Individual Assignment: Patience is a Virtue

Michal Wojciech Goly [mwg2]

1st May 2015

Contents

1 Introduction

1.1 Project description

Patience is a simple card game for one player, in which the objective is to end up with one pile of cards on the table. There are 52 playing cards at the start of the game, all facing downwards in a pack. Player can then deal a card which will be removed from the deck and put on the table facing upwards. By continuing to do so, there could potentially be 52 cards on the table all facing upwards, unless a move was made. Apart from dealing cards there are two additional valid moves in the game. You can join two cards together if they have the same suit or value and if:

- They are next to each other
- There are two other cards between them

When joined, the card further to the right will be placed on top of the other, regardless of the order in which they have been selected. Each move is worth 10 points in the game. Therefore because there are 51 available moves in total, the highest possible score is 510 points.

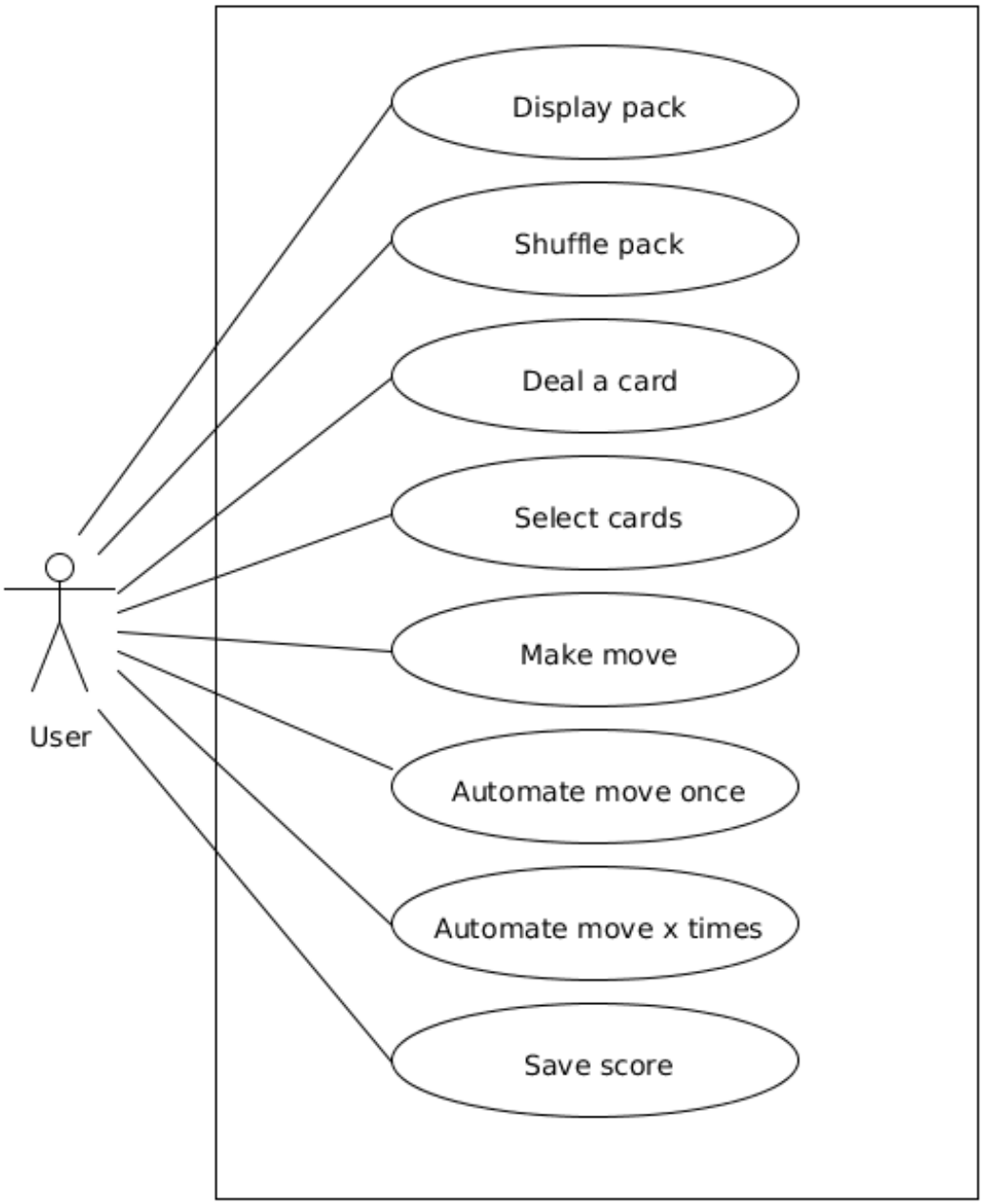
1.2 Game controls

Because the user interface is fully graphical, to play the game user can simply click on the cards. For example if the player wants to deal a card, he should click on the pack and a move will be made. Similarly, to select a card user has to click on it. In order to indicate which card was selected, a blue border will be painted around it.

