

System Architecture Design

Advance Programming Practices (SOEN 6441)

Discworld\_Ankh Morpork – Build 3

Submitted To: - Prof. Steven Winikoff

|  |  |
| --- | --- |
| Document drafted by: Team 5 | |
| Deep Pandya | 7162391 |
| Meet Anadkat | 7122829 |
| Parikshit Pandya | 7585195 |
| Chintan Sarvaiya | 7148879 |
|  |  |

Game Architecture Design

# 1. Introduction

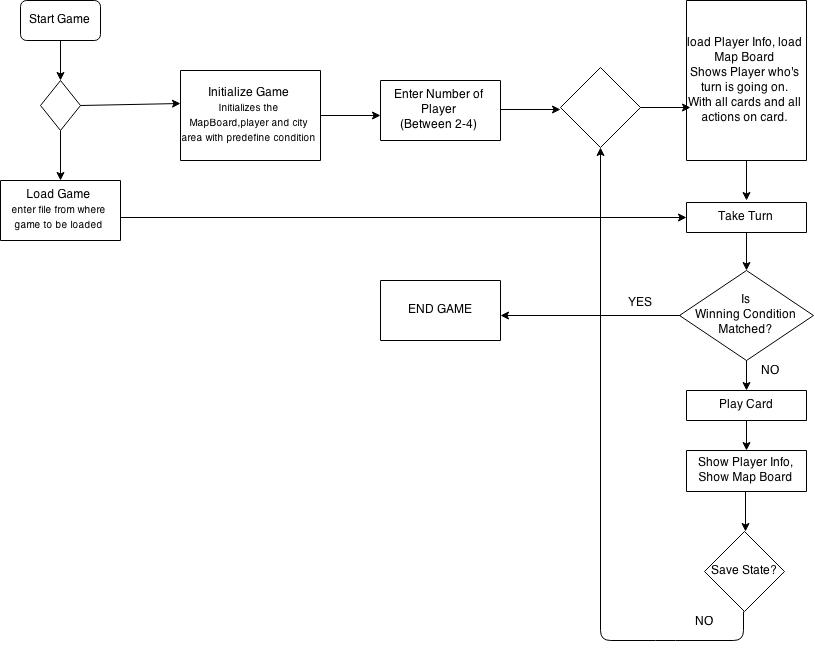
This document is to demonstrate the overview of the build 3 game project design as well, illustration of the architecture and functionality of the java implementation of board game called: (Ankh Morpork)

## 1.1Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of Ankh- Morpork board game. It presents a number of different graphical views to explain different features of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

# 2.Architectural Representation

The architecture of the Ankh Morpork board game uses the Model-View-Controller architecture pattern for implementing **user interfaces**. The architecture of the game is divided into three interconnected parts and internal component functionality as recommended by best object orientated practices.



## 2.1 Packages and Classes

There are mainly four packages. Source, PlayerCards, PersonalityCards, JUnit.

1. Source

-Start.java

-Player.java

-CityArea.java

-Helper.java

-MapBoard.java

-PayToBankFailureTest.java

1. Personality Cards

-SuperClass of personality cards (PersonalityCards.java)

-All Seven Personality cards

-for each personality each java class

1. Player Cards

-PlayerCardActions.java

-RandomEvents.java

-Each java class for each green cards, There are 48 Green cards.

1. JUnit

-Java classes for testing most of the rules of the game.

**2.2 Design Pattern**

Factory design pattern

We have used factory design pattern in our project.

Factory pattern is one of most used design pattern in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object. In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

The basic idea behind using factory design pattern is when your application needs to create objects dynamically by deferring its instantiation to the sub-classes.

In our design, we have separate hierarchy for green player cards classes and another hierarchy where these green player card objects get instantiated using factory design pattern. In this separate hierarchy, we let the subclasses decide on runtime on which green player card to instantiate based on the “name” parameter passed in the argument.

