BILKENT UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CS319 - Object-Oriented Software Engineering

Project Name: Seven Wonders

Design Report

Group No: 2G
Ömer Faruk Oflaz
Deniz Doğanay
Safa Alperen Oruç
Göktuğ Öztürkcan
Ahmet Berk Eren

Contents

1. Introduction

1.1 Purpose of the system

Board games help people to learn new skills, strategies and drag people to think. While doing these, board games provides people from all ages to have fun, socialize, share. The purpose of this project is to implement the virtualized version of Seven Wonders board game and add new features to make the game more challenging and fun. Seven Wonders is a board game in which player needs to advance the wonder in terms of several aspects and collect the most victory points to win the game. It features representations of ancient civilizations, military conflicts and economic activity. There are advantages of the virtualized version of Seven Wonders compared to the real version. In the digital version additional features can be added to the game also in the digital version game would be much quicker because of the easiness to set up. No need to setup physical game will make the game more fun. Implementing new features will make the game unique and more qualified. Getting user name and board type information from the player make the game unique and having a sound effects and music will attract the player.

1.2 Design Goals

Criteria

User Friendly:

User friendliness of seven wonders is very important for making the game more playable. Simplicity is very important criteria for digital board games. Since the game is very complicated, we decided to make the in game interfaces and menu interfaces simpler so that player can focus on the game and not get distracted with the details. In order to make the game more user friendly, we pay attention for the number of clicks in the game. Player can start the game with only two clicks, access how to play? screen with one click, can arrange the sound and music with two clicks. Only in the user name part player interacts via

keyboard, all of the other parts in the game only requires the mouse events.

Easy to Learn:

If the player does not know how to play the game, digital version of seven wonders have a how to play? option which explains the detailed version of the game. Also player can access how to play? screen in game or before game.

Supportability:

For good supportability we choose to make our game in Java so it can be accessed no matter which operating system the player has. The only prerequisite for players to play this game is they need to have Java installed in their computer.

Modifiability:

Best part about creating a digital version of a board game is that it is highly modifiable. Cards can be changed, the powers of wonders can be changed, etc. Basically every little detail in the game could be changed or new features could be easily implemented in the game. Also we will be writing the code in a way that's easy to read which is going to increase modifiability.

Trade-Offs

Development Time vs Performance:

Since this is going to be a simple game compared to what processors nowadays can handle, performance of the game won't be taken into account for the most part. So development time of the game is not going to increase much because of our worries about performance.

Functionality vs Usability:

Usability for a digitalized board game is quite important. As we mentioned above we will try to keep user friendliness level of the game as high as possible. However high level of usability does not mean it has to decrease the functionality of this game. We plan on introducing some new features into the game which will increase its functionality. However these new features won't be so drastic that the player will get frustrated while playing. New features will increase complexity slightly. But we believe this low level of increased complexity will not lower the games usability.

2. High-level Software Architecture

2.1 Subsystem Decomposition

While implementing the game, we will follow MVC software design pattern, which will allow us to divide the program into three elements. All three elements will execute their specific part of task as well as communicating with other elements.

Model is the main component of the application, which maintains game components and mechanics by receiving input from Controller and providing data for Controller.

View is a presentation of Model to the user by the interface. This is what the user only sees. It gets input from Controller and provides data for Controller.

Controller is the middle bridge between Model and View. It takes input from View via user and transfers it to Model; and takes input from Model and transfers it to View, which presents it to the user.

2.2 Hardware/Software Mapping

Seven Wonders is implemented using JavaFX. Player needs to download Java JRE that is suitable for the device.

Mouse is necessary to play this game. Speaker is optional; however, use of speaker is recommended regarding enjoying game with background sounds.

Player needs to run .jar file to open and play the game.

2.3 Persistent Data Management

Images and sounds will be in the game. These features will be stored in local storage.

2.4 Access Control and Security

Seven Wonders is a desktop game which is event-driven. Events in the game are in flow dependant to player's decisions. Decisions are made in menus or during playing a game. This decisions are assessed by instances of MainMenu and PlayGame.

Before starting the game, player enters a username to be used in the game.

2.5 Boundary Conditions

There are 3 boundary conditions such as starting the game, terminating the game and errors.

2.5.1 Starting the game

Player starts the game by running .jar file.

