αυτή η κλάση δηλώνει ότι ένα τετραγωνάκι το σπίτι ενός παίκτη. Για τον λόγο αυτό κάνει extend την κλάση PlayersSquares.

Έχει δύο επιπλέον attributes το private Pawn pawn1 και το private Pawn pawn2. Είναι τα πιόνια του παίκτη έτσι ώστε να είναι πιο εύκολα διαθέσιμα και να τα παίρνουμε κατευθείαν από το τετραγωνάκι και όχι απο τον παίκτη.

Constructor: public HomeSquare(Color color, int value, Player player) καλεί τον κατασκευαστή της υπερκλάσης της.

Έχει τις παρακάτω μεθόδους:
public void setPawn1(Pawn pawn1) // Setter για το πρώτο πιόνι που θα έχει το τετραγωνάκι με βάση αυτό που δίνεται.
public void setPawn2(Pawn pawn2) // Setter για το δεύτερο πιόνι που θα έχει το τετραγωνάκι με βάση αυτό που δίνεται.
public Pawn getPawn1() // Getter για το πρώτο πιόνι που έχει το τετραγωνάκι.
public Pawn getPawn2() // Getter για το δεύτερο πιόνι που έχει το τετραγωνάκι.

- public abstract class SlideSquare extends Square
  αυτή η κλάση δηλώνει ότι το τετραγωνάκι είναι μία τσουλήθρα.
  Γι αυτό και κάνει extend την κλάση Square. (Δεν ήξερα αν θα ήταν καλύτερο να ανήκει σε κάποιον παίχτη αυτή η τσουλήθρα για να κάνει extend καλύτερα την PlayersSquare... θα δείξει I guess.)

  Constructor: public SlideSquare(Color color, int value)
  Ο κατασκευαστής καλεί απλά τον κατασκευαστή της υπερκλάσης του.

- public class StartSlideSquare extends SlideSquare
  αυτή η κλάση δηλώνει ότι το τετραγωνάκι είναι η αρχή μίας τσουλήθρας. Για αυτόν το λόγο κάνει extend την SlideSquare.

  Constructor: public SlideSquare(Color color, int value)
  Ο κατασκευαστής καλεί απλά τον κατασκευαστή της υπερκλάσης του.

- public class InternalSlideSquare extends SlideSquare
  αυτή η κλάση δηλώνει ότι το τετραγωνάκι είναι η μέση μίας τσουλήθρας. Για αυτόν το λόγο κάνει extend την SlideSquare.

  Constructor: public InternalSlideSquare(Color color, int value)
  Ο κατασκευαστής καλεί απλά τον κατασκευαστή της υπερκλάσης του.

- public class EndSlideSquare extends SlideSquare
  αυτή η κλάση δηλώνει ότι το τετραγωνάκι είναι η το τέλος μίας τσουλήθρας. Για αυτόν το λόγο κάνει extend την SlideSquare.

Constructor: public EndSlideSquare(Color color, int value)
Ο κατασκευαστής καλεί απλά τον κατασκευαστή της υπερκλάσης του.

Η κλάση Board:

- public class Board

Η κλάση αυτή αναπαριστά το ταμπλό του παιχνιδιού. Τιποτά γραφικό γιατί τότε δεν θα υπήρχε νόημα να βρίσκεται στο model. Αλλά η λογική του ταμπλού υλοποιείται στην κλάση αυτή.

Έχει τα εξής Attributes: private static int SIZE_OF_BOARD_X που είναι το μέγεθος του ταμπλού στον οριζόντιο άξονα, private static int SIZE_OF_BOARD_Y που είναι το μέγεθος του ταμπλού στον κάθετο άξονα, private final Square[][] board που είναι το ταμπλό σε ένα διπλό πίνακα με τετραγωνάκια με το μέγεθος που έδωσαν τα παραπάνω attributes και τέλος private Player[] players που είναι ένας πίνακας με τους παίκτες που παίζουν στο παιχνίδι.

Constructor: public Board(Player[] players)
κατασκευάζει το ταμπλό με βάση τους παίκτες που θα παίξουν.

Έχει τις παρακάτω μεθόδους.

public void setPlayers(Player[] players) // Setter for the players in the board with the given Player array.

public Player[] getPlayers() // Getter for the players of the board.
public int[][] getPawnPositions() // Function that returns all positions of the pawns of the pawns that are still playing.

Η κλάση Deck

- public class Deck

Η κλάση αυτή υλοποιεί την τράπουλα από κάρτες που έχει το παιχνίδι.

Έχει τα εξής attributes: private Card[] cards που είναι η τράπουλα με τις κάρτες που ακόμα μπορεί να τραβήξει ο παίκτης, private Card[] usedcards που είναι η τράπουλα από χρησιμοποιημένες κάρτες, private static int NUMBER_OF_CARDS ο αριθμός καρτών που θα υπάρχουν στην τράπουλα, private int top η κορυφή της τράπουλας.

Constructor: public Deck() κατασκευάζει την τράπουλα ανακατεύοντας τα χαρτιά της

Έχει τις παρακάτω μεθόδους:

public void addCardToDeck(int index, Card card) // προσθέτει μία κάρτα που δίνεται στην τράπουλα στο σημείο index που δίνεται.

public void shuffleDeck(Card[] cards) // Ανακατεύει την τράπουλα που δίνεται.
public Card drawCard() // Τραβάει κάρτα από την τράπουλα.
public void redoDeck() // Βάζει τα χρησιμοποιημένα χαρτιά πίσω στην τράπουλα αφού τα ανακατέψει.

public boolean isEmpty() // Ελέγχει αν οι κάρτες στην τράπουλα έχουν τελειώσει.
public Card[] getCards() // Getter για την τράπουλα.
public Card[] getUsedCards() // Getter για την χρησιμοποιημένη τράπουλα.
public void setCards(Card[] cards) // Setter για την τράπουλα με την τράπουλα που δίνεται.
public void setUsedCards(Card[] usedcards) // Setter για την τράπουλα με τα χρησιμοποιημένα χαρτιά με την τράπουλα που δίνεται.

public Card removeCard(Card[] cards) // αφαιρεί και επιστρέφει την κάρτα στην κορυφή της τράπουλας που δίνεται.

Η κλάση Pawn:

- public class Pawn

  Η κλάση αυτή υλοποιεί το πιόνι του παιχνιδιού και έχει όλα τα attributes και ολες τις μεθόδους για την λογική ενός πιονιού.