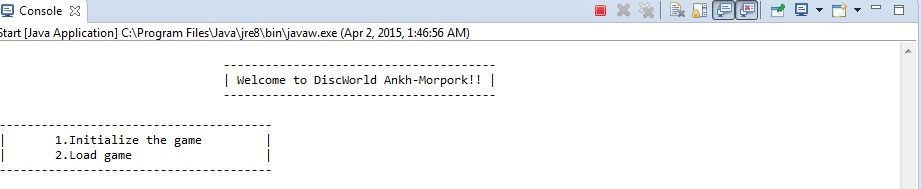
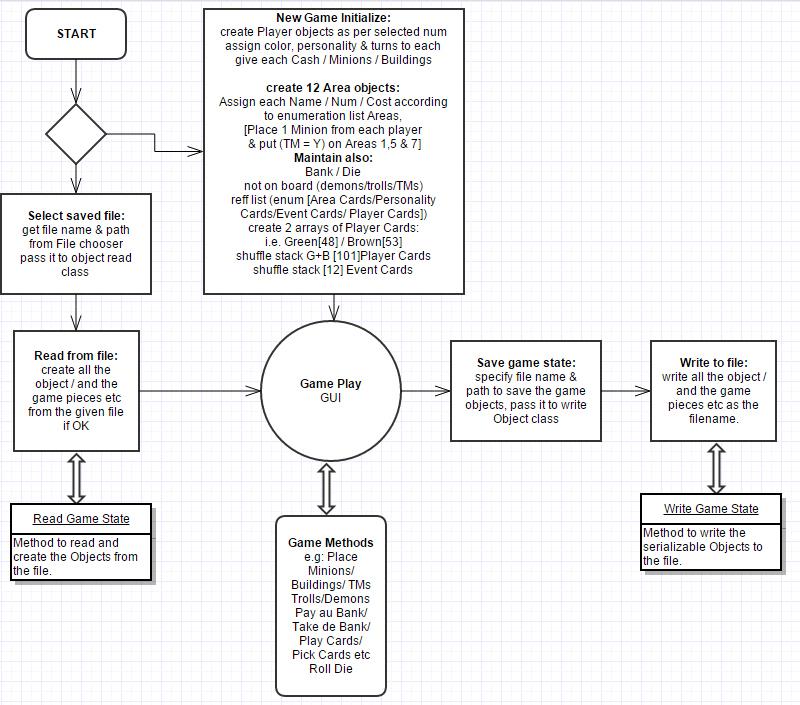
There’s class called **PlayerCardsFactory.java** which is factory class for dynamically initializing all the green cards objects.

# 3. Console Explanation

All variables and data structures are instantiated at load time, even City area and player is initialized with the pre define, load time values.

Various possible console output and state changes is explained below.

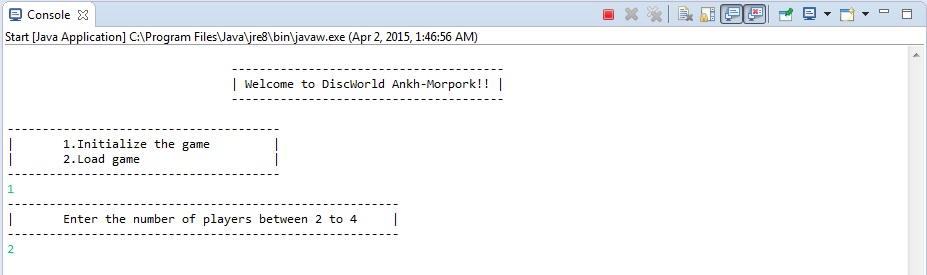
* 1. At the start of the program, you have two options, Initialize Game and Load game



Here, user has to give input, 1 for initializing the game and 2 for load game.

If he chooses 1, program initializes the game with the predefine condition and shows following option.

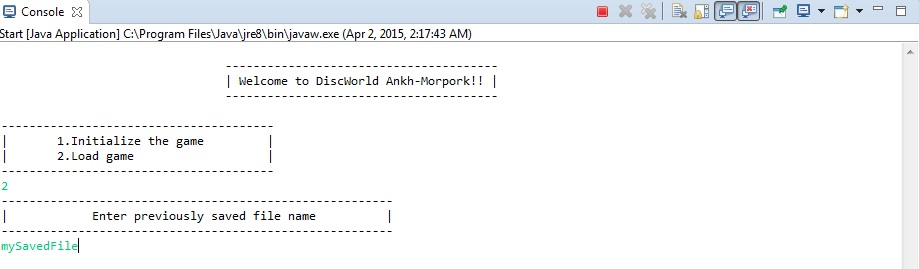
3.2Enter the number of player who is going to play the game.



Max 4 player can play and less than 2 is not acceptable. Players must be between 2and 4 only. Once you enter the player number, the game initialize according to that, assign area, color, turn and on basis of color it places the minion, building & troublemaker in city area that is define at initialization.

