AP PROJECT README

Group Members – Sahil Gupta , Vashu.

Roll Numbers – 2022430 , 2022606

CONTROLS OF THE GAME

Mouse click – a pressed mouse click would increase the length of the stick continuously till the mouse is unpressed .

Spacebar – Use spacebar to flip the stickman to collect the cherries .

How to run the program :

Note – The system running the game should have maven dependency for junit .

1)Please run the StickHeroApplication file to start the program and follow the controls of the game to play the game .

OOPS Concepts Used

1) Inheritance –
   Inheritance is used in the Gameovercontroller and StickHeroController classes .

```
vashu
public class GameOverController extends StickHeroController{
    2 usages
    @FXML
    public Label gameOverScoreLabel;
    2 usages
```

2) Polymorphism –
   Polymorphism is used in overriding the object class and the thread methods and using the interface in the different classes .
   Some examples are :

```
vashu
@Override
public void run() {
    if(Thread.currentThread().getName().equals("move")){
        transitions();
    }
}
```

```
no usages  vashu
@Override
public void run() { System.out.println("Hero is Running"); }

sahilguptasg2017
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;

    Hero hero = (Hero) o;

    return Objects.equals(heroImage, hero.heroImage);
}

sahilguptasg2017
@Override
public int hashCode() {
    return heroImage != null ? heroImage.hashCode() : 0;
}
no usages
private int hero_score ;
```

Interfaces :

We have used 2-3 interfaces in the

```java
package StickManHero;

import ...

2 implementations  sahilguptasg2017
public interface Controller {

    1 usage  1 implementation  sahilguptasg2017
    public void showExitConfirmationDialog() ;
    1 usage  1 implementation  sahilguptasg2017
    public void onStartButtonClick() throws IOException;

}
```

```java
package StickManHero;

import javafx.scene.image.ImageView;

1 implementation  vashu
public interface MainHero {
    1 usage  1 implementation  vashu
    ImageView getImageView();
    no usages  1 implementation  vashu
    void makeStick();
    no usages  1 implementation  vashu
    void run();
}
```

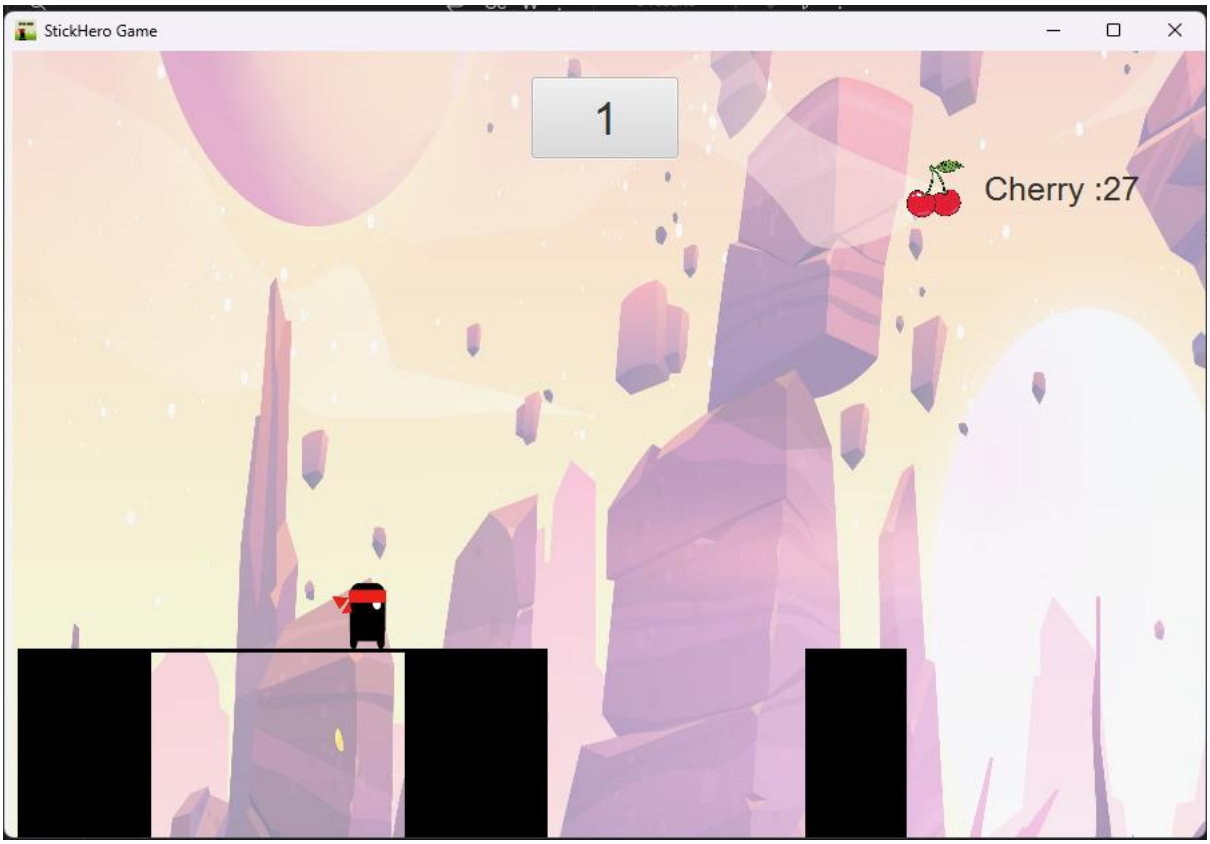
Polymorphic variable :

```java
        stage.show();

        // Polymorphism is used here
        ((StickHeroController) controller).anchorPane.requestFocus();
    });
    // Start the combined fade-out and fade-in transition
```