  Έχει τα εξής attributes: private int position η θέση του πιονιού, είναι το value που έχει το τετραγωνάκι στο οποίο βρίσκεται.
  private int positionX η θέση του πιονού στο ταμπλό στον οριζόντιο άξονα
  private int positionY η θέση του πιονού στο ταμπλό στον κάθετο άξονα.
  private int progress η πρόοδος που έχει κάνει το πιόνι από το τετραγωνάκι που άρχισε.
  private int[] home οι συντεταγμένες απο το τετραγωνάκι που είναι το σπίτι του πιονιού στο ταμπλό
  private int[][] safezone οι συντεταγμένες απο τα τετραγωνάκια που το safezone του πιονιού στο ταμπλό.
  private boolean isAtHome μία σημαία για το εάν το πιόνι βρίσκεται σπίτι του ή όχι.
  private boolean isInSafeZone μία σημαία για το εάν το πιόνι βρίσκεται σε κάποιο τετραγωνάκι που είναι SafeZone του πιονιού.
  private Color color το χρώμα που έχει το πιόνι.

  Constructor: public Pawn(Color color) κατασκευάζει στο πιόνι με το

χρώμα που δίνεται.

Έχει τις παρακάτω μεθόδους:
public void move(int steps) // μετακινεί το πιόνι όσα τετραγωνάκια όσα τα βήματα που δίνονται.
public void sendHome() // στέλνει το πιόνι πίσω στο σπίτι του.
public void isAtHome() // Ελέγχει αν το πιόνι είναι στο σπίτι του.
public void isInSafeZone() // Ελέγχει αν το πιόνι είναι σε κάποιο τετραγωνάκι που είναι SafeZone του πιονιού.
public int getPositionX() // Getter για την θέση του πιονιού στο ταμπλό στον οριζόντιο άξονα.
public int getPositionY() // Getter για την θέση του πιονιού στο ταμπλό στον κάθετο άξονα.
public int[] getPositionXY() // Getter για την θέση του πιονιού στο ταμπλό στον οριζόντιο και κάθετο άξονα.

public Color getColor() // Getter για το χρώμα του πιονιού.
public int getProgress() // Getter για την πρόοδο του πιονιού.
public void setPositionXY(int positionX, int positionY) // Setter για την θέση του πονιού στον οριζόντιο και κάθετο άξονα με βάση την θέση που δίνεται.

public void setPositionX(int positionX) // Setter για την θέση του πιονιού οριζόντιο άξονα.
public void setPositionY(int positionY) // Setter για την θέση του πιονιού στον κάθετο άξονα.

public void setAtHome(boolean home) // Setter για την σημαία που ελέγχει αν το πιόνι είναι στο σπίτι του ή όχι.

public void setInSafeZone(boolean safe) // Setter για την σημαία που ελέγχει αν το πιόνι είναι στην ασφαλής ζώνη ή όχι.

Η κλάση player:

- public class player
  είναι η κλάση που περιέχει όλες τις ιδιότητες και μεθόδους για έναν παίκτη στο παιχνίδι.
  Έχει τα εξής attributes:

  private String name το όνομα του παίκτη, private Color playerColor το χρώμα του παίκτη, private Pawn pawn1 το πρώτο πιόνι που ανήκει στον παίκτη, private Pawn pawn2 το δεύτερο πιόνι που ανήκει στον παίκτη, private boolean isPlayerTurn μία σημαία που δηλώνει αν είναι η σειρά του παίκτη ή όχι, private boolean isCurrentlyPlaying, μία σημαία που δηλώνει αν ο παίκτης έχει τραβήξει κάρτα ή όχι, int progress η συνολική πρόοδος του παίκτη.

  Constructor: public Player(String name, Color playerColor)

φτιάχνει τον παίκτη με το όνομα και το χρώμα που δίνεται.

Έχει τις παρακάτω μεθόδους
public String getName() // Getter για το όνομα του παίκτη.
public Color getPlayerColor() // Getter για το χρώμα του παίκτη.
public Pawn getPawn1() // Getter για το πρώτο πιόνι του παίκτη.
public Pawn getPawn2() // Getter για το δεύτερο πιόνι του παίκτη.
public boolean isPlayerTurn() // Επιστρέφει αν είναι η σειρά του παίκτη ή όχι.
public int getProgress() // Getter για την πρόοδο του παίκτη.
public boolean isCurrentlyPlaying() // Επιστρέφει αν ο παίκτης έχει τραβήξει κάρτα ή όχι.
public void setName(String name) // Setter για το όνομα του παίκτη με όνομα που δίνεται.
public void setPlayerColor(Color playerColor) // Setter για το χρώμα του παίκτη με το χρώμα που δίνεται.
public void setPawn1(Pawn pawn1) // Setter για το πρώτο πιόνι του παίκτη με το πιόνι που δίνεται.
public void setPawn2(Pawn pawn2) // Setter για το δεύτερο πιόνι του παίκτη με το πιόνι που δίνεται.
public void setPlayerTurn(boolean isTurn) // Setter για την σημαία που δείχνει αν είναι η σειρά του παίκτη ή όχι.
public void setCurrentlyPlaying(boolean isPlaying) // Setter για την σημαία που δείχνει αν ο παίκτης έχει τραβήξει κάρτα ή όχι.
public void setProgress(int progress) // Setter για την πρόοδο του παίκτη με την πρόοδο που δίνεται.

public boolean playerWon() // Επιστρέφει αν ο παίκτης έχει νικήσει όχι.

Η Color

- public enum color, περιέχει όλα τα χρώματα που μπορεί να χρεαιστούν στο παιχνίδι. YELLOW, RED, BLUE, GREEN, WHITE.

- Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Το πακέτο αυτό είναι υπεύθυνο για την ομαλή λειτουργικότητα του παιχνιδιού. Λαμβάνει τις κινήσεις που κάνει ο χρήστης στο γραφικό κομμάτι του project και στην συνέχεια αλλάζει τα δεδομένα αυτού ωστέ να υπάρχει η λογική που χρειάζεται. Δηλαδή είναι η γέφυρα μεταξύ του Model και του View.