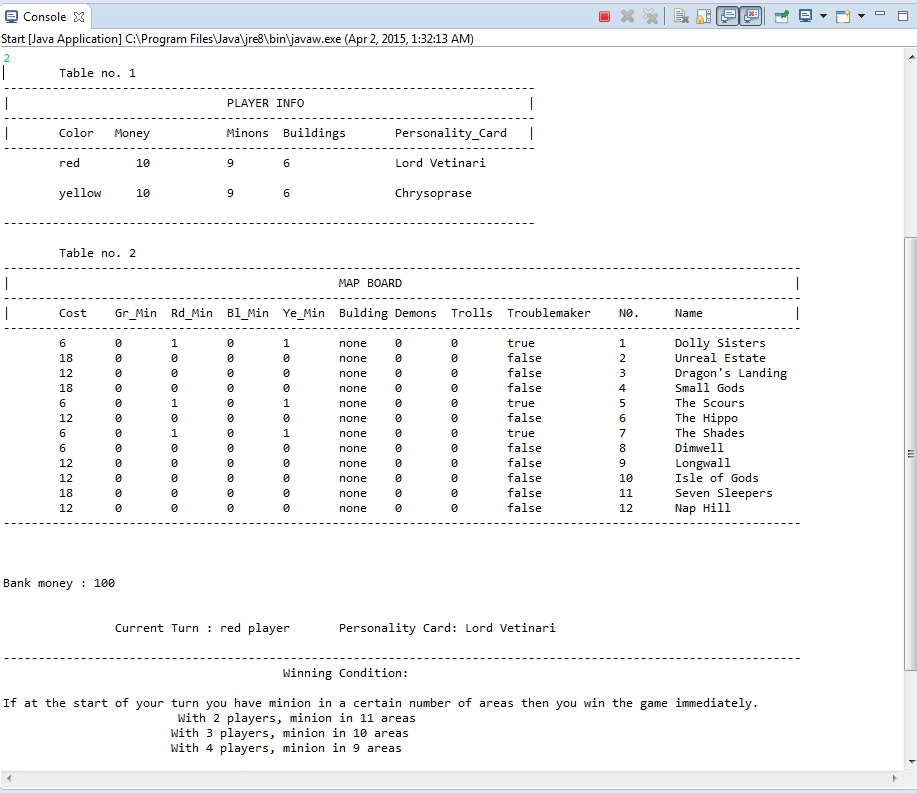
But if user chooses option 2, Load game than following action happen on console.

3.3Load game from file already been saved.



Enter the previously saved file name and game will start from there only.

3.4After initialization and loading program loads the player information, map board and shows one player whose turn is, with all cards in hand.



**PLAYER INFO**

Here, information of players who are playing is shown.

Like,

Color of player: we have entered two players at in previous step so it shows two colors, red and yellow.

Money that player holds: here at the start of game each player has 10$

Minions: Each player has 12 minions at the start of the start of the game, from 12, 3 minions are allocated to city area at time of initialization. So each player holds 9 minions on side.

Buildings: Each player has 6 buildings at the start of the game.

Personality Card: Each player has one personality card. Personality card is very important because it holds the winning condition. Each player must achieve status that is written on the personality card in order to win the game.

**MAP BOARD**

Map board shows the state of city area, it shows which city area has how many minions, building, troublemaker, demons, trolls, city area name and the cost of that city area.

Cost: Cost for placing the building in the city area.

Gr\_Min, Bl\_Min, Rd\_Min, Ye\_Min: Indicates Green, Red, Blue, Yellow color player minion

Buildings: It shows the status of city area, if city area has building, it shows color of player that indicates building belongs to that player.

Demons: Number of demons city area holds.

Trolls: Number of trolls city area holds.

Troublemaker: If city area has troublemaker then status would be true else it would be false.

Name: Name of the city area, there are total 12 city areas.

Bank Money: Money in the bank

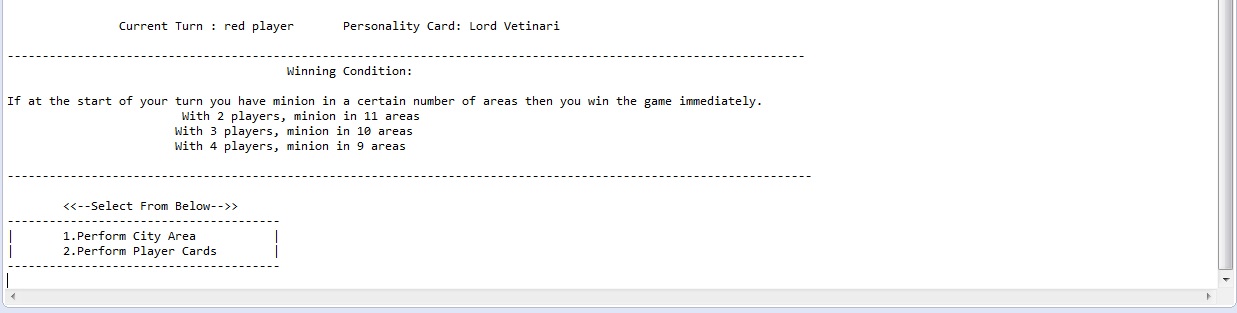
Then after, what you see is current player with all cards that player holds and the events on the cards.