2.5.2 Terminating the game

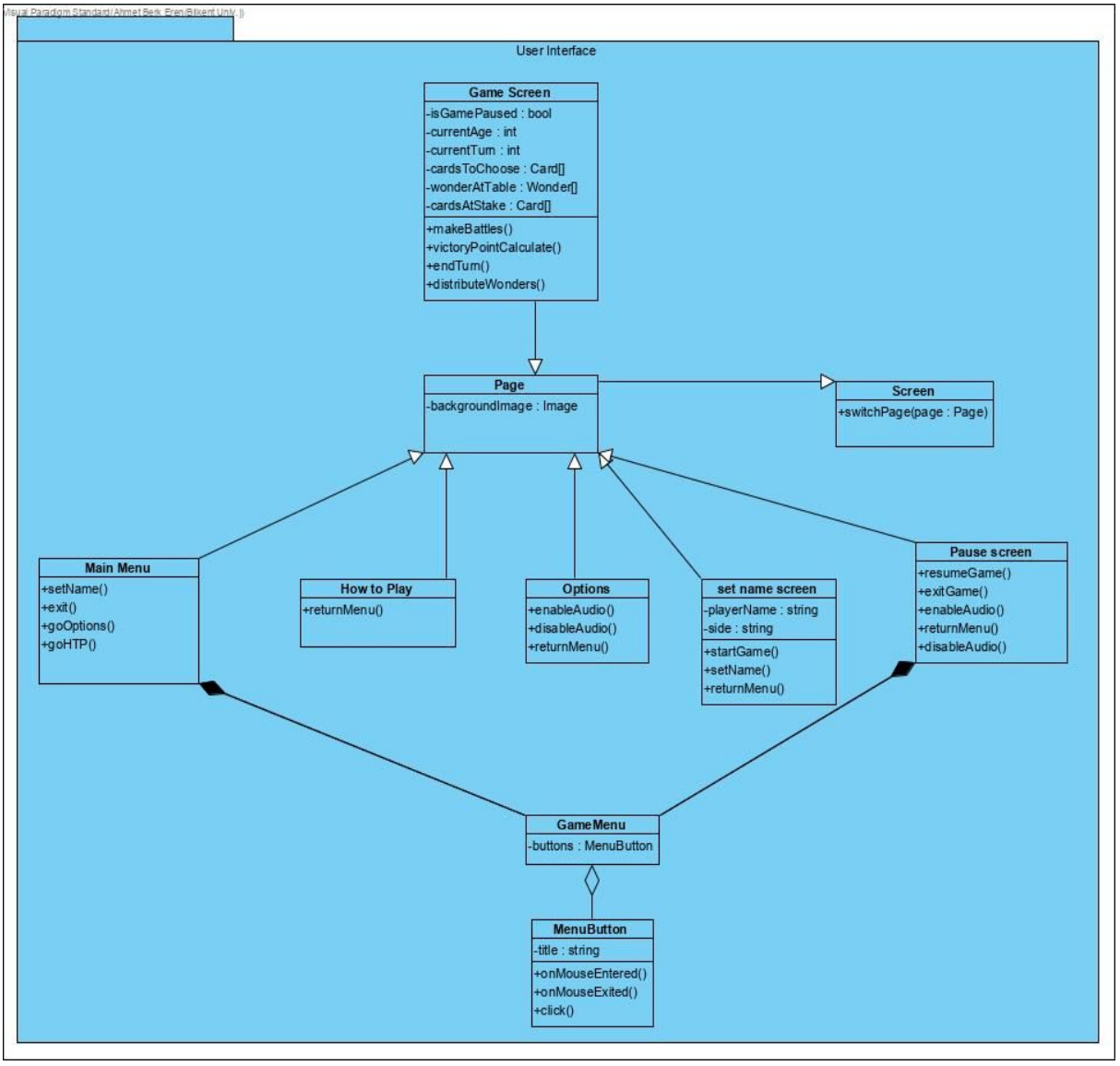
Player terminates the game by clicking the cross button on the right corner of the window.

2.5.3 Errors

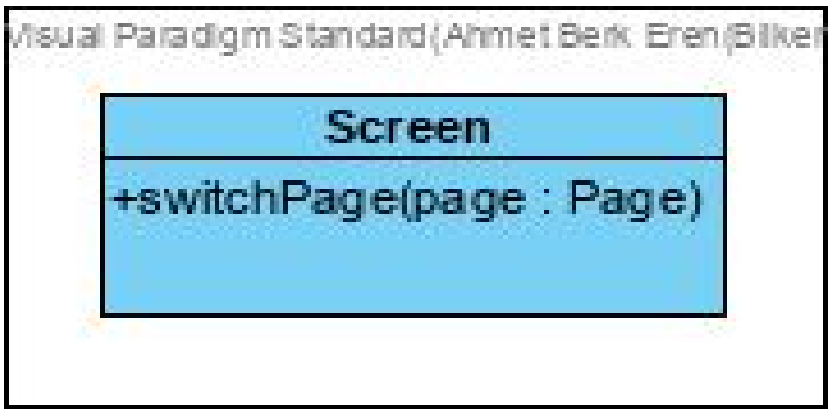
There may be errors and when it occurs, the game terminates itself

3. Subsystem Services

a. User Interface



i. Screen



This class is the main user interface of the application. It provides the user to change the screen.

Methods:

- public static void switchPage(Page page) : this method sets the application's screen as given page.

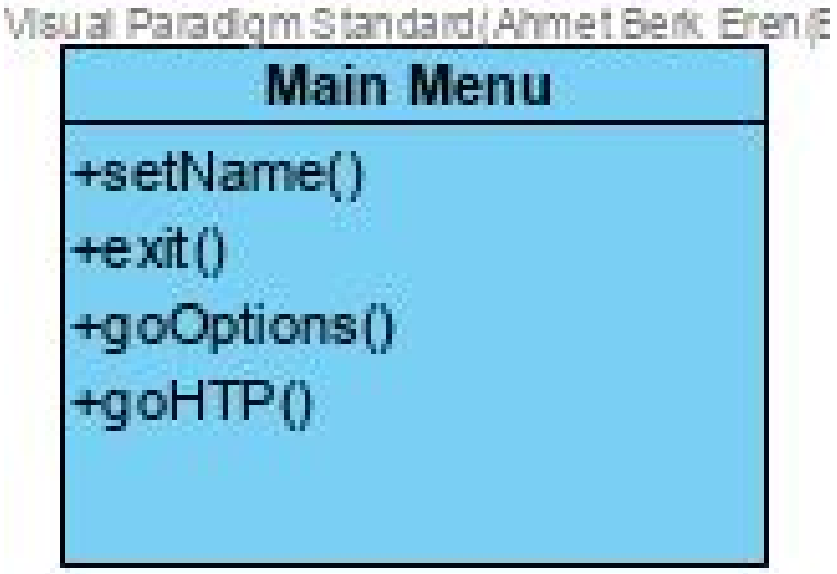
ii. Page

This is an abstract class which holds different scenes according to its unique design.