Η κλάση Controller
Τα attributes της
private Menu menuView;

private SorryGraphics gameView;

```
private Board board;

private Player[] players;

private Deck cards;

private int currentPlayer;
```

Οι μεθόδοι που έχει:

public void newCard() // Εμφανίζει μία κάρτα απο το Deck στην οθόνη πάνω από την άλλη.
public void movePawn() // Μετακινέι το πιόνι ανάλογα την κάρτα.
public void movePawns() // Μετακινεί τα πιόνια ανάλογα την κάρτα.

public void endTurn() // Τελειώνει την σειρά του παίκτη.
public void checkForWinner() // Ελέγχει αν υπάρχει νικητής στο παιχνίδι.

public void updateInfoBox() // Ανανεώνει το info box.
public void foldPressed() // Η λογική σε περίπτωση που κάποιος παίκτης πατήσει το κουμπί fold.
public void pawnCollision() // Η λογική σε περίπτωση που δύο πιόνια καταλήξουν στο ίδιο τετράγωνο.
public void saveGame() // Η λογική όταν πατηθεί το save game button.
public void continueGame() // Η λογική όταν πατηθεί το continue game button.
public void exitGame() // Η λογική όταν πατηθεί το exit game button.
public void startNewGame() // Η λογική όταν πατηθεί το start new game button.

public void updatePlayer(Player player) // Αννανεώνει τα δεδομένα του παίκτη στο τέλος της σειράς του.

public void showPossibleMoves(Card received) // Δείχνει τις κινήσεις που μπορεί να κάνει ο παίκτης με βάση την κάρτα η οποία τραβήχτηκε.

public void shuffleCards(Deck deck) // Ανακατεύει τις κάρτες ωστέ να ταιριάζουν οι εικόνες με το deck.

public void slidePawn(Pawn pawn, int i, int j) // Μετακινέι το πιόνι όλες τις θέσεις ξεκινώντας από την θέση του εώς την θέση που δίνεται.

public void startTurn(int currentPlayer) // Αρχίζει την σειρά του παίκτη που δίνεται.

- Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Στο πακέτο View υπάρχουν όλες οι κλάσεις όπου χρησιμοποιούνται για την δημιουργία

του γραφικού περιβάλλοντος του επιτραπέζιου παιχνιδιού. Υπάρχουν 2 κλάσεις. Η SorryGraphics που σε αυτή υπάρχει ένα JFrame που είναι το παράθυρο του παιχνιδιού. Στα δεξιά του JFrame θα τοποθετηθεί ότι χρειάζεται ώστε να μπορεί ο χρήστης να πατήσει και να δει μία κάρτα (Jbuttons) , οι πληροφορίες της κάρτας μαζί με τις πληροφορίες του γύρου (JtextArea) καθώς και ένα fold button. Ακόμα στα αριστερά θα υπάρχει ένα JLayeredPane που θα είναι το ταμπλό με τα τετραγωνάκια (Jbutton[][] ή και ImageIcon[][]) καθώς και τα πιόνια του κάθε παίκτη. Τέλος υπάρχει και η κλάση Menu που εκεί θα υπάρχουν όλα τα κουμπιά που έχουν να κάνουν με το μενού.

Η κλάση SorryGraphics:

Τα attributes:

```
private JFrame frame;

private URL imgURL;

private ClassLoader classLoader;

private JLabel[][] squares;

private JLabel[][] iconSquares;

private JButton[] pawns;

private ImageIcon[] pawnsIcons;

private Image image;

private JLayeredPane tableBoard;

private JLabel receiveCard;

private JLabel currentCard;

private JTextArea infoBox;

private JButton fold;

private JButton rCardBt;

private JButton cCardBt;
```

Ο Constructor: public SorryGraphics() εμφανίζει τον πίνακα στην οθόνη.

Έχει τις παρακάτω μεθόδους:

private void initializeBoard(Board board) // Αρχικοποιεί ότι χρειάζεται το παιχνίδι

private void initializePawns(Player[] players, Board board, JlayeredPane tableBoard) // Αρχικοποιεί τα πιόνια στο ταμπλό.

Public void placePawn(int pawnInPawns, int I, int j) // Τοποθετεί το εικονίδιο ενός πιονιού στις συντεταγμένες που δίνονται.

Private void initializePawnIcons(JButton[] pawns) // Αρχικοποιεί τις εικόνες των πιονιών.

Public Jlabel[][] getSquares() // Getter για τα JLabels για τα τετραγωνάκια.

public JButton getPawns() // Getter για τα Buttons των πιονιών.

public JButton getPawn(int pawnInPawns) // Getter για κάποιο συγκεκριμένο Button ενός πιονιού.

public ImageIcon getPawnIcon(int pawnIconInIcons) // Getter για κάποιο συγκεκριμένο Icon ενός πιονιού.

public ImageIcon[] getPawnIcons() // Getter για τα Icons των πιονιών.

public void setPawnIcon(int pawnInPawns, ImageIcon icon) // Setter για το Icon ενός πιονιού στο Index του πίνακα που δίνεται με το Icon που δίνεται.

public void updateBoard() // Μέθοδος που ανανεώνει τα γραφικά του παιχνιδιού.

Τέλος έχουμε προσθέσει κάποια ActionListeners που θα είναι χρήσιμα ώστε να μπορούν να αλληλεπιδρούν οι κλάσεις μεταξύ τους.

Η κλάση Menu.

Τα attributes:

private JLayeredPane menuFrame;

private JButton newGamebt;

private JButton saveGamebt;

private JButton continueGamebt;

private JButton exitGamebt;

Έχει τις παρακάτω μεθόδους:

public JButton getNewGamebt() // Getter για το new game button

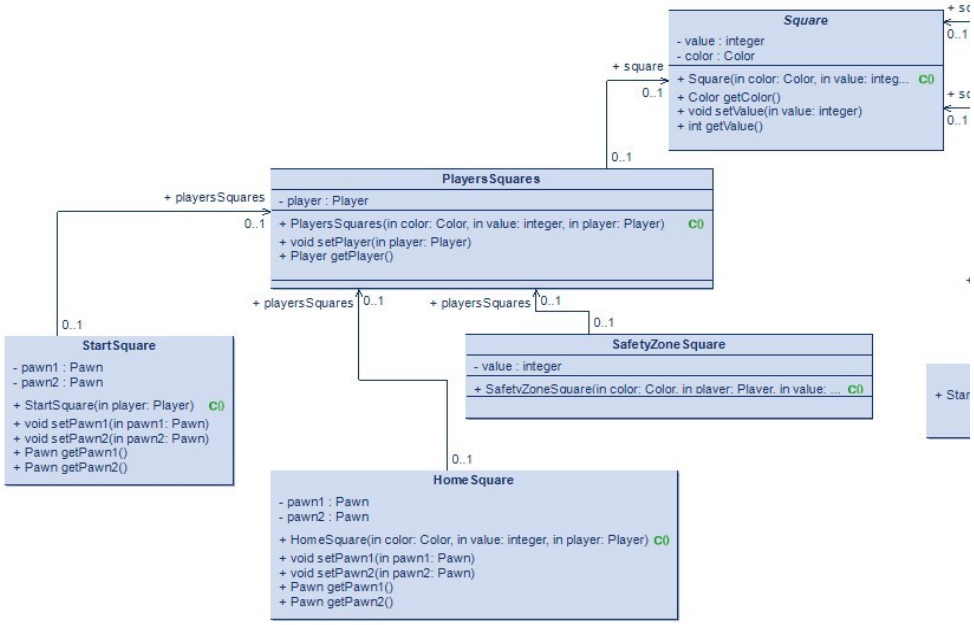
public JButton getSaveGamebt() // Getter για το save game button

