Bilkent University

Department of Computer Engineering

CS 319 - Object-Oriented Software Engineering

CS319-2F-DE: Defender

Term Project - Final Report

Iteration 1

- **Project Name:** Defender
- **Group No:** 2F-DE
- **Group Members:** Büşra Ünver, Celal Bayraktar, Javid Haji-zada, Samir Suleymanli, Selen Uysal

1. Introduction

We begin to write out program with respect to classes and design patterns we decided during requirements elicitation and system design process. Implementation progress is done on intelliJ IDE mainly and GitHub repository is used to code synchronously between different environments. As of this report's date implementation includes some of main game screens and functionalities of Defenders game.

2. Implemented Functionalities

For the first demo, we have implemented Main menu screen, buttons in the main menu, game play screen, how to play and exit. We prioritize the functionalities that provided main functions of Defenders game. We didn't spare much time on developing UI to see the functionality faster.

3. Design Changes

While coding the project, we decided to remove some classes such as screens and managers since they were redundant. In this section, we will discuss which changes we made.

3.1. Changes in the Types of Classes

We have changes Screen class to abstract class which was interface before. We did this change in order to override our methods of Screen class easily when we need to or just use as it is in the parent class.

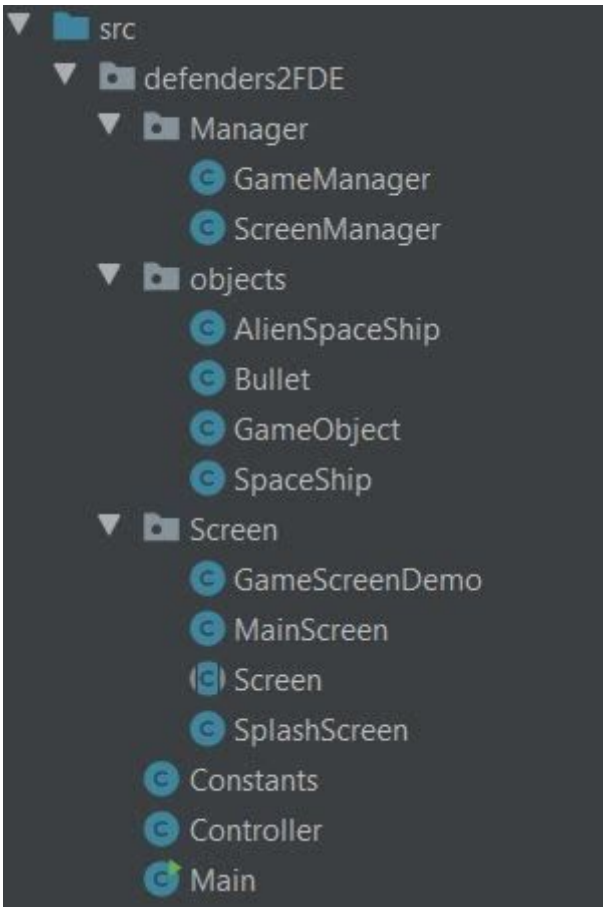
3.2. Additional Classes

A new public class Constants added to initialize and hold constants such as screen sizes. We didn't included it in analysis and design reports.

3.3. Merged Classes

For this iteration we didn't use separate manager and screen classes in order to save time and merged some of them. For instance, we didn't write second class MainMenuManager and included managing type coding inside MainMenuScreen. The situation is the same for InputManager class. We just handled the input events

inside the GameScreen class. For the next iteration, we will make the manager and screen classes separate.



4. Lessons Learnt

During the first implementation phase, we could not properly manage task division. We tried to divide activities into tasks and tasks into subtasks. Our implementation progressed as follows:

Task #1

- Implement an algorithm to determine collisions of game objects.

Task #2

- Implement ScreenManager
- Implement SplashScreen
- Implement MainScreen
- Implement GameScreen

Task #3

- Implement GameObject classes: GameObject, SpaceShip, AlienSpaceShip, Bullet, their attributes and operations.

Task #4

- Implement MainScreen buttons, Play, Exit, How to Play and their functionalities.

Some of our group members also learnt how to collaborate through VCS (Version Control System), in our case Git.

5. Users Guide

5.1 System Requirement & Installation

The game can run any operating systems that supports Java runtime environment. Game is not required any serious memory and storage to run game because the game uses small amount memory which is around 200 MB RAM and storage which is around 300KB HDD.

5.2 How to Use

5.2.1 Play Game

The players needs to survive as long as they can. Players should shoot the blue boxes and avoid their bullets. If player receive a shot from blue box, he/she will be eliminated. Every blue box destroyed gives 100 points.

5.2.2 How to Play

To see key bindings of the game, the player needs to click “How to Play” button on the main menu. As a result key bindings of the game will be displayed on a pop up screen.