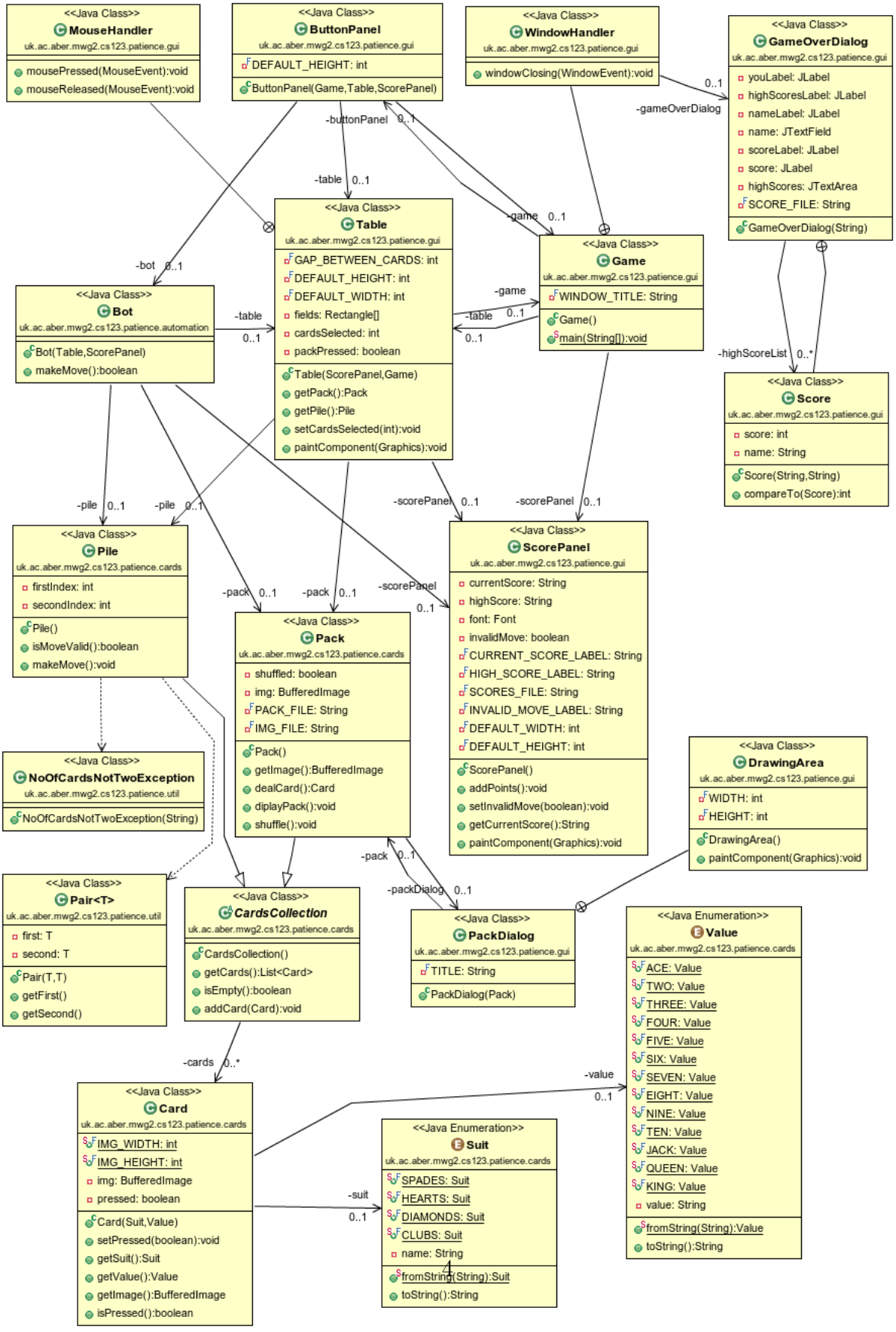
Additional options are available within the button panel at the bottom of the window. Player can display contents of the pack, as well as shuffle it. Second option is only permitted once and should typically be selected at the start of the game. Pack is not randomized by default due to the requirements specification of the assignment. Last two options allow an automation of the gameplay. Player can either make use of the 'Play for me once' option which as expected will make one valid move in the game, if there is one, or specify the amount of moves to be made by clicking the 'Play for me x times' button.

Game ends either if the automation algorithm detects that there are no more moves available, user presses the 'x' exit button on the top of the window or game is won. Before the application closes, a smaller window will pop up to ask the player to enter his name in order to save his score. User can choose not to store his result by leaving the name field blank.

