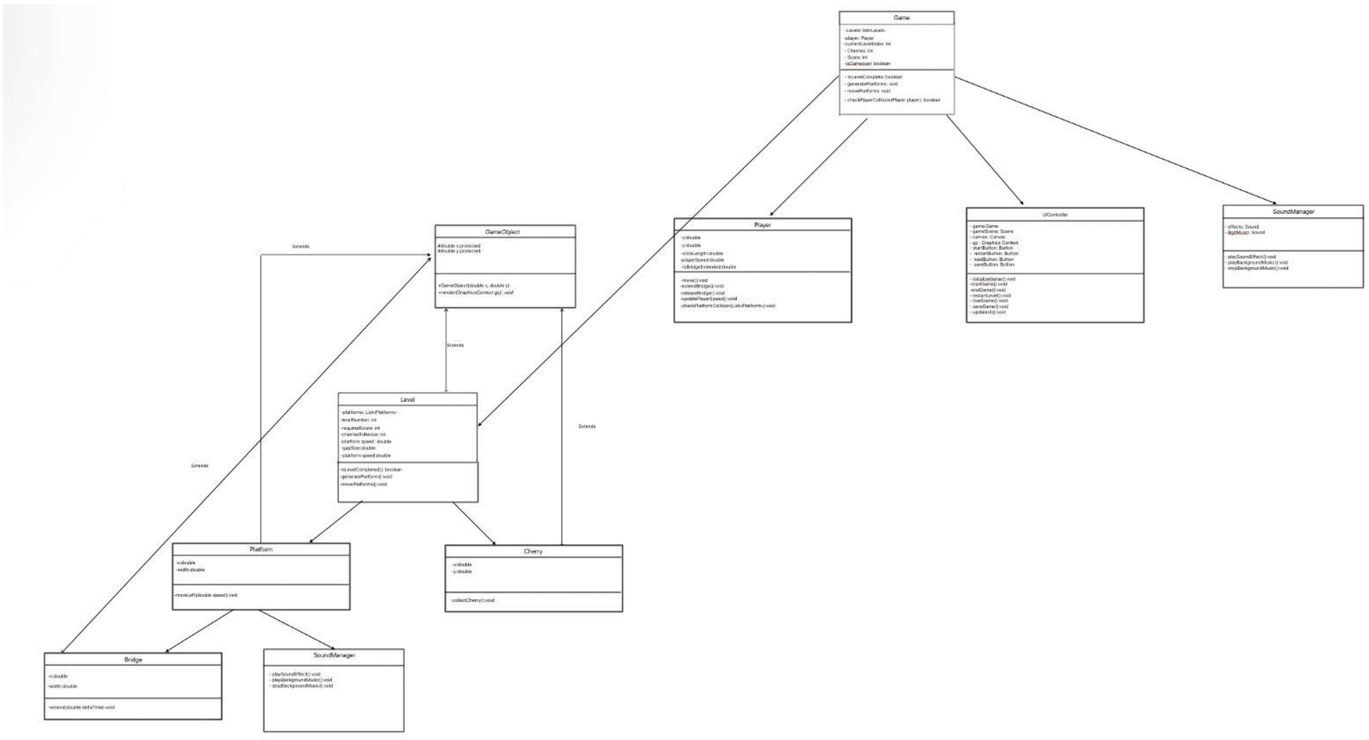
Project: Deadline - I

Group 111

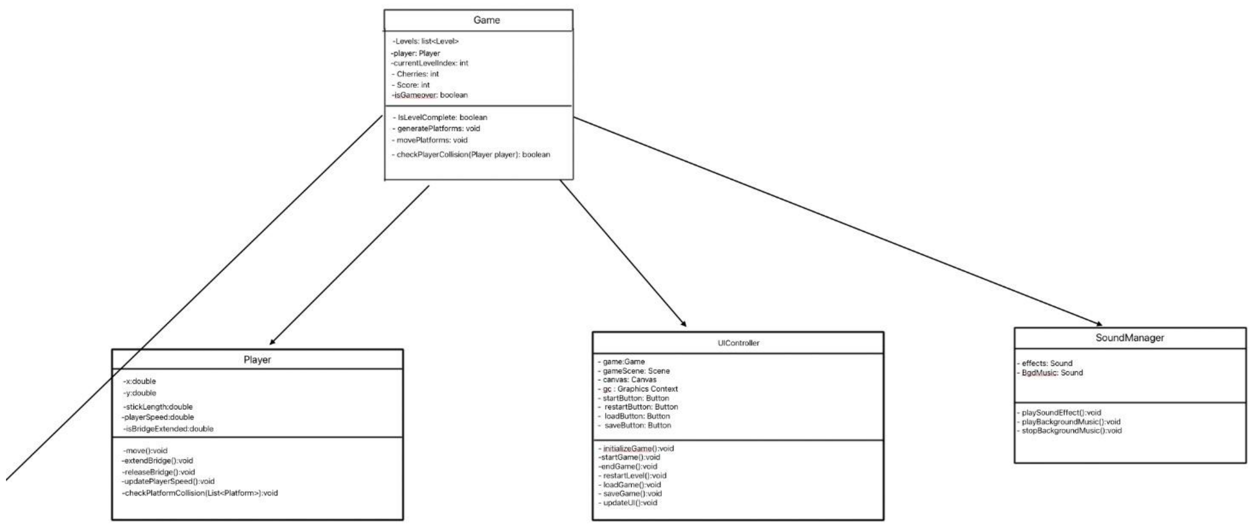
