EECS 3311 - Lab 05
Design Report

Amir Mohamad, Saniz Momin, Akif Prasla, Hasnain Saifee

TA - ta name

Nov 7, 2021

Contents

1 Introduction

The software project is about generating a soccer game where there are two players: a goalkeeper and a striker. You have 60 seconds to score as many goals as possible, after each goal is scored your score increments and the game is paused. You can resume the game by pressing key R or by going to the menu option selecting control which has a sub menu resume and pause. To score a goal the striker can move by pressing up down left right key and to score a goal striker has to press space key. The goalkeeper randomly defends the gate by moving in the left and right direction bounded by the gate's length. After the timer becomes 0 the user gets the statistics of how many goals the striker has scored and how many saves has the goalkeeper made. When the game is over, the user can start playing again by navigating to the menu bar, clicking on the Game menu and clicking to a new game or exit if the user wants to end playing. The main-goal of this software project is to use OOP concepts and efficient software design patterns to write code efficiently and follow the DRY principle which is (Don't repeat yourself). In this software project, to properly organize our soccer game system, the system is divided into a set of layers where each layer focuses on a specific aspect and has an abstraction level. MVC architectural pattern is used to divide the system into three main subsystems where the model comprises data of the system, for example keeping record of total goals and saves of striker and goalkeeper respectively. The view consists of the user interface part where the data in the model subsystem can be viewed. The controller which handles interaction between the modal and the view. The detailed design of the software system can be seen using creational design patterns such as factory design pattern as well as prototype design pattern which focuses on creation of objects. For example players are created via the PlayerFactory class which has getplayer method which takes string as parameter. All that is needed to pass is a string and it will return the appropriate object. Doing a project with partial implementation takes time to understand the logic and the functionality of code written. It took a while until we all understood what is the role of each subsystem, once we got that it was easy to implement and finish the rest. Based on our understanding of this software project we are going to write the report with the basic idea and the designs implemented for the top level and organization of the software project. We will create a UML diagram to make our explanation of the design patterns used more clear. We will then explain the main elements in this system and explain their functionalities. We will conclude our report by giving brief overview of our journey in building this software project.

2 System Design

2.1 Overview

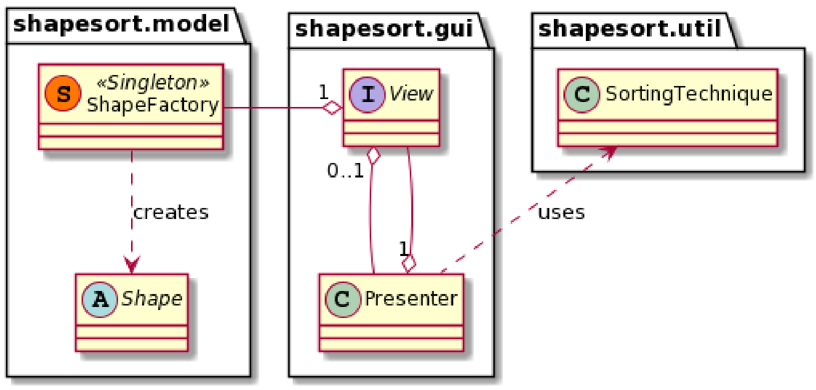

Figure 1: The system design `SOCCERGAME`'s (example placeholder image)