Attributes:

- private Image backgroundImage: background image of the screen.

iii. MainMenu



This is the main menu page which is the starting screen of the game.

Methods:

- private void setName() : this method changes the page from the main menu page to set name page.
- private void exit(): this method closes the game.
- private void goOptions(): this method changes the page from the main menu page to options page.
- private void goHTP(): this method changes the page from the main menu page to how to play page.

iv. How to Play

This shows how to play page.

Methods:

- private void returnMenu(): this method provides to return to main menu page.

v. Options

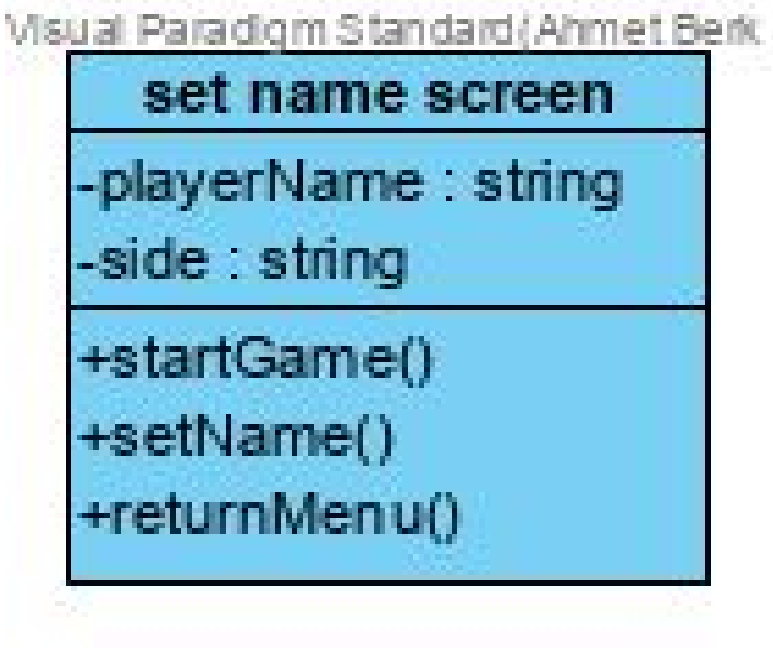
This shows the options page and the user can configure the options.

Methods:

- private void enableAudio(): this method opens the music of the game.
- private void disableAudio(): this method closes the music of the game.

- private void returnMenu(): this method provides to return main menu.

vi. Set name page



This provides the user to set his or her name and side of the wonder and start the game.

Attributes:

- private String playerName: this string is the user's name.
- private String side: this shows which side of the wonder will be played.

Methods:

- private void startGame(): this method change the page to game page.
- private void setName(): this method provides the user to set his or her name.
- private void returnMenu(): this method provides to return main menu.

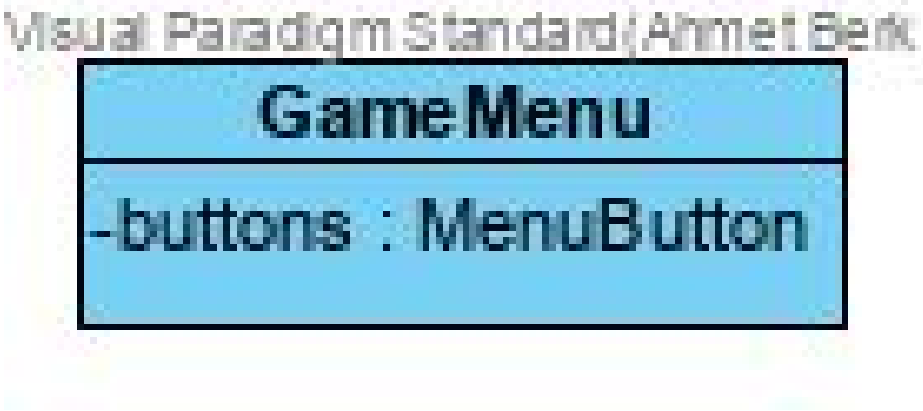
vii. Pause screen

This screen occurs when the user user pauses the game.

Methods:

- private void resumeGame(): this method returns the game page.
- private void exitGame(): this method closes the game.
- private void enableAudio(): this method opens the music of the game.
- private void disableAudio(): this method closes the music of the game.
- private void returnMenu(): this method provides to return main menu.

