**Projects:**

DiskWorld build 1 Artifact

**Members of team 6:**

Parinaz Barakhshan (ID# 7675518)  
Niloufar Behrouzy (ID# 7676727)  
Gay Hazan (ID# 3974650)  
Lawrence Ng (ID# 5739411)

**Due Date:**

February 5th, 2015

**Design Diagram:**

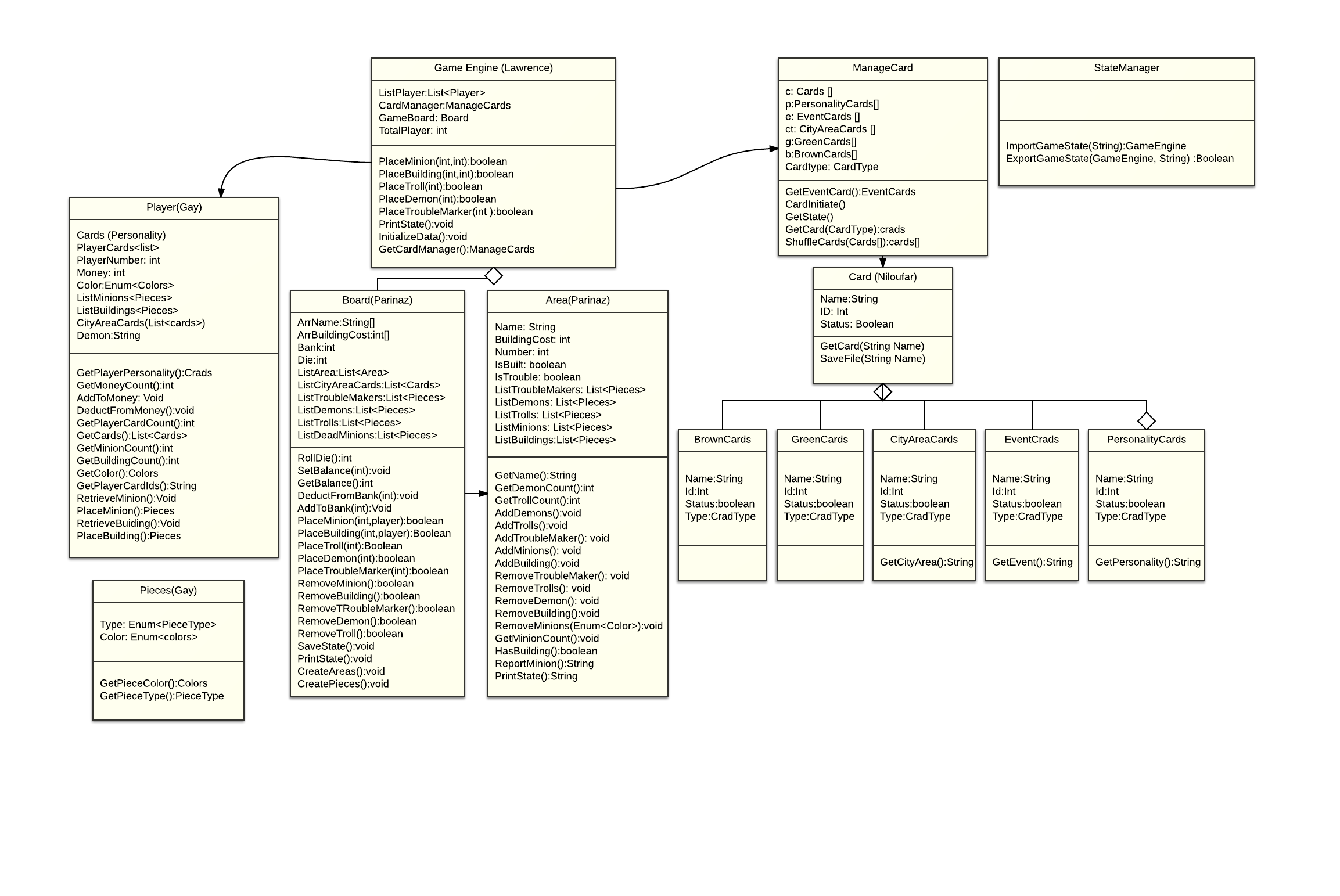
****

Table of Contents

0- Software Design overview

[1-Game engine 2](#_Toc410574194)

[2-State Manager 2](#_Toc410574195)

[3-Board 2](#_Toc410574196)

[4-Area 2](#_Toc410574197)

[5-Card Manager 2](#_Toc410574198)

[6-Cards 2](#_Toc410574199)

[7-Player 2](#_Toc410574200)

[8-Pieces 2](#_Toc410574201)

SOFTWARE DESIGN OVERVIEW**:**

When creating each class, we tried to keep in mind the class who are responsible to create and perform actions on that class and delegate the functions to that class (Responsibility Driven Design). Here is a list of interactions among classes.