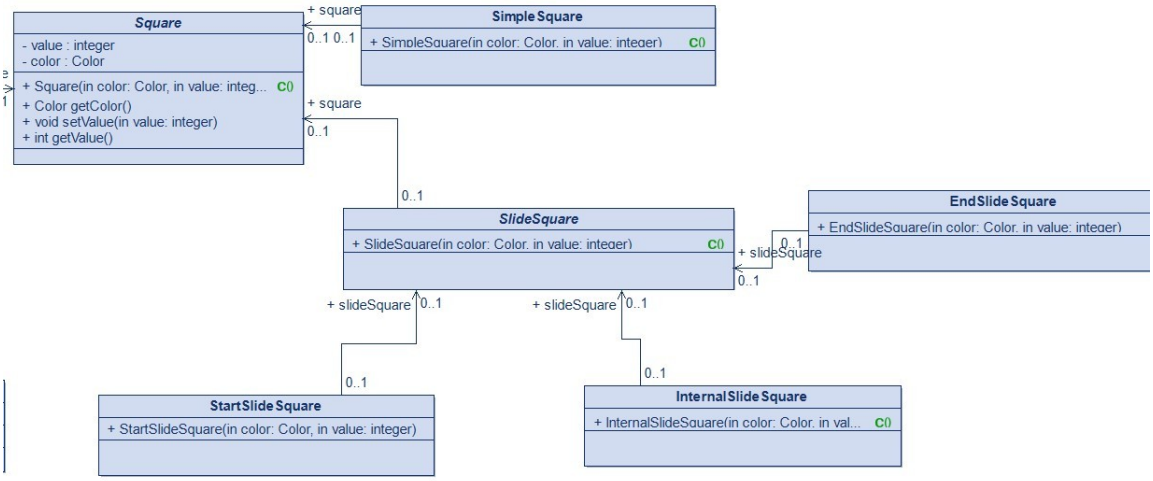
public JButton getContinueGamebt() // Getter για το continue game button.
public JButton getExitGamebt() // Getter για το extit game button.

Τέλος έχουμε προσθέσει κάποια ActionListeners που θα είναι χρήσιμα ώστε να μπορούν να αλληλεπιδρούν οι κλάσεις μεταξύ τους.

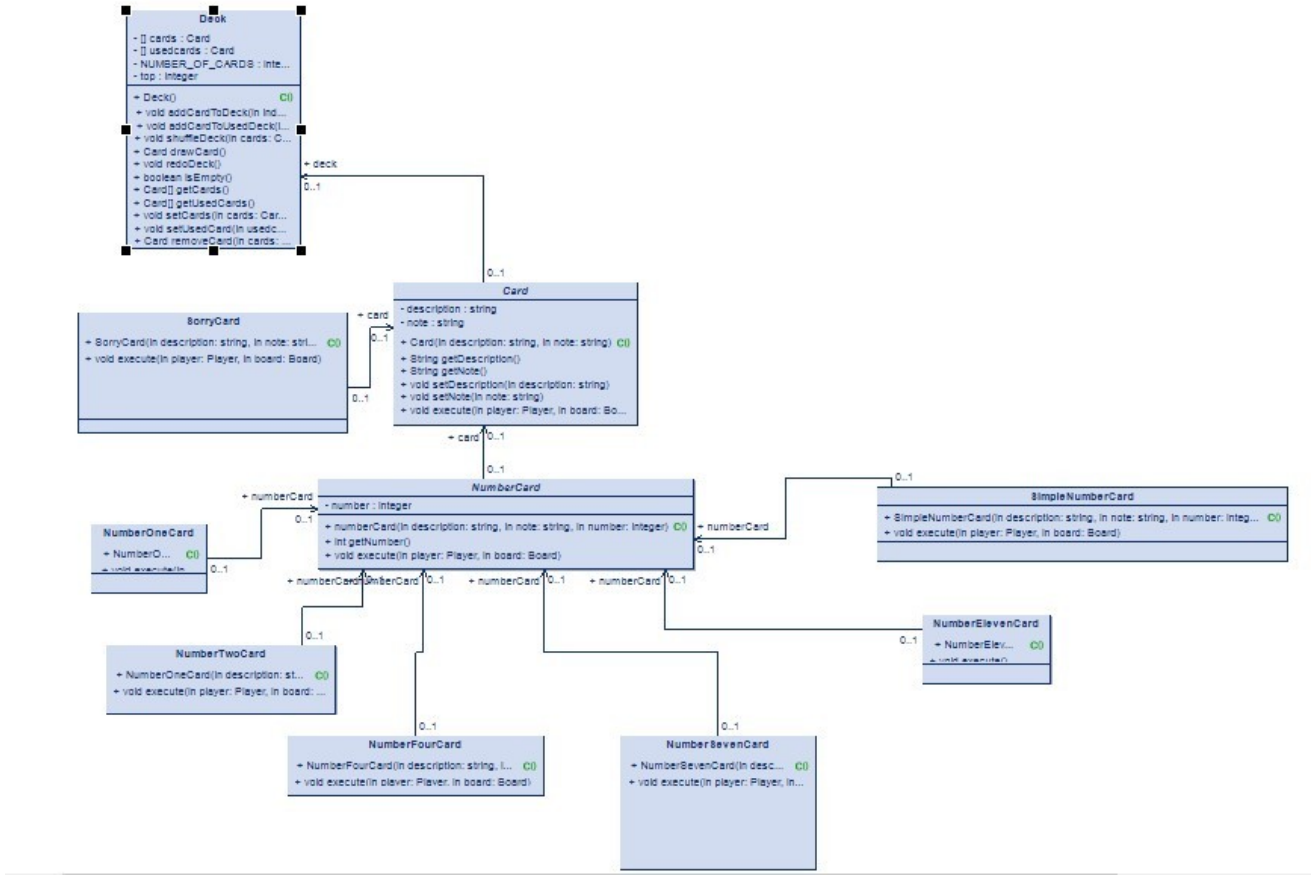
- Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

Τα UML της κλάσης Square:

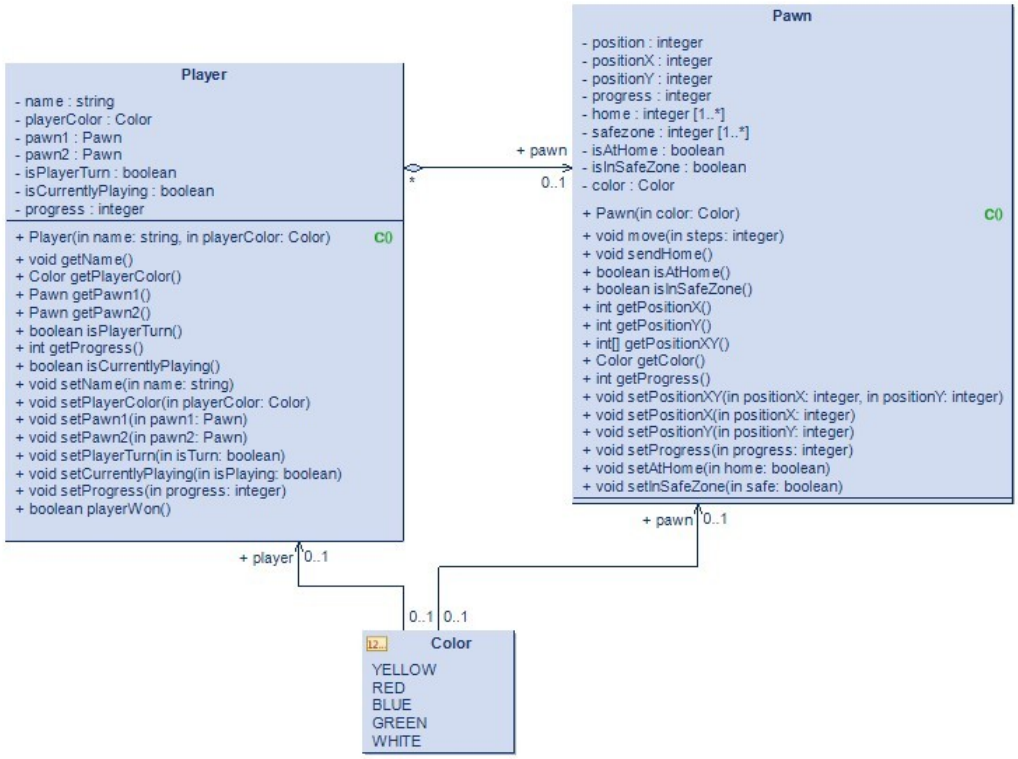




Τα UML της κλάσης Card:



Τα UML κλάσης Player, Pawn:



- Λειτουργικότητα (Β Φάση)

Στόχοι που επιτεύχθηκαν:

Αρχικοποίηση Παικτών-Ταμπλό-Στοίβας Καρτών.
Τήρηση Σειράς .
Χρήση καρτών με αριθμό 1 & 2.
Χρήση καρτών με αριθμό 3,5.
Χρήση καρτών με αριθμό 8, 12.
Χρήση καρτών με αριθμό 4 &10.
Χρήση κάρτας με αριθμό 7.
Χρήση Κάρτας Sorry! (+ έλεγχος για Safety Zone).
Έλεγχος για Fold.
Κανόνες για πιόνια στην ίδια θέση.
Χρήση Κάρτας με αριθμό 11 (+ έλεγχος για Safety Zone).
Σωστή χρήση θέσης Slide.
Σωστή χρήση θέσης Home- Νικητής.

Στόχοι που δεν επιτεύθηκαν:

Παιχνίδι έως 4 παίκτες (Bonus 5%).
Αποθήκευση Παιχνιδιού (Bonus 5%).

Μέθοδοι και Μεταβλητές που προστέθηκαν στην φάση Β:

Κλάση Controller:

Attributes:

private Card currentCard; // Keeps tract of the card that was drawn.
boolean canClickReceive; // Flag for if the player can click the ReceiveCardBt.
boolean receivesCardAgain; // Flag for if the player can click the ReceiveCardBt again.
boolean canClickFold; // Flag for if the player currently playing can click the Fold button.
boolean canChosePawn; // Flag for if the player currently playing can choose a pawn.
boolean canChoseYellow; // Flag for if the player currently playing can choose a yellow pawn.
boolean canChoseRed; // Flag for if the player currently playing can choose a red pawn.
boolean canChosePawn1; // Flag for if the player currently playing can choose pawn1.
boolean canChosePawn2; // Flag for if the player currently playing can choose pawn2.

Methods:

public void handleReceiveCardBt() :

Η μεθοδος αυτή διαχειρίζεται την ενέργεια του κουμπιου για την λήψη μιας κάρτας. Τραβάει μία νέα κάρτα και ενημερώνει τα γραφικά ανάλογα.

public void handleCurrentCardBt() :

Η μέθοδος αυτή ελέγχει και εκτελεί την ενέργεια της τραβηγμένης κάρτας, Ανάλογα την κάρτα που ο παίκτης τράβηξε, η μέθοδος δίνει την επιλογή στον παίκτη να επιλέξει ένα πιόνι για να μετακινήσει, ή να επιλέξει μία κίνηση μέσω ενός παραθύρου επιλογών.

public void pawnSelection():

Η μέθοδος αυτή δίνει την δυνατότητα στον παίκτη που παίζει να επιλέξει ένα απο τα πιόνια του.

public void handlePawnChosen(model.Color color, int id):

Η μέθοδος αυτή εκτελεί την ενέργεια της κάρτας στο πιόνι που ο παίκτης επέλεξε. Είναι η συνάρτηση στο Action Listener των πιονιών.

public void handleCardSorry(Pawn pawn, Player currentPlayer, Player otherPlayer, int id):

Η μέθοδος αυτή εκτελεί την ενέργεια της κάρτας Sorry. Καλείται εφόσον ο παίκτης που παίζει γίνεται και έχει επιλέξει να ανταλλάξει ένα πιόνι του από την θέση start στην θέση που βρίσκεται το πιόνι του αντιπάλου, εφόσον το πιόνι του αντιπάλου δεν βρίσκεται στην θέση start.

public void handleCard7():

Η μέθοδος αυτή εκτελεί την ενέργεια της κάρτας 7. Ελέγχει τους τρόπου όπου μπορεί ο παίκτης που παίζει να μετακινήσει τα πιόνια του 7 θέσεις.
Αν υπάρχει μόνο ένα πιόνι που μπορεί να μετακινήσει τότε ο παίκτης πρέπει να