Nishant Singh 2022328
Shashank Gadamsetty 2022469

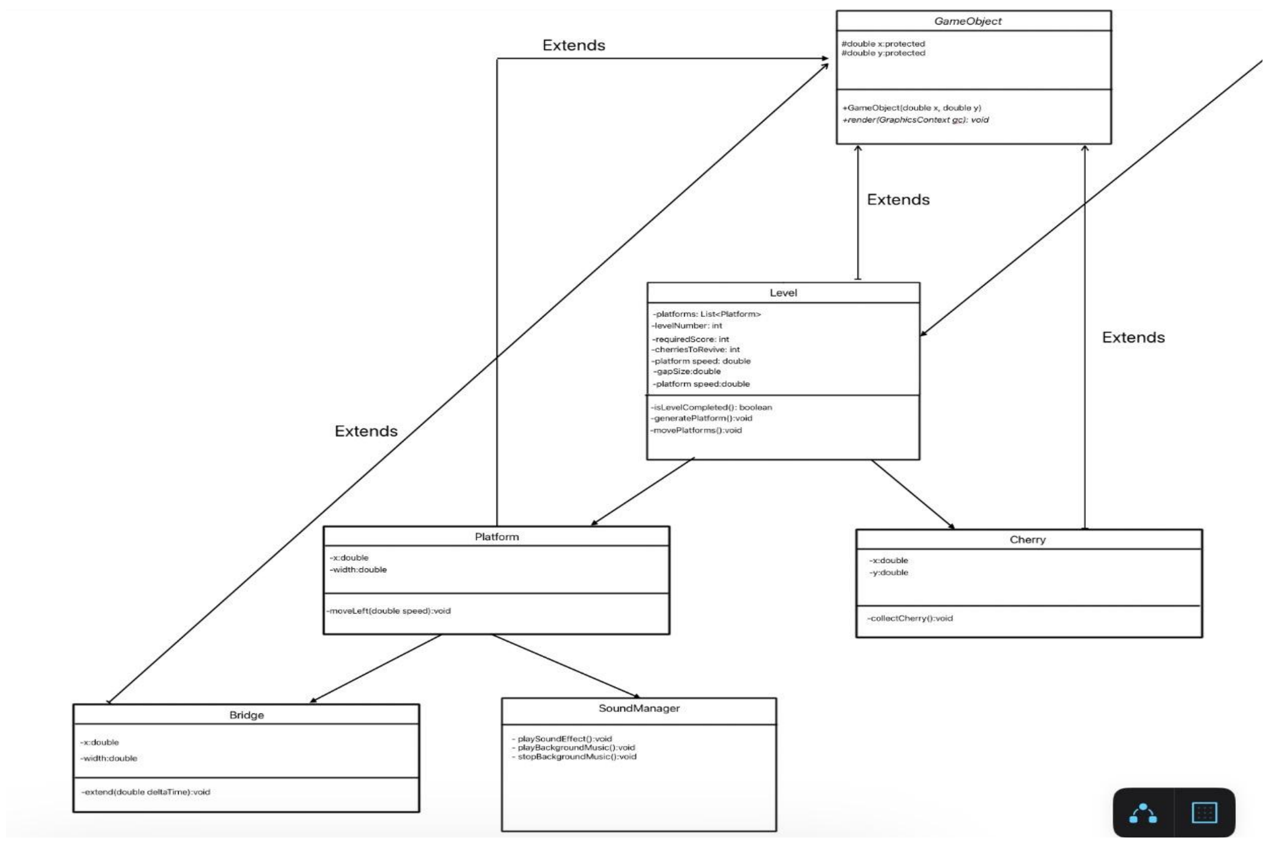
UML Diagram:

1.) Full View (A bit blurry but for full context)

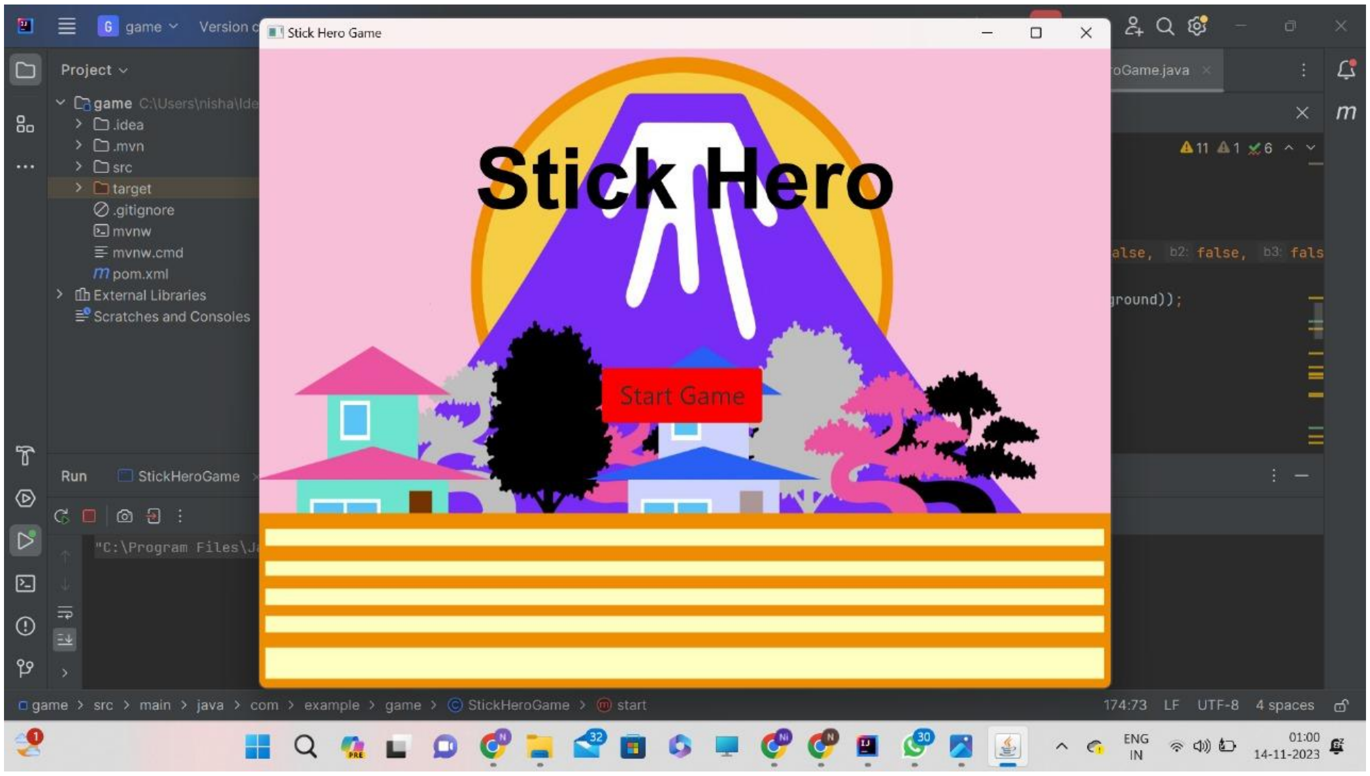


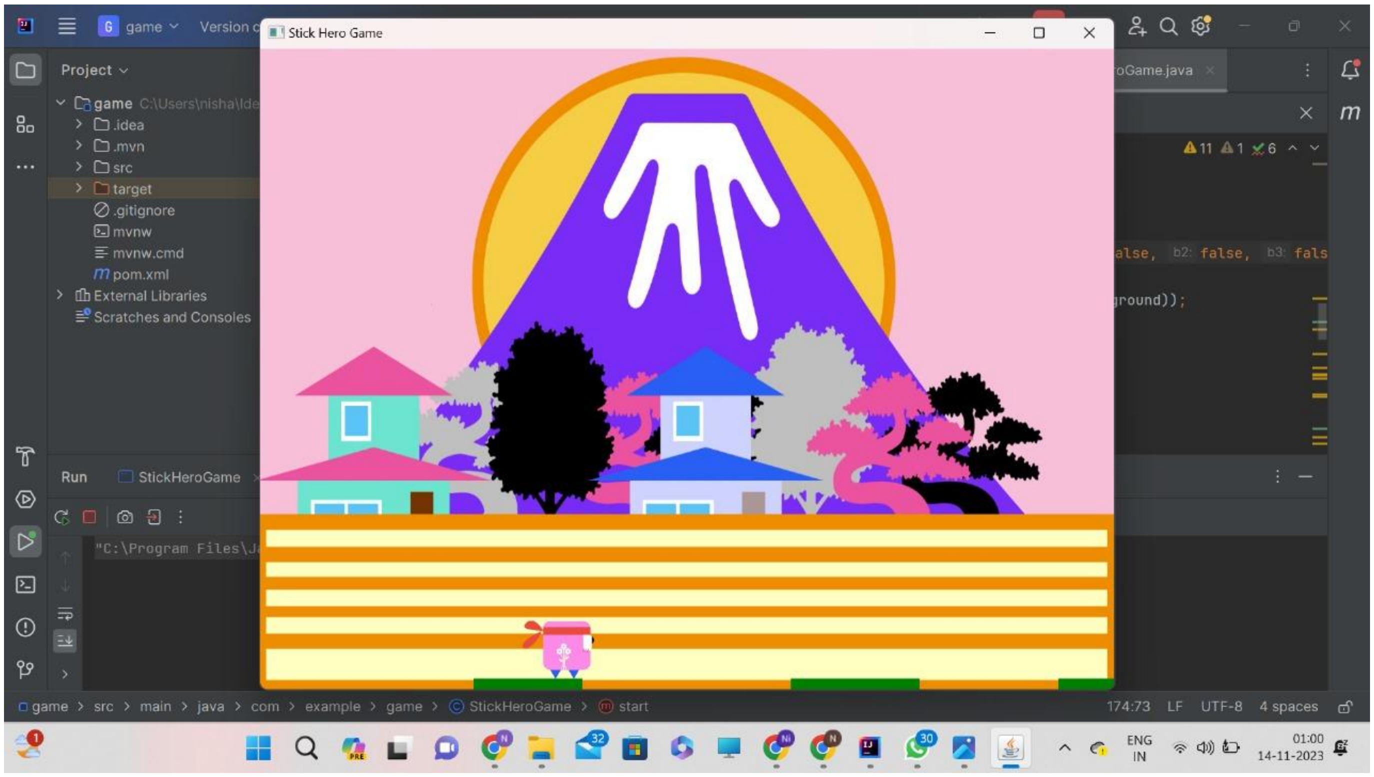
2.) Half-Half View (More Clarity for reading content)


Game
-Levels: list<Level>
-player: Player
-currentLevelIndex: int
- Cherries: int
- Score: int
-isGameover: boolean
- IsLevelComplete: boolean
- generatePlatforms: void
- movePlatforms: void
- checkPlayerCollision(Player player): boolean
Player
-x:double
-y:double
-stickLength:double
-playerSpeed:double
-isBridgeExtended:double
-move():void
-extendBridge():void
-releaseBridge():void
-updatePlayerSpeed():void
-checkPlatformCollision(List<Platform>):void
UIController
- game:Game
- gameScene: Scene
- canvas: Canvas
- gc : Graphics Context
- startButton: Button
- restartButton: Button
- loadButton: Button
- saveButton: Button
- initializeGame():void
-startGame():void
-endGame():void
- restartLevel():void
- loadGame():void
- saveGame():void
- updateUI():void
SoundManager
- effects: Sound
- BgdMusic: Sound
- playSoundEffect():void
- playBackgroundMusic():void
- stopBackgroundMusic():void