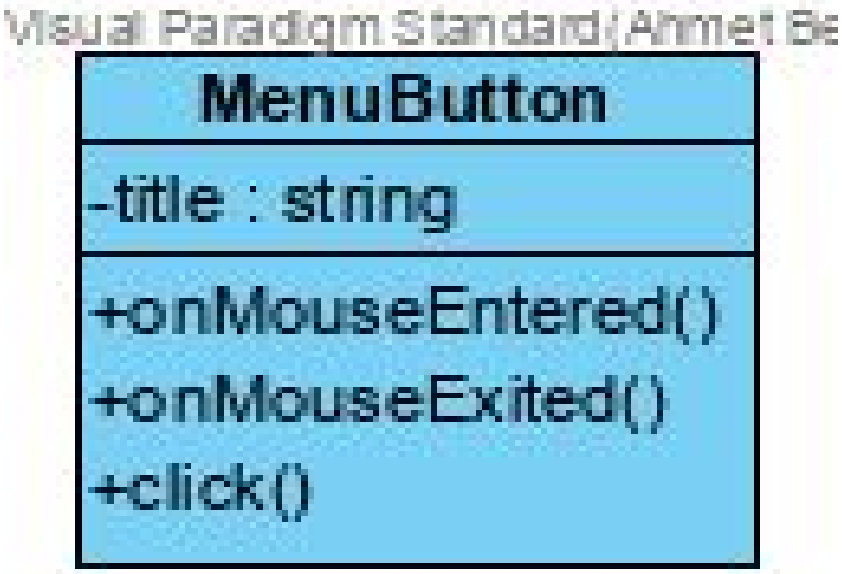
viii. Game menu



This is the abstract of the menus of the game.
Methods:

- private MenuButton buttons: this is options of the menu.

ix. Menu Button



This is the abstract of menu buttons
Attributes:

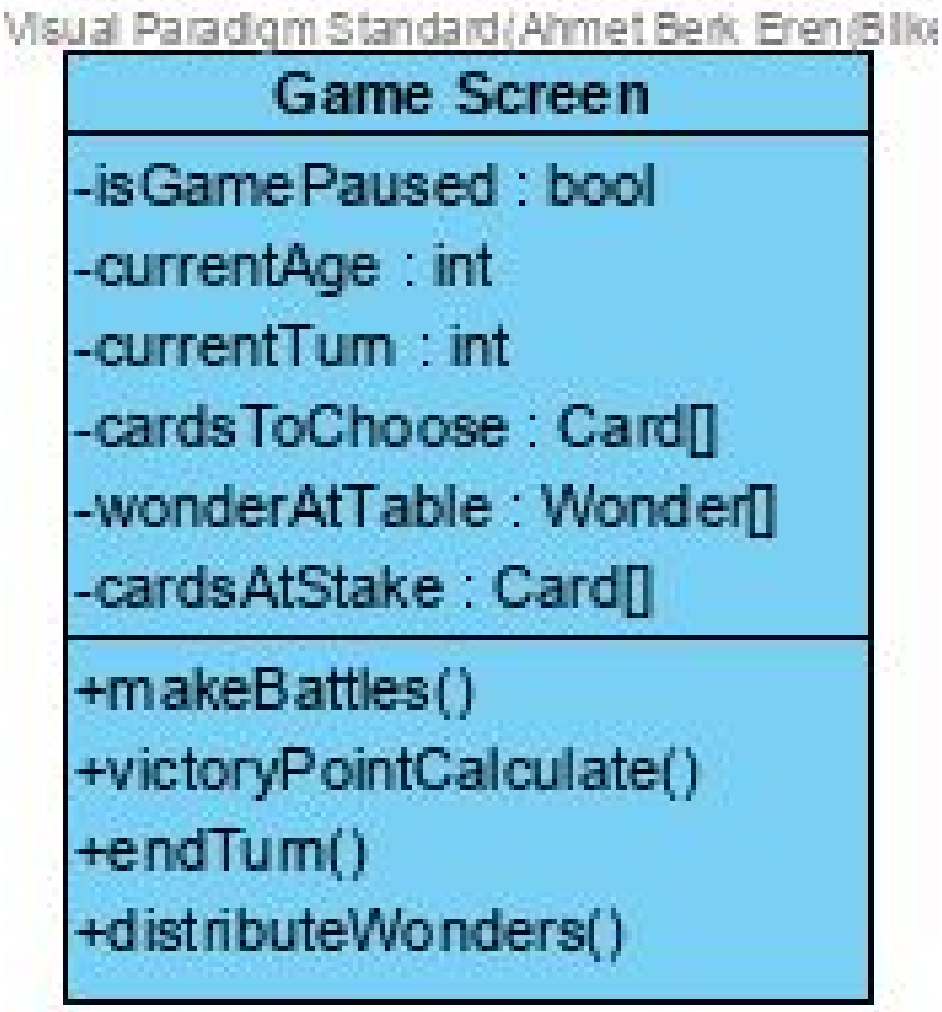
- private String title: the title of the button.

Methods:

- private void onMouseEntered(): this method activates when mouse entered the button.

- private void onMouse Exited(): this method activates when mouse exited the button.
- private void click(): this method activates when mouse clicked on button.

x. Game Screen



This class is the main user interface of the core game.