3 Front end

3.1 Overview

3.2 Class Diagram

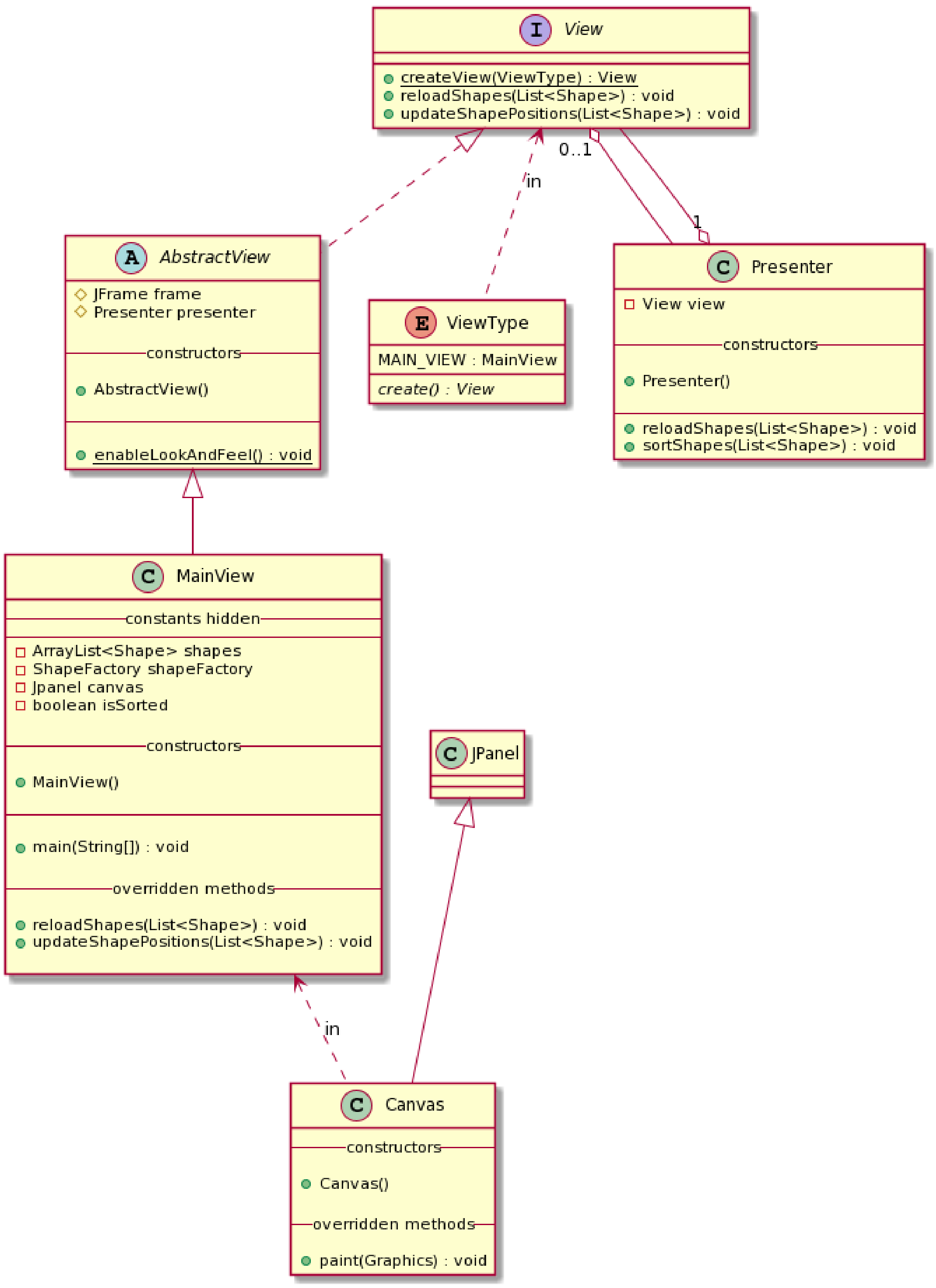

Figure 2: The class diagram of SOCCERGAME's front end model (example placeholder image)