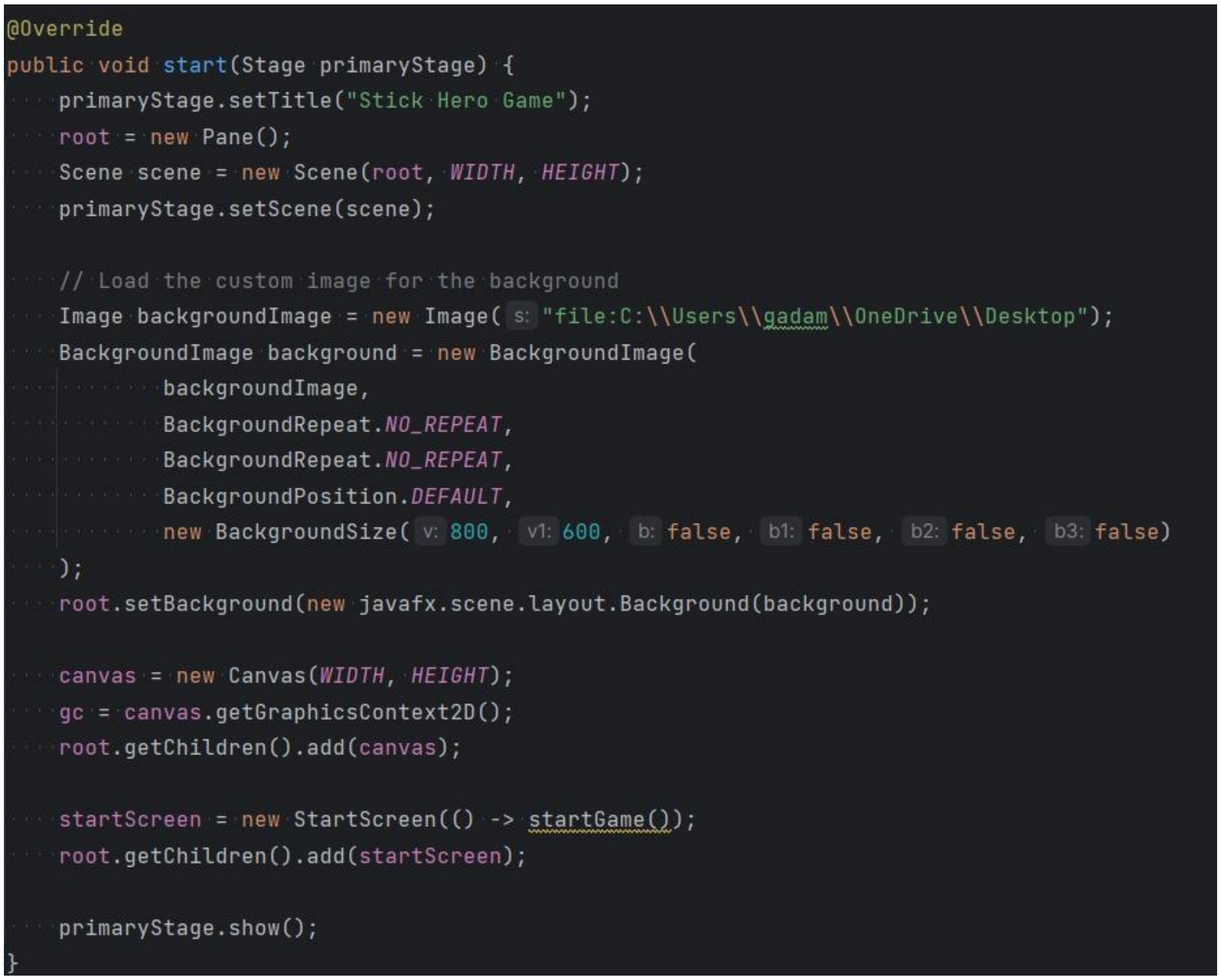

Extends
GameObject
#double x:protected
#double y:protected
+GameObject(double x, double y)
+render(GraphicsContext gc): void
Extends
Level
-platforms: List<Platform>
-levelNumber: int
-requiredScore: int
-cherriesToRevive: int
-platform speed: double
-gapSize:double
-platform speed:double
-isLevelCompleted(): boolean
-generatePlatform():void
-movePlatforms():void
Extends
Extends
Platform
-x:double
-width:double
-moveLeft(double speed):void
Cherry
-x:double
-y:double
-collectCherry():void
Bridge
-x:double
-width:double
-extend(double deltaTime):void
SoundManager
- playSoundEffect():void
- playBackgroundMusic():void
- stopBackgroundMusic():void