Current Turn: Which player is currently playing?

Personality Card: Personality card that player holds from seven personality.

Winning Condition: Winning condition that must achieve by the player in order to win the game.

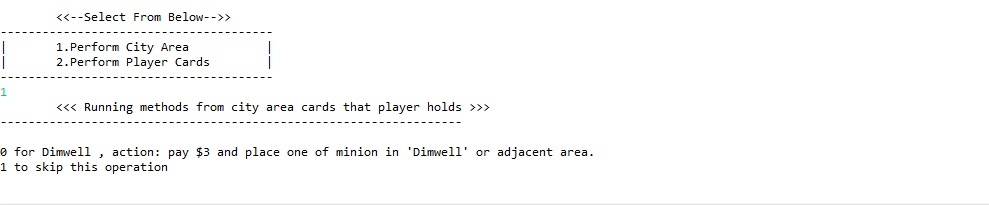
Next, you see on console is below.



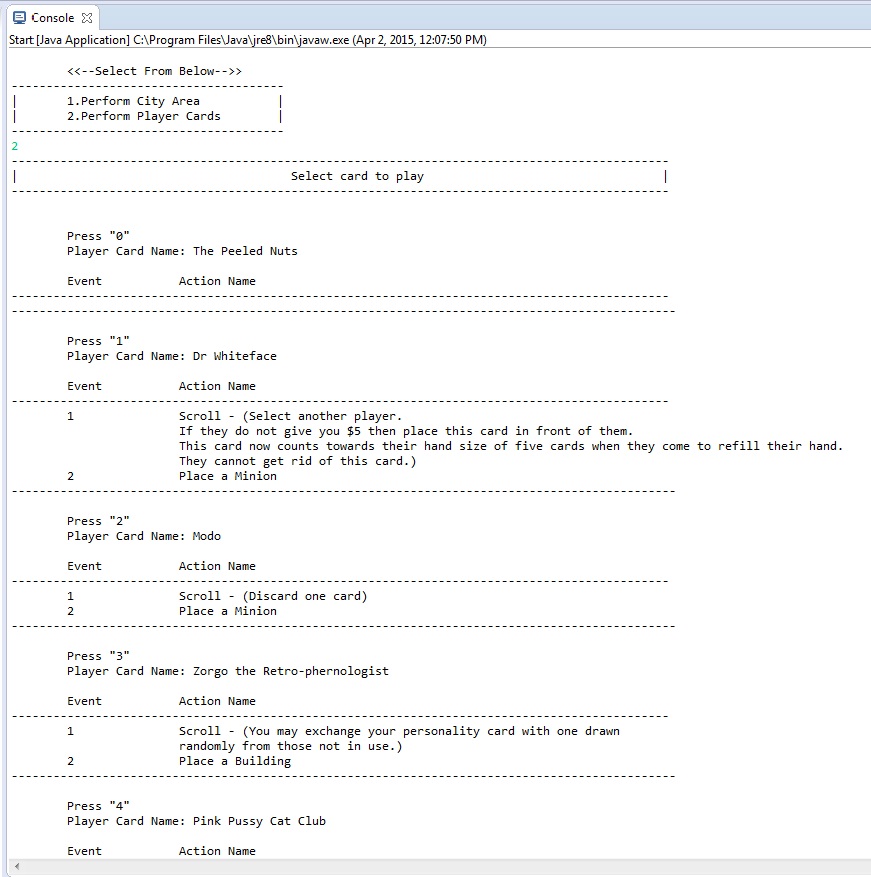
It will ask you to select any of 2 option,

