COURSE: EECS 3311 F
SECTION B
TA: Alireza Naeiji
GROUP MEMBERS:

- Sanyam Panchal (Student id: 216855322)
- Krutarth Patel (Student id: 217190588)
- Dhruv Ukani (Student id: 216821902)
- Happy Patel (Student id: 216829038)

SOFTWARE PROJECT 2 REPORT

Part 1:

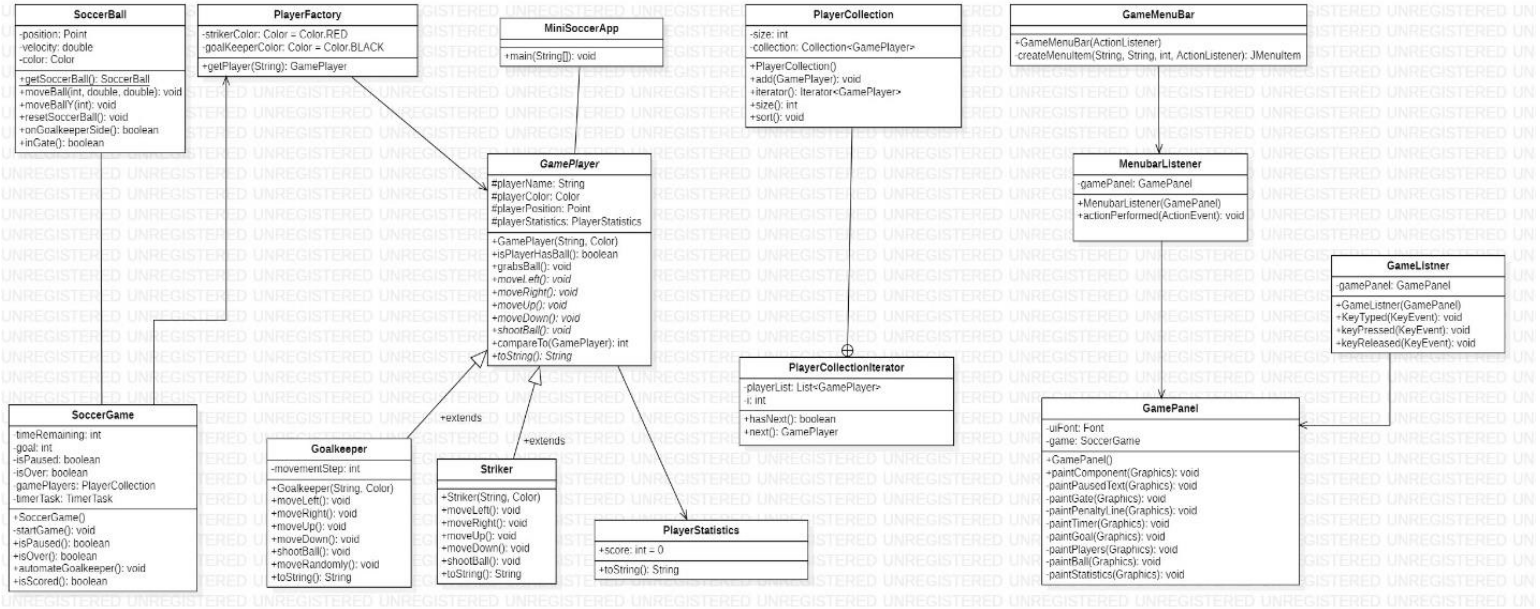
The goal of the software project 2 is to design a mini soccer ball game using OOD principles and design pattern. In this game there is a goalkeeper who randomly move in between gate (which works like goal post) and a striker who has ball and striker can move left, right, up and down and can kick the ball using arrow keys and space key. Moreover, there is menu bar on the top of window which have option of new game, exit, pause and resume.

One of the mistake we committed in this project was not planning how we will approach each individuals part in this project. As this was group project, we initially divided project in parts all four of us were working on different parts. And as we were not were on whole project, we were facing some problems as we were concentrating on project as individual and not as a whole project. But then we finished project by organising on zoom meeting which made our work a lot easier.

The OOD principles we used in this project are Inheritance, Abstraction, Encapsulation and Polymorphism along with Singleton Pattern. Here, Goalkeeper and striker are inherited from GamePlayer which is example of Inheritance. Here, private variable and methods are used in many classes (eg. SoccerBall, GameListener, etc.) which is example of encapsulation. And normally all the classes follow Polymorphism property. We have also used Singleton Pattern for Player Factory class. This helps us to ease out with resetting the player's properties, such as name, score statistics and more, when the game is restarted.