Skeleton Screen :




Stick Hero Game
Project
Run
StickHeroGame
game > src > main > java > com > example > game > StickHeroGame > start
174:73 LF UTF-8 4 spaces

Section for UI :



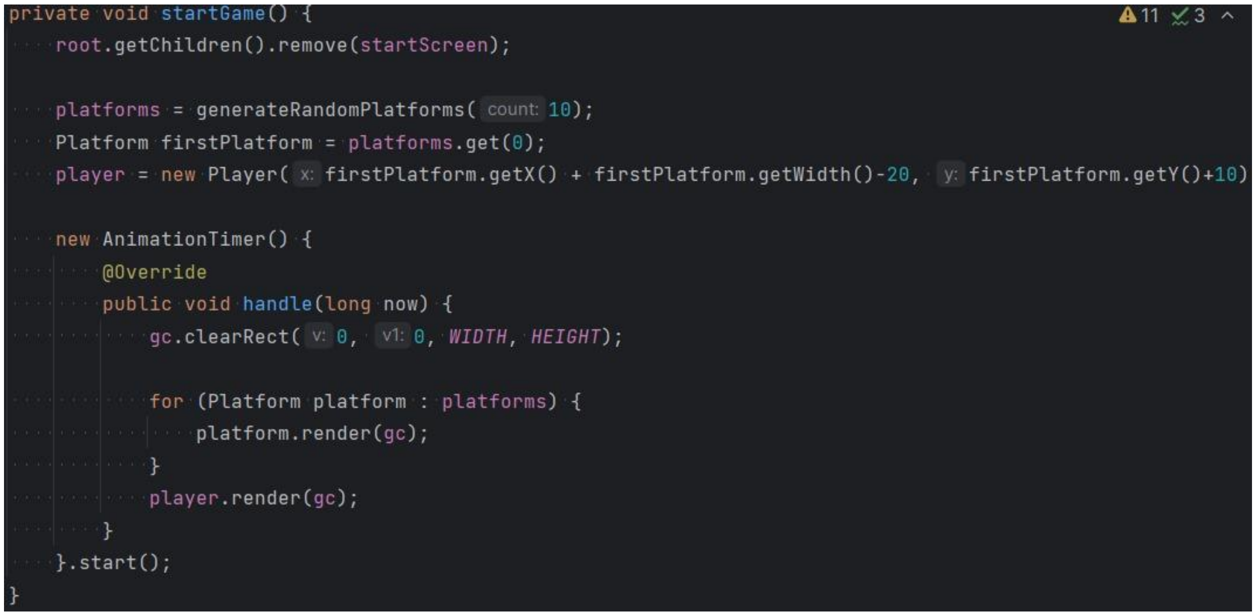
```java
@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle("Stick Hero Game");
    root = new Pane();
    Scene scene = new Scene(root, WIDTH, HEIGHT);
    primaryStage.setScene(scene);

    // Load the custom image for the background
    Image backgroundImage = new Image( s: "file:C:\\Users\\gadam\\OneDrive\\Desktop");
    BackgroundImage background = new BackgroundImage(
            backgroundImage,
            BackgroundRepeat.NO_REPEAT,
            BackgroundRepeat.NO_REPEAT,
            BackgroundPosition.DEFAULT,
            new BackgroundSize( v: 800,  v1: 600,  b: false,  b1: false,  b2: false,  b3: false)
    );
    root.setBackground(new javafx.scene.layout.Background(background));

    canvas = new Canvas(WIDTH, HEIGHT);
    gc = canvas.getGraphicsContext2D();
    root.getChildren().add(canvas);

    startScreen = new StartScreen(() -> startGame());
    root.getChildren().add(startScreen);

    primaryStage.show();
}
```