Attributes:

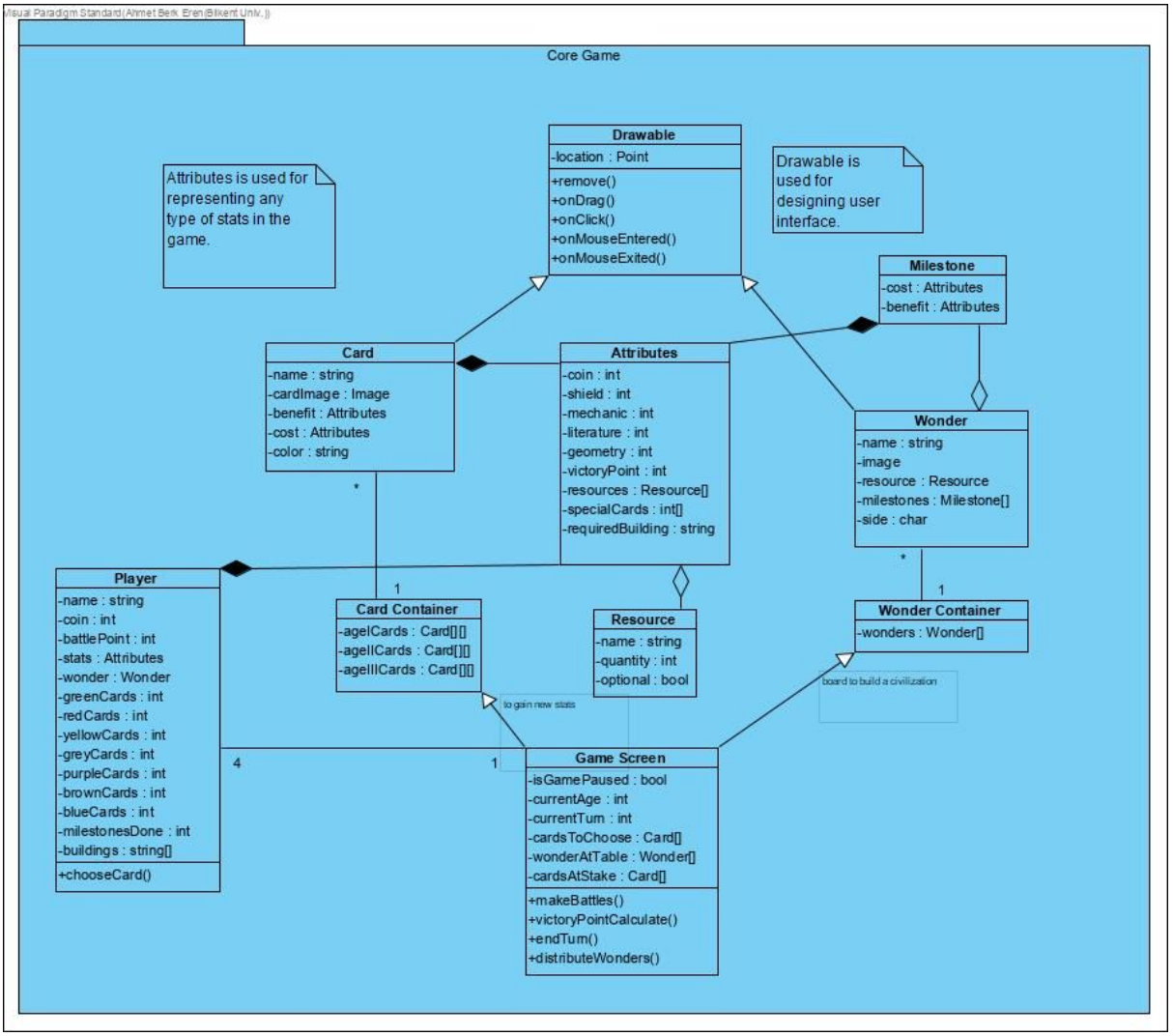
- private bool isGamePaused: This shows that the game is paused or not.
- private int currentAge: This shows the current age of the game.

- private int currentTurn: This shows the current turn of the game.
- private Card[] cardsToChoose: This are the cards which are shown to the user to choose in the current turn.
- private Wonder[] wonderAtTable: This shows players' wonders.
- private Card[] cardsAtStake: This keeps the cards which are discarded.

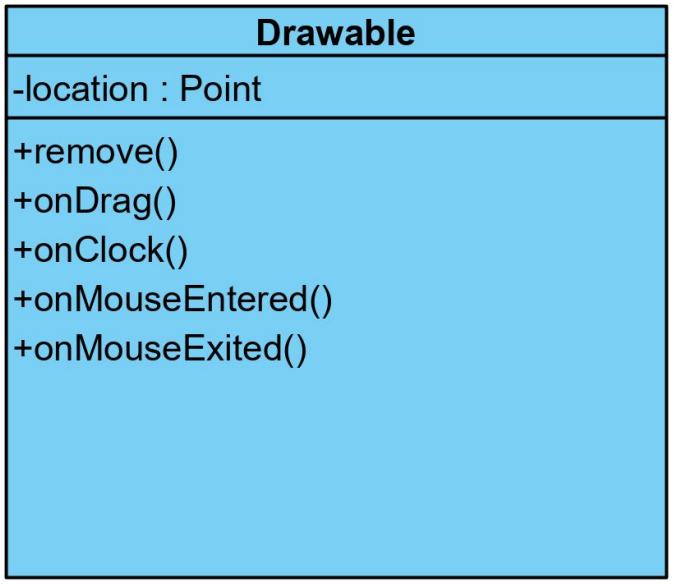
Methods:

- private void makeBattles(): At the end of each age, this method make the players make battles with its neighbors.
- private void victoryPointCalculator(): At the end of the game, this method calculates each player's victory points.
- private void endTurn(): At the end of each turn, this method updates wonders according to the cards which are selected by players.
- private void distributeWonders(): Before the game start, this method distribute the wonders to players randomly.