Report is structured as: The introduction of the project, Challenges that we as a group have faced while completing the project. UML diagram with brief details about classes and their functionality, Implementation of the MiniSoccerGame, Conclusion with the pros and cons of the Project.

Part 2: Design of the solution



- We have used the following design patterns. The first is Iterator pattern. PlayerCollection and PlayerCollectionIterator are example of iterator design pattern. Iterator is used to traverse through the collection of GamePlayers. The second design pattern is Factory design pattern. Here, goalkeeper and striker implements GamePlayer which is treated as an interface. We also have used Singleton Pattern for PlayerFactory class.

Part 3: Implementation of the solution

In this software project, the main method is included in Main.MiniSoccerApp class. In order to run the whole project, we only need to run MiniSoccerApp.java because it is the one with main method. If the file compiles and runs in Eclipse without any errors, we can see a MiniSoccer JFrame with Menu bar that controls the action of game such as new game, pause/resume game or exit game. The main method will generate two players, GoalKeeper

and Striker, will paint all the required component i.e. Goalkeeper, Initialize Striker with Soccer ball, Classify the gate and show the statistics of the game on the top left corner.

Description of the classes used:

GameListener: This class has one variable, one constructor and one method named keypressed. The main purpose of this method is the movement of the striker player and shooting the ball on the press of space key if the striker has the ball. But all this keypressed will work if game is not pause or over.

MenubarListener: The main purpose of this class is to control all the menu option available on menu bar which is on the top of Frame. So this class control pause (if game is not pause) and resume (if game is pause) of the game. The function of new game and exit the game is also mentioned in this class.

MiniSoccerApp: This class contain the main method. So, you need to run this class in order to start the game.

SoccerBall: All the attributes and function related to the ball is mentioned in this class. The color of the ball is mentioned in this class. Moreover, velocity of the ball is manipulated in this class. Furthermore, direction of the ball when striker shoot the ball and then goalkeeper shoot it back is controlled in this class. Moreover, position of the ball which get reset after goal is due to this class.

SoccerGame: Time countdown and goal count are maintained by this class. The game starts with 60 seconds and when timer become 0 then the isOver attribute is set to true and game get over. And when the goal is scored, game became pause and goal is incremented by 1 and striker is set to its initial position with the ball. Moreover, the movement of goalkeeper is mentioned in this class. If goalkeeper grabs the ball then he shoots the ball back otherwise goalkeeper keeps moving randomly in between the gate.

GamePlayer: It is abstract class used for new player such as GoalKeeper and Striker. It defines all the properties of players.

Goalkeeper: This class defines how much players moves when user press up, down, left and right button. Moreover, this method defines how goalkeeper moves

randomly. It also show chances of goalkeeper moving left or right as it moves randomly.

PlayerFactory: This class is a factory class for the game and is Singleton class.

Striker: This class defines how much striker make movement on the press of the key and how the ball moves when striker shoots the ball.

GameMenuBar: The menu bar which we can see on the top of the window in the game is defined in this class.

To Test this project with Junit to have a Jacoco of roughly 80%, we created 7 Test files that are briefly explained below:

In the package model.players,

TestGamePlayers.java contains Junit test methods to cover all possible test cases for GamePlayer.java, Goalkeeper.java, and Striker.java.

TestPlayerCollection.java contains Junit test methods to cover all possible test cases for the class PlayerCollection.java and PlayerCollectionIterator.java.

TestPlayerStatistics.java contains Junit test methods to cover all possible test cases for the class PlayerStatistics.java.

TestPlayerFactory.java contains Junit test methods to cover all possible test cases for the class PlayerFactory.java.

Similarly, In the package model,

We created **SoccerBallTest.java** and **SoccerGameTest.java** to test methods of the classes SoccerBall.java and SoccerGame.java, respectively.

Lastly, In the package view,

GameTest.java contains Junit test methods to cover all possible test cases for the class GamePanel.java.

The tools which I have used are: Eclipse, StarUML, MS Word. We have used Eclipse Version: 2020-12 (4.18.0). JDK version is 14.0.2. Junit5.

Part 4: Conclusion

- Well in the second project, most of the code was already provided and we were need to add few classes in the project. So, there were less coding compared to first project which was good.
- We were facing trouble running the main class and connecting all the classes together in such a way to obey the principles of OOD.
- In this project, we have learned how to make JUnit test cases alongwith MVC model.
- Advantages of group project is to mange the collaboration and learn the team work. Disadvantages is to manage the team, especially during this pandemic situation.
- Recommendations:

    More smaller labs that can help to understand the concepts before making the projects.
    More practice by reading books and understanding the core concepts.
    Collaborate with others to ease out the process of understanding.