**SOEN 6441**

**Advanced Programming Practices**

**Summer 2015**

**Software Architecture**

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

(Team-D)

Khan, Muhammad Raza

Momo Vofo, Patrick

Muco, Brice

Patil, Sushil

Puranik, Tejas

Ragunathan, Vinod Kumar

Contents

[I) Introduction 3](#_Toc424752879)

[II) Description and Functionalities of the modules in the class diagram 3](#_Toc424752880)

[III) Relationship between classes 8](#_Toc424752881)

[IV) Reasoning of choosing the corresponding data structures 9](#_Toc424752882)

# 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.

****

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:

Initialize the game board.

Shuffle the lake tiles.

**DedicationTokens**

Description:

Dedication tokens are separated by type into 3 stacks and arranged in descending order of value. The total of 30 dedication tokens includes four of a kind (9 tokens starting from the value 8), three pair (9 tokens starting from the value 9), seven unique (9 tokens starting from the value 10), and generic dedication token (3 tokens each with the value of 4).The value represents the honor a players earns from the dedication.

Functionality:

This class implements the below functionalities:

Initialization of dedication tokens for both “New Game” and “Existing Game”.

Maintain three primary types stack and one generic type stack.

Retrieve the count of cards of each type from the stack.

Print the state of corresponding dedication tokens by type.

**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:

Initialize favor tokens.

Retrieve the number of available token in the stack.

Increment and decrement the tokens in the stack.

**GameEngine**

Description:

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

Functionality:

This class implements the below functionalities:

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.

**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:

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.

**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:

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.

**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:

User Interface

\ Load new and existing game.

Save the current state of the game.

Display game in the text mode.

**Player**

Description:

The entities related to players are managed.

Functionality:

This class implements the below functionalities:

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.

# Relationship between classes

|  |  |  |
| --- | --- | --- |
| Class (C1) | Class (C2) | Relationship Description |
| LanternsApplication | GameEngine | C1 has an instance of C2 to save or load the state of the existing game and initialize the new game. |
| GameEngine | Player | C1 has an instance of C2 to initialize the player attributes like lake tile, favor token, dedication tokens and lantern cards and maintain the attribute value as the game progresses. |
| LakeTiles | C1 has an instance of C2 to initialize Lake tile attributes based on the number of players, save and load the state of lake tile in the existing game. |
| LanternCards | C1 has an instance of C2 to initialize lantern card attributes based on the number of players, save and load the state of lantern cards in the existing game. |
| FavorTokens | C1 has an instance of C2 to initialize favor token attributes, save and load the state of favor token in the existing game. |
| DedicationTokens | C1 has an instance of C2 to initialize dedication token attributes, save and load the state of dedication token in the existing game. |
| Board | LakeTiles | C1 has an instance of C2 to initialize the board with the start lake tile and shuffle the lake tiles. |
| Player | LakeTiles | C1 has an instance of C2 to maintain the lake tiles own by each player in their respective stacks. |
| LanternCards | C1 has an instance of C2 to maintain the lantern cards own by each player in their respective stacks. |

# Reasoning of choosing the corresponding data structures

1. Array List is used to maintain

The lake tiles placed on the board

The player list

The lake tiles hold by the players

Array list can grow or shrink its size dynamically. The Array list doesn’t demand to specify the size during its declaration.

1. Stack is used to maintain

The dedication token based on primary and general type of lantern cards

The favor tokens

The lake tiles in the general pile

The lantern cards based on the color

The above objects are retrieved on the basis of last in first out so the appropriate data s structure which satisfies the objective is Stack.