b. Core Game


Visual Paradigm Standard(Ahmet Berk Eren(Bilkent Univ.))
Core Game
Drawable
-location : Point
+remove()
+onDrag()
+onClick()
+onMouseEntered()
+onMouseExited()
Attributes is used for representing any type of stats in the game.
Drawable is used for designing user interface.
Milestone
-cost : Attributes
-benefit : Attributes
Card
-name : string
-cardImage : Image
-benefit : Attributes
-cost : Attributes
-color : string
Attributes
-coin : int
-shield : int
-mechanic : int
-literature : int
-geometry : int
-victoryPoint : int
-resources : Resource[]
-specialCards : int[]
-requiredBuilding : string
Wonder
-name : string
-image
-resource : Resource
-milestones : Milestone[]
-side : char
Player
-name : string
-coin : int
-battlePoint : int
-stats : Attributes
-wonder : Wonder
-greenCards : int
-redCards : int
-yellowCards : int
-greyCards : int
-purpleCards : int
-brownCards : int
-blueCards : int
-milestonesDone : int
-buildings : string[]
+chooseCard()
Card Container
-ageICards : Card[][]
-ageIICards : Card[][]
-ageIIICards : Card[][]
Resource
-name : string
-quantity : int
-optional : bool
Wonder Container
-wonders : Wonder[]
to gain new stats
board to build a civilization
Game Screen
-isGamePaused : bool
-currentAge : int
-currentTurn : int
-cardsToChoose : Card[]
-wonderAtTable : Wonder[]
-cardsAtStake : Card[]
+makeBattles()
+victoryPointCalculate()
+endTurn()
+distributeWonders()
1
*
4
1

i. Drawable



This is an abstract class used to design the user interface in the game.

Attributes:

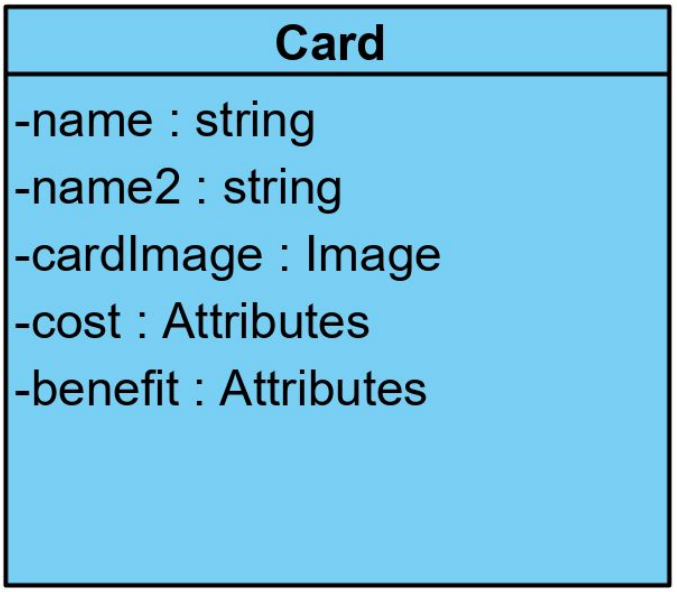
- private Point location: This attribute holds the position of the component.

Methods:

- public void remove(): This method is used to remove certain things from the interface.
- public void onDrag(): This method helps create events when drag occurs.
- public void OnClick(): This method detects clicks from the mouse.
- public void onMouseEntered():This method is used to create events when the cursor hovers over a certain area.
- public void onMouseExited():This method is used to create events when the cursor leaves a certain area.

ii. Card

Visual Paradigm Standard (Göktuğ (Bilkent Univ.))

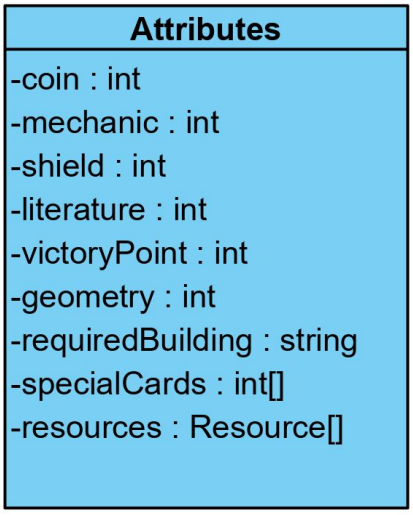


This class is used to hold all the information about a card. Since each card in seven wonders is special they all have a string which holds their name.

Attributes:

- private string name: This attribute holds name of the Card.
- private Image cardImage: This attribute keeps the image of the Card.
- Private Attributes benefit: This attribute holds the benefit the Card provides.
- private Attributes cost: This attribute holds the cost to build each Card.
- private string color: This attribute holds the color of the C

iii. Attributes

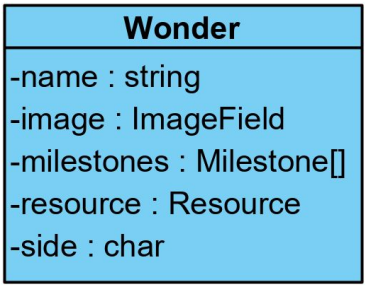


This class represents the type of stats that are used in Seven Wonders.