επιλέξει το πιόνι αυτό για να μετακινηθεί 7 θέσεις. Αλλιώς εμφανίζεται ένα παράθυρο επιλογών με τις επιλογές που μπορεί να κάνει ο παίκτης.
Αν η επιλογή δεν είναι διαθέσιμη στην θέση της εμφανίζεται null και ο παίκτης δεν γίνεται να το πατήσει. Εάν το πατήσει, τότε το παράθυρο εμφανίζεται ξανά.
Ανάλογα με τις κινήσεις που επιλέγει να μοιράσει ο παίκτης, τα πιόνια του μετακινούνται και η σειρά του τελειώνει.

public void handleCard10():

Παρομοίως όπως με την παραπάνω μέθοδο μόνο που δίνει επιλογές ανάλογα με το αν ο παίκτης θέλει και μπορεί να μετακινήσει τα πιόνια του -1 θέση ή +10 θέσεις.

public void slidePawn(Square square):

Η μέθοδος αυτή εκτελεί την διαδικασία μετακίνησης ενός πιονιού σε μία τσουλήθρα. Παίρνει ως παράμετρο ένα τετραγωνάκι που ανήκει στο ταμπλό και ελέγχει αν αυτό είναι κλάσης StartSlideSquare. Αν είναι τότε καλέι την συνάρτηση Slide της StartSlideSquare και για κάθε pawn που μετακινήθηκε, η συνάρτηση το μετακινεί και στα γραφικά.

Κλάση SorryGraphics:

Attributes:

private controller controller; // The controller of the game.
private JTextArea errorBox; // The box for displaying Errors and Guidence.
private ImageIcon boardIcon; // The background image of the game.
private JLabel background; // The background label of the game.
JButton redPawn1Button;
JButton redPawn2Button;
JButton yellowPawn1Button;
JButton yellowPawn2Button;
private ImageIcon rCardIcon; // Icon for the receive card button.
int squareSize; // The size of a single square.

Methods:

public void updateCurrentCardDisplay(Card card) :

Ανάλογα με την κάρτα που δίνεται, ενημερώνει την εικόνα του Current Card κουμπιού ή το αρχικοποιεί αν δεν υπάρχει ήδη.

public void updateTextBox(String numberOfCardsLeft, String playersTurn, String description) :

Ενημερώνει το Info Box, με το πόσες κάρτες μένουν, τον παίκτη που έχει σειρά,

και την περιγραφή της κάρτας.

public void showErrorMessage(String message) :
Αρχικοποιεί ή ενημερώνει ένα παράθυρο που μέσα του υπάρχει κείμενο για κινήσεις που δεν γίνεται να κάνει ο παίκτης στην σειρά του (όπως αν προσπαθήσει να τραβήξει κάρτα δύο φορές κτλ.) ή και τον βοηθάει με το τι να κάνει ανάλογα με την κάρτα που επέλεξε.

public int showOptions(String message, String title, String[] options) :
Εμφανίζει ένα JOptionPane που δείχνει επιλογές στον παίκτη ανάλογα με την κάρτα που τράβηξε.

Ακόμα προσθέθηκαν κάποια παραπάνω Action Listeners για κάθε πιόνι του κάθε παίκτη.

Κλάση NumberSevenCard:

Methods:

public boolean canMove(int moves1, int moves2, Player player, Board board):
Ελέγχει αν τα πιόνια του παίκτη μπορούν να μετακινηθούν και τα δύο τις θέσεις που δίνονται.

public int[] executeCard7(int moves1, int moves2, Player player, Board board):
Μετακινεί και τα δύο πιόνια του παίκτη που δίνεται τις θέσεις που δίνονται. Επιστρέφει ένα Array με την νέα τοποθεσία του πιονιού ωστέ να ενημερωθούν τα γραφικά.

Κλάση NumberTenCard:

Methods:

public boolean canMove(int spaces, Pawn pawn, Board board):
Ελέγχει αν το πιόνι που δίνεται μπορεί να προχωρήσει μπροστά όσες θέσεις δίνονται.

public boolean canMoveBackwards(int spaces, Pawn pawn, Board board):
Ελέγχει αν το πιόνι που δίνεται μπορεί να προχωρήσει πίσω όσες θέσεις δίνονται. Επιστρέφει την νέα θέση για να μπορέσουν να ενημερωθούν τα γραφικά.

public int[] executeCard10(int moves, Pawn pawn, Board board) :

Μετακινεί το πιόνι όσες κινήσεις δίνονται. Αν ο αριθμός κινήσεων είναι αρνητικός, το πιόνι πάει τόσες κινήσεις πίσω. Επιστρέφει την νέα θέση για να μπορέσουν να ενημερωθούν τα γραφικά.

Κλάση SorryCard:

Methods:

public int[] executeSorryCard(Pawn pawn1, Pawn pawn2, Board board) :

Πηγαίνει το pawn 1 στην θέση του Pawn2 και το Pawn2 στην αρχή του.

Κλάση SimpleNumberCard:

Methods:

public boolean canMove(int spaces, Pawn pawn, Board board):

Ελέγχει αν το πιόνι που δίνεται μπορεί να μετακινηθεί κατά τον αριθμό spaces.

Κλάση NumberElevenCard:

Methods:

public int countProgress(Pawn pawn):

Μετράει το progress του πιονιού που δίνεται από το αρχικό κουτάκι που αυτό αρχίζει εώς την θέση όπου βρίσκεται.

public boolean canMove(int spaces, Pawn pawn, Board board):

Ελέγχει αν το πιόνι που δίνεται μπορεί να μετακινηθεί κατά τον αριθμό spaces.

public int[] executeSwapCard(Pawn pawn1, Pawn pawn2, Board board):

Ανταλλάσει τα πιόνια μεταξύ τους πάνω στο board.

public int[] executeCard11(int moves, Pawn pawn, Board board):

Μετακινέι το πιόνι κατά τις θέσεις που δίνονται.