2 Requirements Analysis



3 Design


<<Java Class>>
MouseHandler
uk.ac.aber.mwg2.cs123.patience.gui
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
<<Java Class>>
ButtonPanel
uk.ac.aber.mwg2.cs123.patience.gui
DEFAULT_HEIGHT: int
ButtonPanel(Game,Table,ScorePanel)
<<Java Class>>
WindowHandler
uk.ac.aber.mwg2.cs123.patience.gui
windowClosing(WindowEvent):void
<<Java Class>>
GameOverDialog
uk.ac.aber.mwg2.cs123.patience.gui
youLabel: JLabel
highScoresLabel: JLabel
nameLabel: JLabel
name: JTextField
scoreLabel: JLabel
score: JLabel
highScores: JTextArea
SCORE_FILE: String
GameOverDialog(String)
-buttonPanel 0..1
-table 0..1
-gameOverDialog 0..1
-game 0..1
-bot 0..1
<<Java Class>>
Table
uk.ac.aber.mwg2.cs123.patience.gui
GAP_BETWEEN_CARDS: int
DEFAULT_HEIGHT: int
DEFAULT_WIDTH: int
fields: Rectangle[]
cardsSelected: int
packPressed: boolean
Table(ScorePanel,Game)
getPack():Pack
getPile():Pile
setCardsSelected(int):void
paintComponent(Graphics):void
<<Java Class>>
Game
uk.ac.aber.mwg2.cs123.patience.gui
WINDOW_TITLE: String
Game()
main(String[]):void
<<Java Class>>
Bot
uk.ac.aber.mwg2.cs123.patience.automation
Bot(Table,ScorePanel)
makeMove():boolean
-table 0..1
-game 0..1
-highScoreList 0..*
<<Java Class>>
Score
uk.ac.aber.mwg2.cs123.patience.gui
score: int
name: String
Score(String,String)
compareTo(Score):int
-pile 0..1
-scorePanel 0..1
<<Java Class>>
Pile
uk.ac.aber.mwg2.cs123.patience.cards
firstIndex: int
secondIndex: int
Pile()
isMoveValid():boolean
makeMove():void
-pack 0..1
<<Java Class>>
ScorePanel
uk.ac.aber.mwg2.cs123.patience.gui
currentScore: String
highScore: String
font: Font
invalidMove: boolean
CURRENT_SCORE_LABEL: String
HIGH_SCORE_LABEL: String
SCORES_FILE: String
INVALID_MOVE_LABEL: String
DEFAULT_WIDTH: int
DEFAULT_HEIGHT: int
ScorePanel()
addPoints():void
setInvalidMove(boolean):void
getCurrentScore():String
paintComponent(Graphics):void
<<Java Class>>
Pack
uk.ac.aber.mwg2.cs123.patience.cards
shuffled: boolean
img: BufferedImage
PACK_FILE: String
IMG_FILE: String
Pack()
getImage():BufferedImage
dealCard():Card
diplayPack():void
shuffle():void
<<Java Class>>
DrawingArea
uk.ac.aber.mwg2.cs123.patience.gui
WIDTH: int
HEIGHT: int
DrawingArea()
paintComponent(Graphics):void
<<Java Class>>
NoOfCardsNotTwoException
uk.ac.aber.mwg2.cs123.patience.util
NoOfCardsNotTwoException(String)
-packDialog 0..1
<<Java Class>>
Pair<T>
uk.ac.aber.mwg2.cs123.patience.util
first: T
second: T
Pair(T,T)
getFirst()
getSecond()
<<Java Class>>
CardsCollection
uk.ac.aber.mwg2.cs123.patience.cards
CardsCollection()
getCards():List<Card>
isEmpty():boolean
addCard(Card):void
<<Java Class>>
PackDialog
uk.ac.aber.mwg2.cs123.patience.gui
TITLE: String
PackDialog(Pack)
<<Java Enumeration>>
Value
uk.ac.aber.mwg2.cs123.patience.cards
ACE: Value
TWO: Value
THREE: Value
FOUR: Value
FIVE: Value
SIX: Value
SEVEN: Value
EIGHT: Value
NINE: Value
TEN: Value
JACK: Value
QUEEN: Value
KING: Value
value: String
fromString(String):Value
toString():String
-cards 0..*
-value 0..1
<<Java Class>>
Card
uk.ac.aber.mwg2.cs123.patience.cards
IMG_WIDTH: int
IMG_HEIGHT: int
img: BufferedImage
pressed: boolean
Card(Suit,Value)
setPressed(boolean):void
getSuit():Suit
getValue():Value
getImage():BufferedImage
isPressed():boolean
-suit 0..1
<<Java Enumeration>>
Suit
uk.ac.aber.mwg2.cs123.patience.cards
SPADES: Suit
HEARTS: Suit
DIAMONDS: Suit
CLUBS: Suit
name: String
fromString(String):Suit
toString():String

4 Testing

5 Evaluation