**SOEN 6441**

**Advanced Programming Practices**

**Summer 2015**

**Software Architecture**

“[Lanterns: The Harvest Festival](http://foxtrotgames.com/lanterns/)”

**{**Build-1**, Build-2,** Build-3**}**

(Team-D)

Khan, Muhammad Raza

Momo Vofo, Patrick

Muco, Brice

Patil, Sushil

Puranik, Tejas

Ragunathan, Vinod Kumar

# Introduction

This document provides a brief architectural overview of the system. The module view of “[Lanterns: The Harvest Festival](http://foxtrotgames.com/lanterns/)” is explained in detail using class diagram. The description covers the features implemented in both the builds.

Fig 1: Class Diagram of “[Lanterns: The Harvest Festival](http://foxtrotgames.com/lanterns/)” application

# Description and Functionalities of the modules in the class diagram

**Board**

Description:

The game board of “Lanterns: The Harvest Festival” is created.

Functionality:

This class implements the below functionalities:

**Build 1:**

Initialize the game board.

Shuffle the lake tiles.

**Build 2:**

Display the current state of the board

**CardToReturn.java**

Description:

It deals with handling three types of dedications, corresponding to the three stacks of Dedication Tokens.

Functionality:

This class will handle the below operations

**Build 2:**

1. Identify the color of the lantern cards that player wants to dedicate
2. Identify the type of dedication
   1. Four of a kind,
   2. Three pair, or
   3. Seven Unique
3. Make appropriate increment and decrement of stacks from both the side i.e. player and general pile.

**FavorTokens**

Description:

The player may spend two favor tokens to exchange one of his lantern cards for a different lantern card. The players can receive tokens if any of the matching tiles have platform on them. The favor token is also used to determine the winner in case of a tie.

Functionality:

This class implements the below functionalities:

**Build 1:**

Initialize favor tokens.

Retrieve the number of available token in the stack.

Increment and decrement the tokens in the stack.

**Build 2:**

**Note:** No Change

**GameEngine**

Description:

It deals with initialization, loading and updating the state of the game.

Functionality:

This class implements the below functionalities:

**Build 1:**

Initialize the game.

Load the existing game from the xml file.

Save the game.

Save the current state of each player.

Save the lake tile structure from the board.

Load the state of each player.

Recreating the lake tiles on the board,

Load the state of lantern cards from file.

Save the state of lantern cards.

Load the state of dedication tokens from file.

Save the state of dedication tokens.

Load the state of dedication tokens from the file.

**Build 2:**

Display the start lake tile on the board.

Allocate each player with 3 lake tiles at the start of the game and display the attributes of the tiles.

Display the lake tiles present on the global stack.

Display the details of dedication and favor tokens.

Display the number of lantern cards of each of seven colors available on the stacks.

Invokes the player’s turn and allocate lantern cards to each player accordingly.

Provide options for players to select any of the below and compute accordingly

1. Placing the lake tile on the board -
2. Players decide on where to place the tile i.e. on which adjacent position.
3. Players decide on what side of Lake Tile should be placed on the specific adjacent position.
4. Exchange of lantern cards
5. Display the number of favor token that each player has
6. Display the number of lantern cards of each color that each player has
7. Players specify which lantern card to pick and to return.
8. Decide if the exchange is successful or not based availability of cards and tokens
9. Make dedication
10. Display the number of favor token that each player has
11. Display the number of lantern cards of each color that each player has
12. Players specify the type of dedication and the computations occurs accordingly
13. Exit

Display the status of lantern cards that each players hold after the Lake Tile is being placed

**LakeTiles**

Description:

Players receive three lake tiles each. The starting lake tile is placed on the center of the play area. There are total of 36 lake tiles and one start tile. If the color on any side of the newly placed Lake Tile matches the color on an adjacent side of another Lake Tile, the active player receives a bonus Lantern Card of that color. If any of the matching Lake Tiles (including the newly placed tile) have Platforms on them, the active player receives one Favor Token for each Platform.

Functionality:

This class implements the below functionalities:

**Build 1:**

Load the lake tiles from file.

Initialize the lake tiles based on number of players.

Allocate tiles to players.

Retrieve tile from the general stack.

Place the tile on the game board.

Generate random colors on tiles.

**Build 2:**

Rotate Lake Tile based on the degree (90, 180, or 270).

**LanternCards**

Description:

There are 56 lantern cards with 7 different colors. Players receive lantern cards corresponding to the color on the side of the starting Lake Tile he is facing. The player with red lantern card starts the game.

Functionality:

This class implements the below functionalities:

**Build 1:**

Initialize the number of lantern cards depending on number of players for both the new and existing game.

Increment and decrement the lantern cards from the stack.

Retrieve the count of lantern cards from the stack of each color.

**Build 2:**

Display the number of lantern cards based on the color of the card

Determine if the lantern card of particular color present in the corresponding stack or not

Add card to the stack

Get card from the stack

Identify non-empty lantern card stacks

Assignment of lantern Cards to each Player according to Lake Tile and Player’s position

Retrieve favor token and assign to players if platform present in the lake tile

**Lanterns Application**

Description:

The main application container which loads the new or existing game based on the user input.

Functionality:

This class implements the below functionalities:

**Build 1:**

User Interface

\ Load new and existing game.

Save the current state of the game.

Display game in the text mode.

**Build 2:**

**Note:** No Change

**Player**

Description:

The entities related to players are managed.

Functionality:

This class implements the below functionalities:

**Build 1:**

Initialization of player attributes for both new and existing game.

Increment of player’s lantern card stacks and decrement of common lantern cards stack while player place the lake tiles.

Increment player’s honor as he/she receives dedication tokens and decrement tokens from the respective stack.

Increment player’s favor token score and decrement the favor token from the stack.

Maintain the player’s lake tile stack.

Determine which lake tile has been placed on the board by a player.

Manage favor tokens while players exchange lantern cards.

Manage lantern cards in both player and common stacks during the exchange.

Retrieve the player’s favor token score.

**Build 2:**

Determine if the player has respective lantern cards to do dedication of type that he/she claim to have.

Decrement the players’ lake tile stack when they place the tile on the board

Assign location to Players with respect to Game Board

Determine which player should play currently

# Relationship between classes

In build2, one new class has been implemented and the rest of the classes are same as in build 1.

|  |  |  |
| --- | --- | --- |
| Class (C1) | Class (C2) | Relationship Description |
| GameEngine | CardToReturn | C1 has an instance of C2 to make appropriate modification in the stacks of lantern cards while players make dedication. |

# Reasoning of choosing the corresponding data structures

In build 1, stack data structure was used to maintain the lantern cards. Each stack have lantern cards of specific color. We found that the Hash Map data structure would be better and appropriate to maintain the lantern cards based on color. The reason is mentioned below,

The count of lantern cards of respective color is frequently retrieved and manipulated in the game. Using the color as key and the number of lantern cards of the specific color as value would be helpful to perform the above operation with ease.