README

Group No. - 121
Student 1 - Akshit Gupta (2022058)
Student 2 - Manit Kaushik (2022277)

STEPS TO FOLLOW TO RUN THE CODE USING MAVEN:

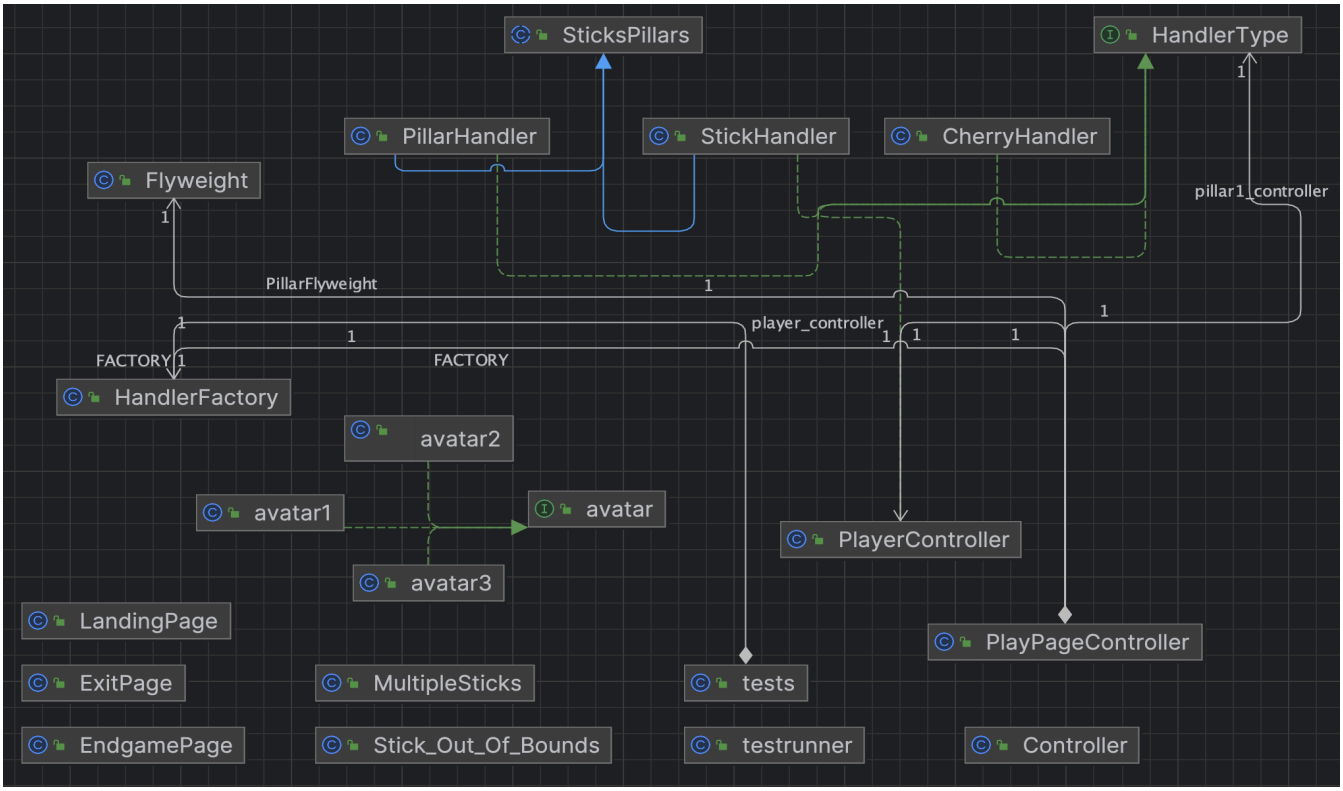
To run the code properly using maven, enter the following on the Terminal in the folder named Project -

1. mvn clean javafx:run

Code Implementation:

For this project, we created a StickHero Game using JavaFX. Our game offers players a thrilling and interactive experience as the player guides the stickman through various obstacles using sticks and exciting surprises await him on scoring. All the features which were required to be implemented as per the given instructions have been included in our project as well.

UML:



OOPS Implementation:

- **Avatar.java interface**: Avatar1.java, Avatar2.java and Avatar3.java implement this interface.
- **Abstract classes**: PillarHandler.java and StickHandler.java both extend the class SticksPillars.java which is an abstract class.
- **Objects** of CherryHandler.java, PillarHandler.java, PlayerController.java, StickHandler.java were used in the PlayPageController.java class.
- **HandlerType.java interface**: PillarHandler.java, StickHandler.java, PlayerController.java and CherryHandler.java all implement this interface.
- **Polymorphism**: Can be seen in the usage of Factory design Pattern, where we can use a general handler type, and then use it to convert to particular handler types when they are dispensed by the factory. This part of code can be seen in PlayPageContoller.java

Design Patterns Implementation:

- **Factory**: Helps us in getting different handlers for different objects. This has been achieved using the **HandlerFactory.java** class.
- **Flyweight**: This design pattern was used for creating unique pillars of different dimensions for each level. This functionality has been achieved using the **Flyweight.java** Class.

JUnit Implementation:

- **tests.java** class includes 3 tests which checks our Factory Design Pattern and ensures we get an object of the correct handler we want. The third test ensures that the image obtained for changing the character is not null.
- A **testrunner.java** class has also been created to run the tests and check them.

Bonus:

- While playing the game, when a player crosses 500 points, their character upgrades to **Mario** which allows a player to score 2x the points.
- Crossing 2000 points upgrades the character to **Sonic** which allows a player to score 3x the points.
- Getting out would again lead a player to the original **StickMan** character.
- These features introduce a very exciting feature to play the game in a more fun manner.
- **MultiThreads:** Controller.java class uses threads. A thread has been used there to transition to the **play page** while another thread has been created to **play background music** infinitely.

Exception Handling:

- **Sticks_Out_Of_Bounds.java** class has another instance of exception handling. The exception would also result in the player dying if the stick they extend goes out of bounds of the screen.
- **MultipleSticks.java** class has one instance of exception handling. This class prevents any player from using multiple sticks together which would be considered as cheating in game and the player would die due to this.

Github Repository Link - https://github.com/ManitK/CSE201_Project