Attributes:

- private int coin: This attribute represents the currency used in the game.
- private int shield: This attribute represents the strength of shields in game.
- private int mechanic: This attribute represents the mechanic stat in game.
- private int literature: This attribute represents the literature stat in game.
- private int geometry: This attribute represents the geometry stat in game.
- private int victoryPoint: This attribute represents the victory point stat in game.
- private Resource[] resources: This attribute holds all types of resources in game.
- private int[] specialCards: This attribute keeps the cards with distinguished powers.
- private string requiredBuilding:This attribute holds the name of the building required to build a card.

iv. Wonder



This class is used to represent the Wonder cards in the game. The wonder cards are also all unique and have their own special names.

Attributes:

- private string name: Holds the name of the Wonder.
- private Image wonderImage: Keeps the image of the Wonder.
- private Resource resource: Keeps the unique resource each wonder provides to the player.
- private Milestone[] milestones: An array of Milestone to keep track of when the wonder could be improved.
- private char side: This attribute holds which side of the wonder card the player is using.

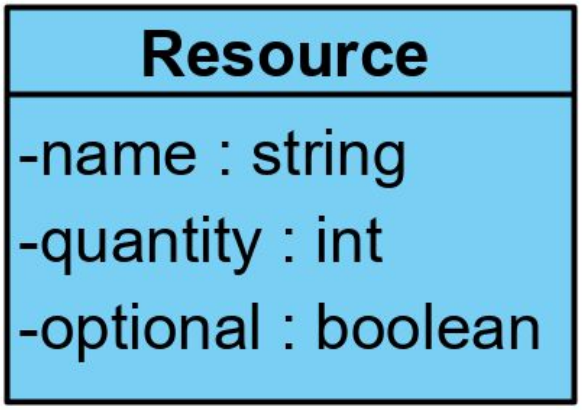
v. Milestone

This class keeps track of the milestones(required materials and etc.) to level your wonder up and the bonuses that will be acquired by levelling your wonder up.

Attributes:

- private Attributes cost: Holds the cost to level up the wonder.
- private Attributes benefit: Holds the benefit that will be gained when the wonders is levelled up.

vi. Resource

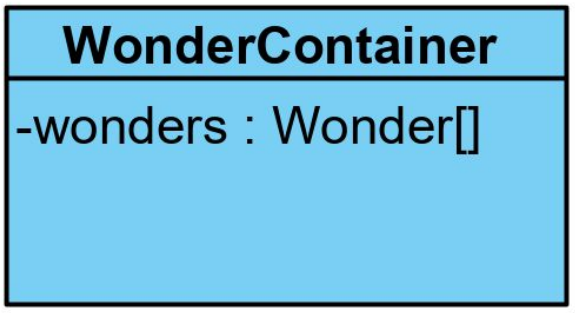


This class is used to identify resource types in the game.

Attributes:

- private string name: Holds the name of the resource.
- private int quantity: Holds the quantity of the resource.
- private boolean optional: Holds if resource is owned.

vii. WonderContainer

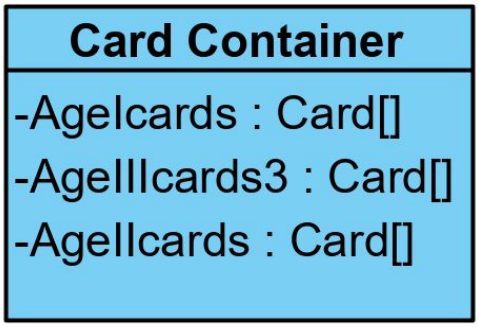


This class keeps all the unique wonders in the game.

Attributes:

- private Wonder[] wonders: Holds all the wonders in the game.

viii. CardContainer

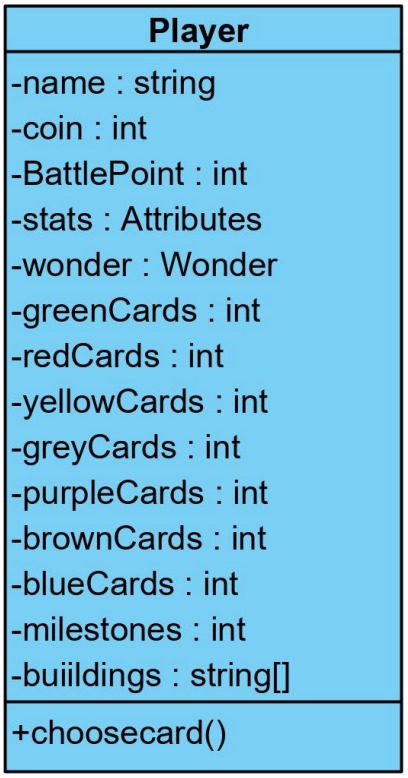


This class keeps all the cards from all three ages in the game.

Attributes:

- private Card[] ageIcards: Holds all the cards from Age I.
- private Card[] ageIIcards: Holds all the cards from Age II.
- private Card[] ageIIIcards: Holds all the cards from Age III.

ix. Player



This class represents and holds the values each player in the game will have.

Attributes:

- private string name: Holds players name.
- private int coin: Holds the players coin.
- private int battlePoint: Holds players battle points.
- private int greenCards: Holds the number of green cards the player has.
- private int redCards: Holds the number of red cards the player has.
- private int yellowCards: Holds the number of yellow cards the player has.
- private int greyCards: Holds the number of grey cards the player has.
- private int purpleCards: Holds the number of purple cards the player has.