1. Perform city area card
2. Perform Player cards

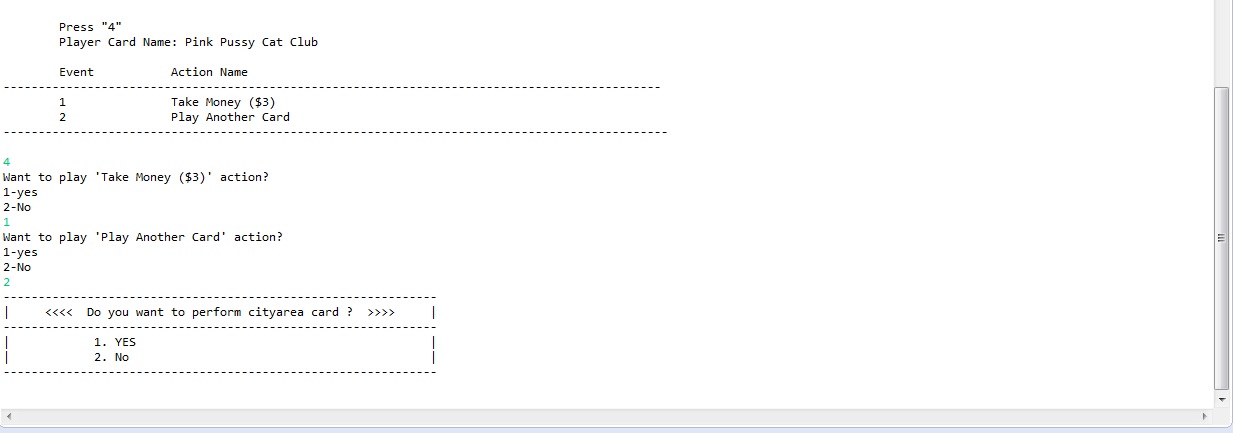
Option 1 to perform city area cards that player holds, you got below result,



Option 2 to perform player cards in hand, and you see below result on console,

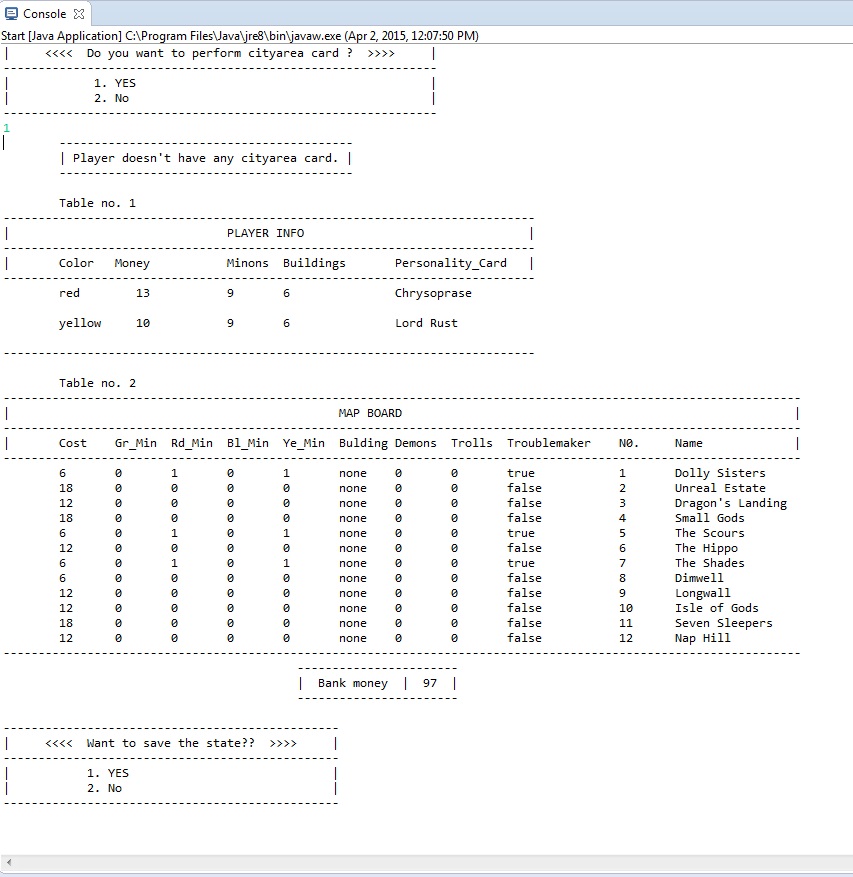


Now, depending on any choice to press “key”, it shows the following actions that selected card has on it,



After playing the card it ask for city area card if player wants to play or not?

If YES then it shows the city area cards that player has, and if player don’t have any city area card it shows following,



Player don’t have city area card, them it shows the player info and map board, bank money status and ask for saving the state.

And this is how game is continue to play until we got winner.