* Main (console user interface) talks to GameEngine and StateManager.
* GameEngine contains a Board, CardManager and a list of Player.
* Board has a list of Area. It also provides functions to change the state of an Area.
* CardManager holds and interacts with all the cards.
* Player will talk to the GameEngine who will update their status during the game.
* StateManager is an independent class that is only responsible to import/export game state.

Additionally, we use the Facade Design so that customer only has to interact with one class who internally call the other classes to perform the actions. This design brings multiple advantages such as:

* Avoids overloading the player with all the internal modules.
* Separate the game into finite modules to avoid (or reduce) dependency scenarios.
* Easier to delegate work to separate programmers.
* Opens the door for adding user interface if later build requires us to (Model-View-Controller patern).

CardManager is a decision we took because it will be easier on later build to associate action to each card and provides a smart interface for GameEngine to shuffle, get a particular card, etc.

The StateManager made use of Java ObjectInputStream/ObjectOutputSream to save the GameEngine in a text file.

**Assumptions:**

* During the start of the program, 5 greencards will be distributed to each player.
* Each area can have at most one TroubleMarker or one building.
* Each Area can have one or more troll pieces or demon pieces
* Each player will get 10 dollars at the beginning

DETAILS ARCHITECTURAL DESIGN:

In order to achieve the goals defined in build 1 of our project we break it down into 8 classes. Below I am going to define the function of each of these classes:

1-GameEngine is like the heart of this program. It calls the appropriate functions mainly defined in other prominent classes like player, Card Manager, and board and coordinates the operations.it is responsible for the interactions that happen between the other classes. It does time and operation management.

2-StateManager: when the player saves the programs this class gathers all the information and states of other classes into it so that when the program reloads it gives back the saved states to each classes so that it can be reloaded from the point it was saved.

3-Board**:** owns Die, money in the bank and 12 Areas.it also keeps track of the pieces in each area. And it has some pieces like Trouble Markers, Trolls and Demons.it has the ability to place or remove pieces to/from Areas. It also has the functions for initiating the areas and pieces of the board.

4-Area**:** this class defines the attributes and functions of each area which includes the name, the assigned number, and the building cost of each area.it returns the number of pieces in each area.it also provide the possibility of adding or removing pieces to/from each Area. It also shows if an area is built or has trouble Marker in it.

5-CardManager: is responsible for managing the distribution of the cards based on the command that it receives from Game Engine.

In the constructor of this class we initiate 7 Personality, 12 City Area and Event, 48 Green and 53 brown cards and during the game we can get a card and use its attributes and functions and also put it away from the game by making its status to false.

6-Cards: we divided them into 5 groups. Event Cards, City Area Cards, Personality Cards, Green Cards, Brown Cards. All of Cards Classes will inherit from a class of Cards that hold 4 common attributes: name, id, type and status of cards and function that a card may have.

At the beginning of the game cards will be initiated via a call from Game Engine. All of their status will be true (available) until a request from Game Engine comes to get a cards then the cards would be shuffled and status put to false before giving the card.

7-Player: player class defines the attributes and functions that each player can have. In this class we get the personality card that each player has from Game Engine. List of player cards is another attribute that we keep; we should make sure that each player has 5 cards in his hand. We also allocate a number and a color to each player in the construction class which defines that player. Each player owns 12 Minions and 6 buildings that we constructed with two separate lists and also some money. Depending on which areas the player build the buildings, he will get a city Area Card which we have implemented it with list. We can retrieve the amount of money that belongs to each player from this class, and we can add to that money or deduct from it. We can check the number of cards that player holds.

8-Pieces: in this class we have defined two attributes. First is the Piece type which is of type Enum and it contains ‘**Minion, Demon, Troll, TroubleMarker, and Building’**. The other attribute is color that defines the color of different pieces .As we have different pieces with different colors we have defined these colors: **None, Red, Green, Blue, Yellow, Orange, Brown, Black.** The reason for defining none as the color is for when the color of the pieces is not important. For example I want to know the total number of minions on the board and the color is not important in that case I send none to the method and I get the total amount in return.or when I want to know about the minions in an area.in case minions are there the colors would be printed but if no minion is there none would be printed.