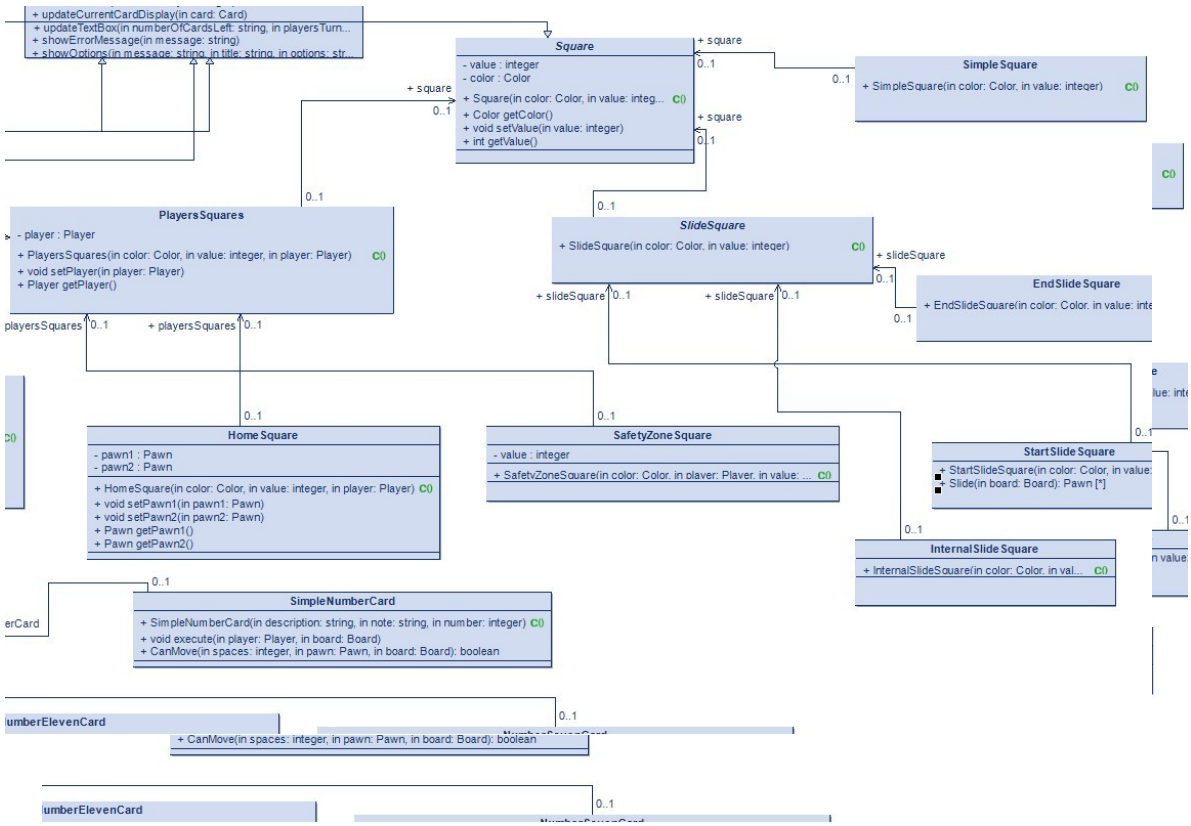
Κλάση StartSlideSquare:

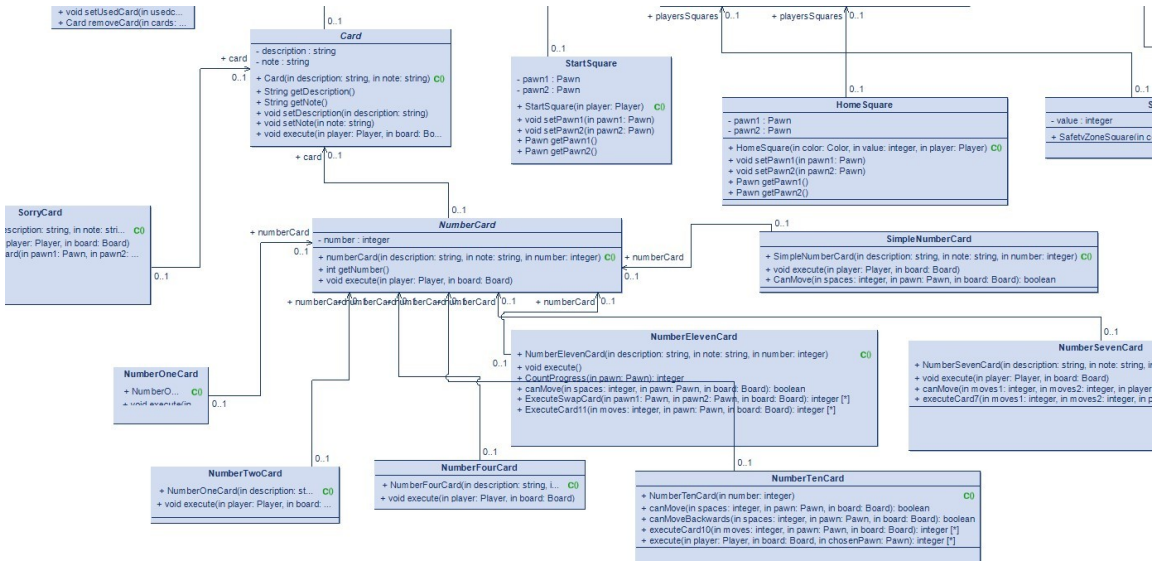
public Pawn[] Slide(Board board):

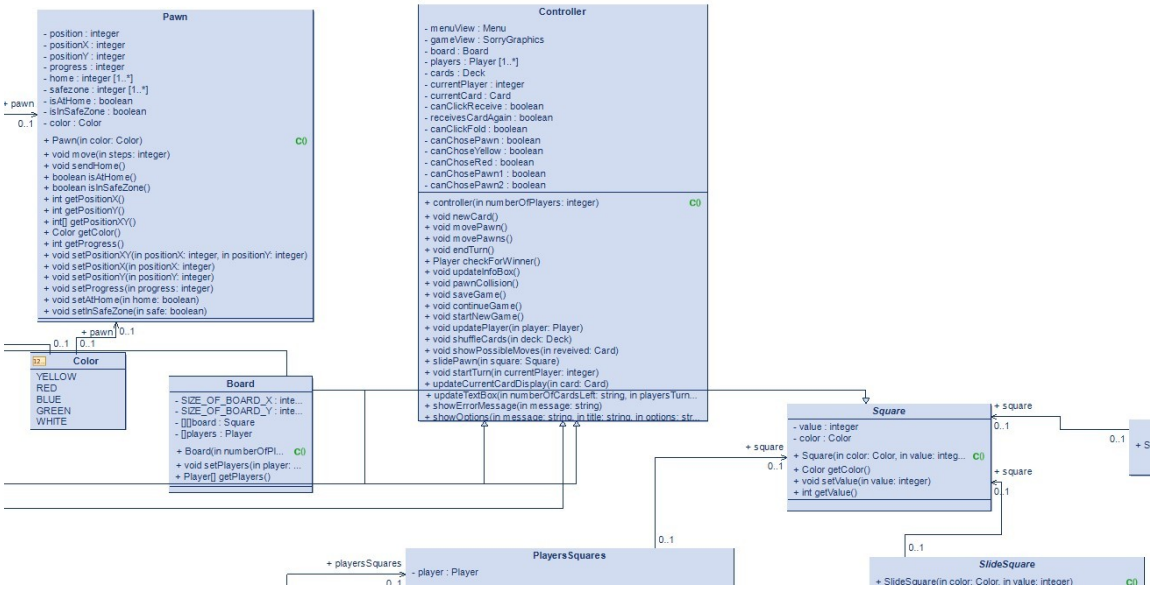
Μετακινεί το πιόνι που βρίσκεται πάνω στο αντικείμενο StartSlideSquare, μέχρι να συναντήσει EndSlideSquare. Αν βρει άλλα πιόνια ωστόσο, τα προσθέτει σε ένα PawnArray και τα μετακινεί πίσω στην αρχική τους θέση.