- private int brownCards: Holds the number of brown cards the player has.
- private int blueCards: Holds the number of blue cards the player has.
- private Attributes stats: Keeps the stats of the player.
- private Wonder wonder: Keeps the wonder of the player.
- private int milestonesDone: Keeps track of milestones the player has completed.
- private string[] buildings: keeps the buildings the player has built.

Methods:

- public void chooseCard: this method is used when the player chooses the card they are going to play in that round.

4. Low-Level Design

4.1 Design Pattern

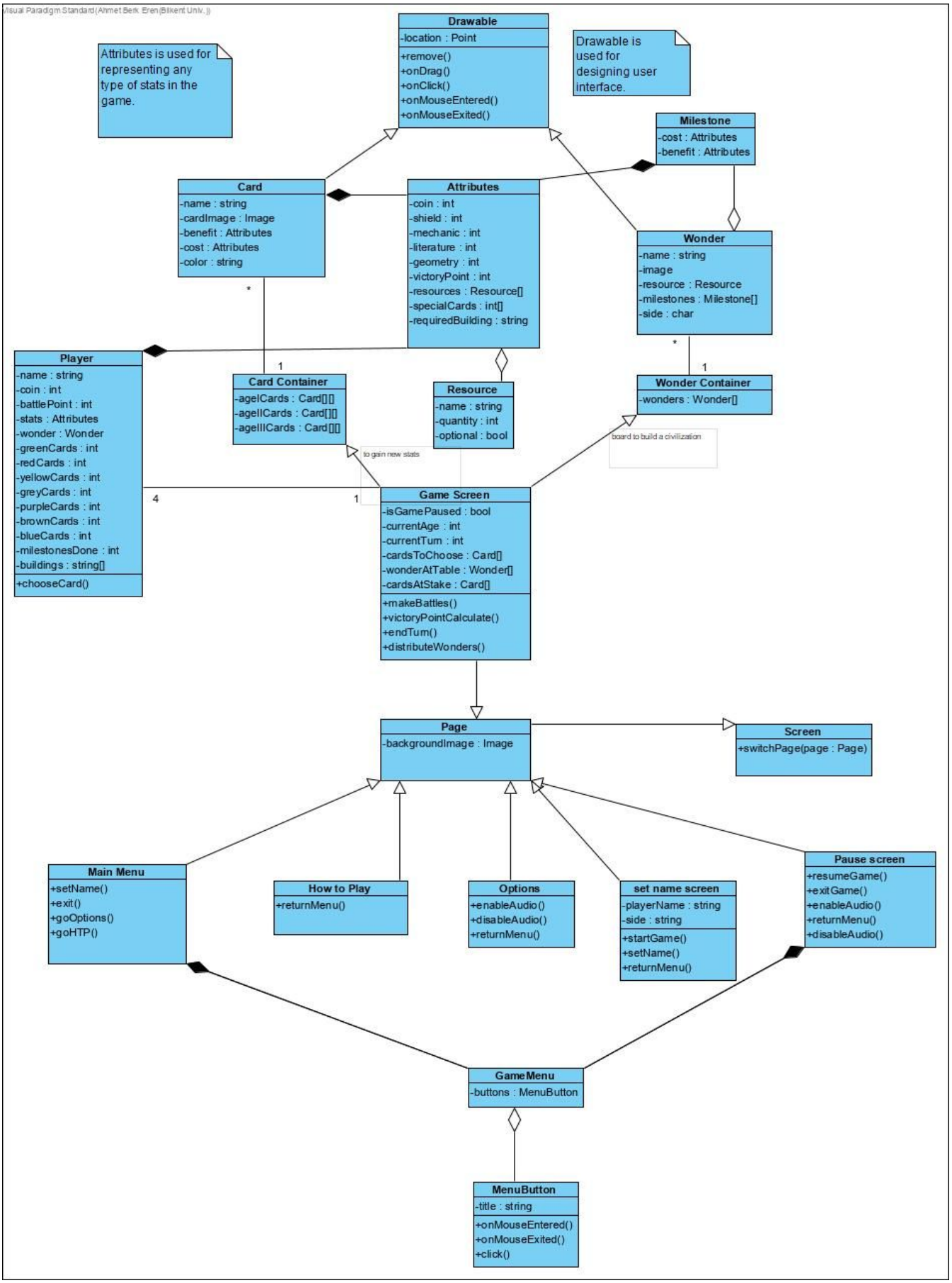
i. Builder Pattern

We will use Builder Pattern to create each and every card and wonder board object. Since these two classes have many pre-determined attributes, we wanted them to be created when the game is started and could not be changed after that.

ii. State Pattern

This pattern could be used to divide the gameplay into stages of Age I, Age II, Age III and End. Since each of the three ages has different decks of cards, using this pattern will make it easier to switch between stages and distribute age-specific cards.

4.2 Final object design



4.3 Packages

.com.javafx.*

JavaFx framework consists of graphics and media packages which provides to design, test, debug applications. One of the packages which javaFx have is javafx.animation, we will use this package for animating the nodes. We will use javaFx framework for building our user interface.

4.4 Class Interfaces

EventHandler<MouseEvent>

This interface will be used to get user outputs from the mouse. When user selects the card or switches the menus, this interface provides actions that will be taken. There is one method which is called *handle(MouseEvent event)*.

5. Glossary & References