

Projects:

DiskWorld build 3 Artifact

Members of team 6:

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

Due Date: March 12th, 2015

# **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-Action**

# **7-Card Action**

# **8-Cards**

# **9-Player**

# **8-Pieces**



**Software Architecture Overview**

In this build the main structure of our classes did not changed however we benefit from certain software architecture pattern to improve our structure and make it more flexible. Here is a list of patterns used in this build:

* The **Factory** pattern was used to encapsulate the details of card creation in Manage Card class. The primary responsibility of this class is to initialize all types of cards and their relevant actions at the beginning of the game .we took all the details code out and put them in separate methods.
* The **Strategy** pattern was used to treat Action Cards as an independent component so modification in Cards class (original class) would be more reusable and efficient.
* In order to facilitate the interaction of users and the game, we made use the pattern **Facade** to hide among classes. The user will only use the public functions of the GameEngine to play the game. The GameEngine will then call each of the other classes to perform the selected actions.

Here are some other decisions we took during our project for build 3:

* In order to ensure that everything is functional, we created a Junit test for each cards used in the game. In fact, we created those Junits before the actual code to use them as per the TDD model.
* Each card information is saved independently and loaded independently. By doing so, we were able add / remove /fix cards independently. We also have the brown cards ready but it has not been tested yet. Here is an example how we stored Mr Bent green card in our game.

|  |
| --- |
| name= Mr Bent  verb=loan; object=10$ bank; condition=pay back 12$ or lose 15 points endgame |
| symbol=S,C |

DETAILS ARCHITECTURAL DESIGN:

In order to achieve this build goals we add certain methods to our classes to accelerate their interactions. Below introduce the main methods added in each class and a short description of their responsibilities.

1-GameEngine As a heart of our game this class has to have key methods to call and manage the game.

* PlayEffect : this method will be called whenever a player turn start in order to apply actions based on cards that player is going to play.
* ActivateCityAreaEffect
* Play Card
* A group of validation methods to check user inputs.
* A group of method to place pieces in the game
* Initialize all data for starting the game; including setting up the board and give the player a personality and a hand.
* Determine the first player.
* Print the state of game.

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:

* Methods to place or remove pieces on the bored.
* Methods to locate pieces on the board.

4-Area:

* Methods to perform basic action for pieces on the board; add or remove.
* Methods used to call when especial decision has to be made to follow the game rules.

5-CardManager

* Showing the image of cards.
* Create Cards

# 6-Action: use as an abstraction class it illustrates the action of cards which is consist of verb: type of action, object: the things an action should be applied to and condition that give us the opportunity to perform dependent rules existed in cards**.**

# 7-Card Action

* Create a list of actions for board game cards.

8-Cards

9-Player

* The ability to add or remove its pieces on the board
* Ability to play cards

10-Pieces