UML :

Οι κλάσεις παρέμειναν ίδιες όπως και την Α φάση οπότε τα UML δεν άλλαξαν πολύ. Προσθέθηκαν όμως οι καινούργιες μεθόδοι και τα Attributes καθώς και η κλάση CardTen που δεν είχε προσθεθεί στην φάση Α λόγο απροσεξίας.


+ updateCurrentCardDisplay(in card: Card)
+ updateTextBox(in numberOfCardsLeft: string, in playersTurn...
+ showErrorMessage(in message: string)
+ showOptions(in message: string, in title: string, in options: str...
Square
- value : integer
- color : Color
+ Square(in color: Color, in value: integ... C0
+ Color getColor()
+ void setValue(in value: integer)
+ int getValue()
+ square
0..1
Simple Square
+ SimpleSquare(in color: Color, in value: integer) C0
PlayersSquares
- player : Player
+ PlayersSquares(in color: Color, in value: integer, in player: Player) C0
+ void setPlayer(in player: Player)
+ Player getPlayer()
playersSquares
SlideSquare
+ SlideSquare(in color: Color, in value: integer) C0
+ slideSquare
EndSlideSquare
+ EndSlideSquare(in color: Color, in value: inte
Home Square
- pawn1 : Pawn
- pawn2 : Pawn
+ HomeSquare(in color: Color, in value: integer, in player: Player) C0
+ void setPawn1(in pawn1: Pawn)
+ void setPawn2(in pawn2: Pawn)
+ Pawn getPawn1()
+ Pawn getPawn2()
SafetyZone Square
- value : integer
+ SafetyZoneSquare(in color: Color, in player: Player, in value: ... C0
StartSlide Square
+ StartSlideSquare(in color: Color, in value:
+ Slide(in board: Board): Pawn [*]
InternalSlide Square
+ InternalSlideSquare(in color: Color, in val... C0
SimpleNumberCard
+ SimpleNumberCard(in description: string, in note: string, in number: integer) C0
+ void execute(in player: Player, in board: Board)
+ CanMove(in spaces: integer, in pawn: Pawn, in board: Board): boolean
+ CanMove(in spaces: integer, in pawn: Pawn, in board: Board): boolean
0..1


+ void setUsedCard(in usedc...
+ Card removeCard(in cards: ...
Card
- description : string
- note : string
+ Card(in description: string, in note: string) C0
+ String getDescription()
+ String getNote()
+ void setDescription(in description: string)
+ void setNote(in note: string)
+ void execute(in player: Player, in board: Bo...
StartSquare
- pawn1 : Pawn
- pawn2 : Pawn
+ StartSquare(in player: Player) C0
+ void setPawn1(in pawn1: Pawn)
+ void setPawn2(in pawn2: Pawn)
+ Pawn getPawn1()
+ Pawn getPawn2()
Home Square
- pawn1 : Pawn
- pawn2 : Pawn
+ HomeSquare(in color: Color, in value: integer, in player: Player) C0
+ void setPawn1(in pawn1: Pawn)
+ void setPawn2(in pawn2: Pawn)
+ Pawn getPawn1()
+ Pawn getPawn2()
SorryCard
NumberCard
- number : integer
+ numberCard(in description: string, in note: string, in number: integer) C0
+ int getNumber()
+ void execute(in player: Player, in board: Board)
SimpleNumberCard
+ SimpleNumberCard(in description: string, in note: string, in number: integer) C0
+ void execute(in player: Player, in board: Board)
+ CanMove(in spaces: integer, in pawn: Pawn, in board: Board): boolean
NumberElevenCard
+ NumberElevenCard(in description: string, in note: string, in number: integer) C0
+ void execute()
+ CountProgress(in pawn: Pawn): integer
+ canMove(in spaces: integer, in pawn: Pawn, in board: Board): boolean
+ ExecuteSwapCard(in pawn1: Pawn, in pawn2: Pawn, in board: Board): integer [*]
+ ExecuteCard11(in moves: integer, in pawn: Pawn, in board: Board): integer [*]
NumberSevenCard
+ NumberSevenCard(in description: string, in note: string,
+ void execute(in player: Player, in board: Board)
+ canMove(in moves1: integer, in moves2: integer, in player
+ executeCard7(in moves1: integer, in moves2: integer, in p
NumberOneCard
+ NumberO... C0
NumberTwoCard
+ NumberOneCard(in description: st... C0
+ void execute(in player: Player, in board: ...
NumberFourCard
+ NumberFourCard(in description: string, i... C0
+ void execute(in player: Player, in board: Board)
NumberTenCard
+ NumberTenCard(in number: integer) C0
+ canMove(in spaces: integer, in pawn: Pawn, in board: Board): boolean
+ canMoveBackwards(in spaces: integer, in pawn: Pawn, in board: Board): boolean
+ executeCard10(in moves: integer, in pawn: Pawn, in board: Board): integer [*]
+ execute(in player: Player, in board: Board, in chosenPawn: Pawn): integer [*]
+ card
+ numberCard
+ playersSquares
0..1



- Συμπεράσματα

Το project ήταν πολύ ικανοποιητικό. Χρησιμοποιήθηκαν οι περισσότερες συναρτήσεις που αναφέρθηκαν στην Α φάση καθώς και η λογική μεταξύ Α και Β φάσης δεν άλλαξε πολύ. Το μόνο πρόβλημα που αντιμετώπισα είναι το ότι δεν είχα πολύ χρόνο να κάνω κάποια πράγματα στην αρχή και έτσι έτρεξα να κάνω όσα πράγματα μπορούσα μέχρι την πρώτη προθεσμία της Β φάσης. Αφού όμως πήρε παράταση, προσπάθησα και κατάφερα να ολοκληρώσω όλο το project άλλα δυστυχώς ο κώδικας μου δεν είναι καθόλου οργανωμένος όπως έπρεπε να είναι. Έμαθα πράγματα που δεν θα είχα μάθει αν δεν είχα ασχοληθεί με το project, τόσο στην οργάνωση του χρόνου και του κώδικα, όσο και στο προγραμματιστικό κομμάτι. Γενικά πολύ ενδιαφέρουσα και ευχάριστη εργασία.