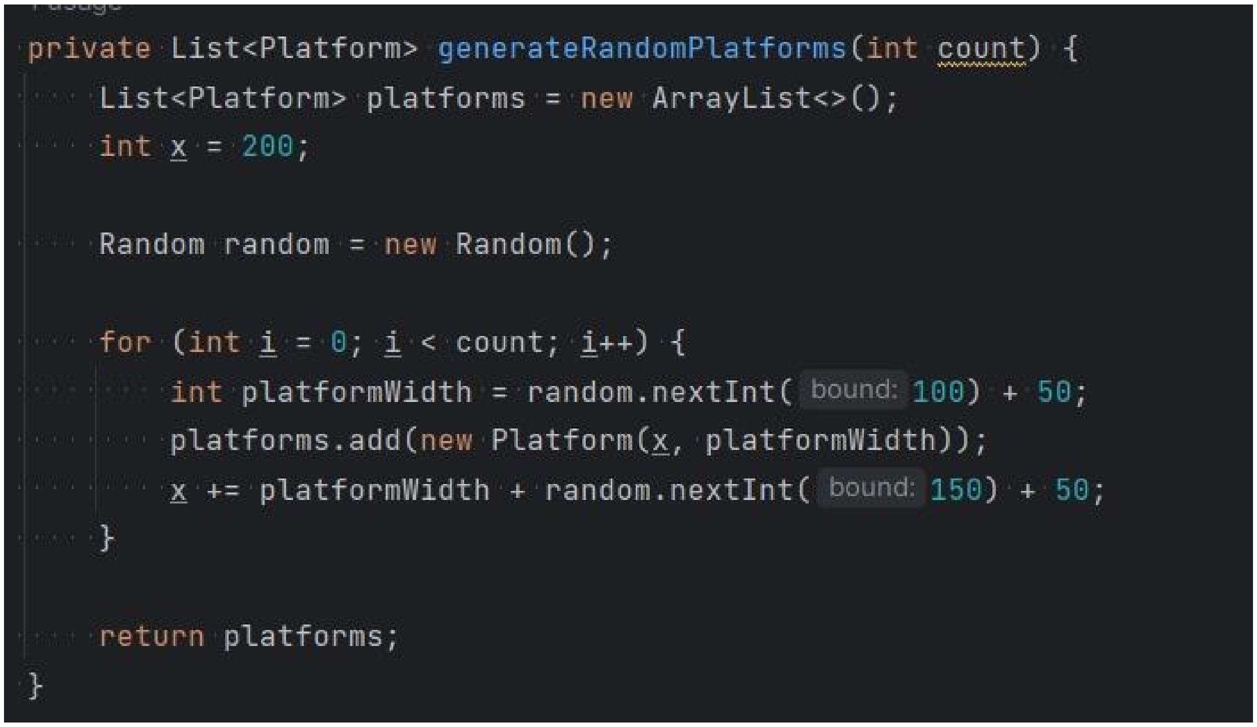


```java
private void startGame() {
    root.getChildren().remove(startScreen);

    platforms = generateRandomPlatforms( count: 10);
    Platform firstPlatform = platforms.get(0);
    player = new Player( x: firstPlatform.getX() + firstPlatform.getWidth()-20,  y: firstPlatform.getY()+10)

    new AnimationTimer() {
        @Override
        public void handle(long now) {
            gc.clearRect( v: 0,  v1: 0, WIDTH, HEIGHT);

            for (Platform platform : platforms) {
                platform.render(gc);
            }
            player.render(gc);
        }
    }.start();
}
```



```java
private List<Platform> generateRandomPlatforms(int count) {
    List<Platform> platforms = new ArrayList<>();
    int x = 200;

    Random random = new Random();

    for (int i = 0; i < count; i++) {
        int platformWidth = random.nextInt( bound: 100) + 50;
        platforms.add(new Platform(x, platformWidth));
        x += platformWidth + random.nextInt( bound: 150) + 50;
    }

    return platforms;
}
```

Hierarchy :

```
                     Hierarchy and Relationship

                              Game
                                |
         _______________________|_______________
         |              |       |               |
       Level          Player  UIController   SoundManager
         |
  _______|______
  |             |
Platform      Cherry
|      \
Bridge   \
      SoundEffect
```

- Game class contains the game logic and manages levels, player, UI, and sound.

- Level class represents a game level and contains platforms and related properties.
- Player class represents the player character and manages their position, movement, and bridge extension.
- Platform class represents the platforms that the player must traverse.
- Bridge class represents the stick bridge that the player extends.
- Cherry class represents the cherries that the player can collect for points.
- The UIController class handles the game's user interface, including buttons and canvas for rendering.
- The SoundManager class handles sound effects and background music in the game
- SoundEffect represents individual sound effects used in the game.