3.3 Classes

list frontend classes here

4 Back end

4.1 Overview

4.2 Class Diagram

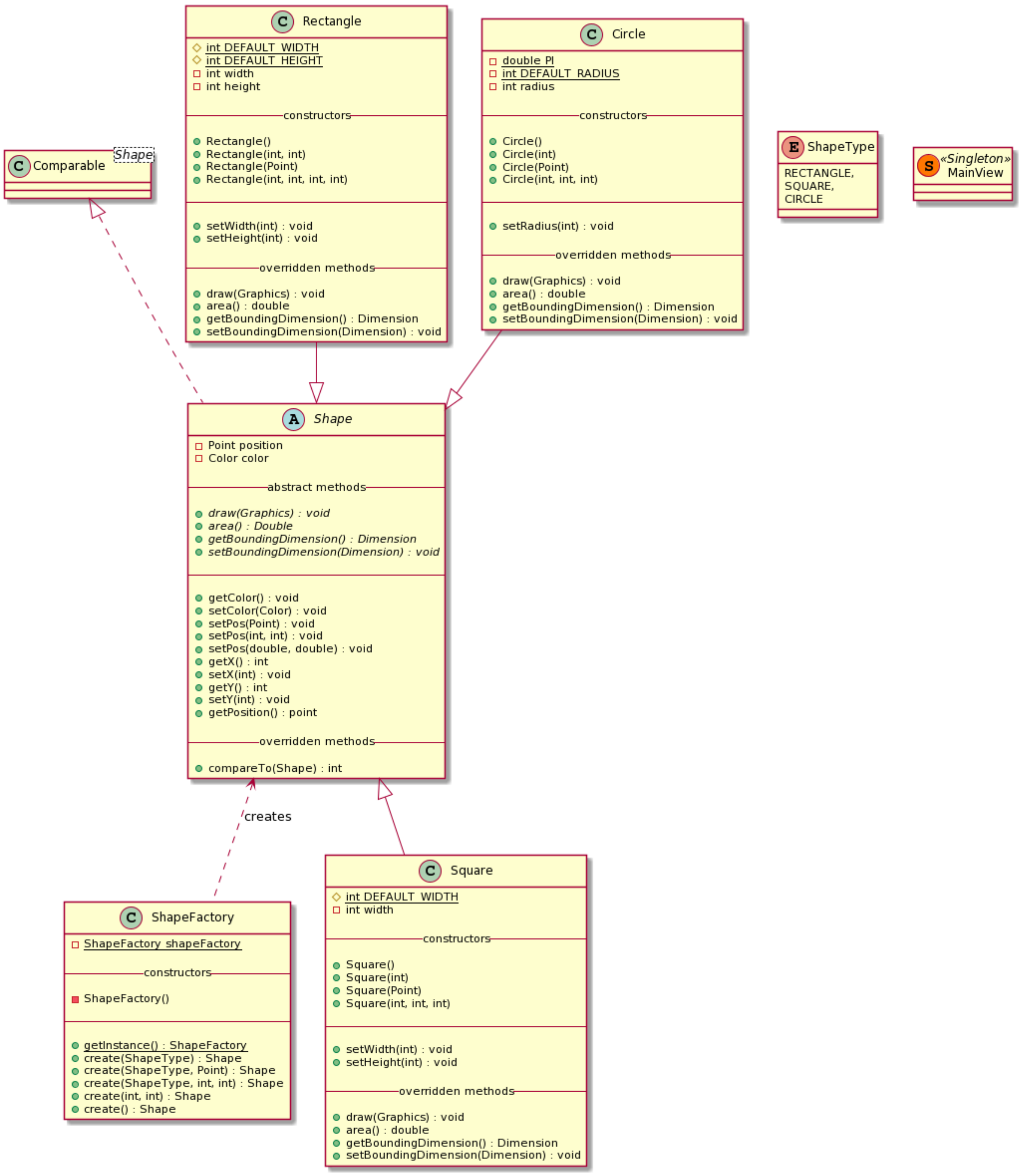

Figure 3: The class diagram of SOCCERGAME's player model (example placeholder image)

4.3 Classes

list backend classes here

5 Implementation

5.1 Class Implementation

disscuss class implementation

5.2 Execution

disscuss execution process using gradle

6 Tools

6.1 Source Code

IDE: IntelliJ IDEA 2021.2.1, to implement Java classes
Documentation: vscode, LaTeX, for documentation (dessign report, etc.)

6.2 Build

Build: Gradle 7.2 (to manage dependencies and build)
Java Version: openjdk version "16.0.1" 2021-04-20
Unit Testing: junit-jupiter-engine:5.8.1
Coverage: JaCoCo library

7 Conclusion

conclusion