At line number 608.

THE IMPLEMENTATION OF THE GAME

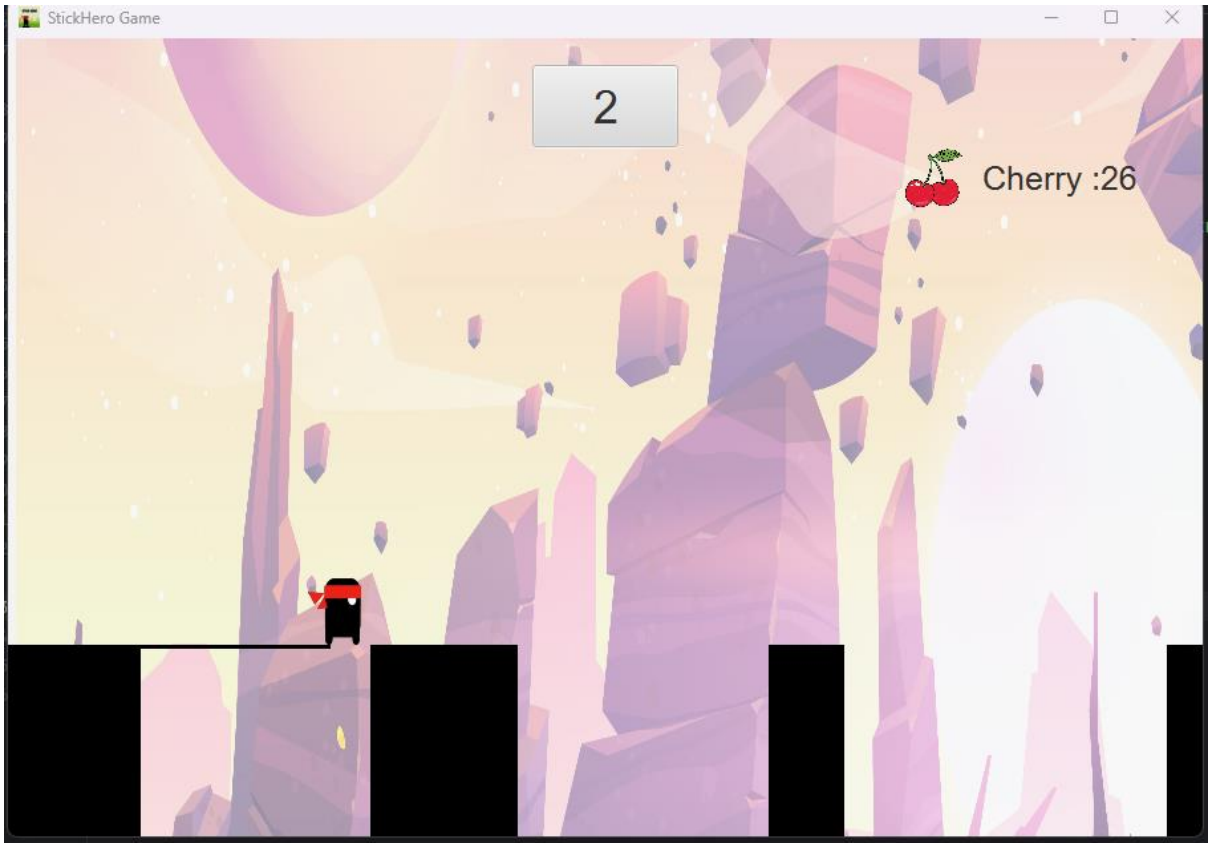
1. Our Game is allowing players to control the character stick hero and it can move between platforms . A image is being added to see how the game is working .



2. Our game include multiple pillars of different widths as we can see from the figure given below :

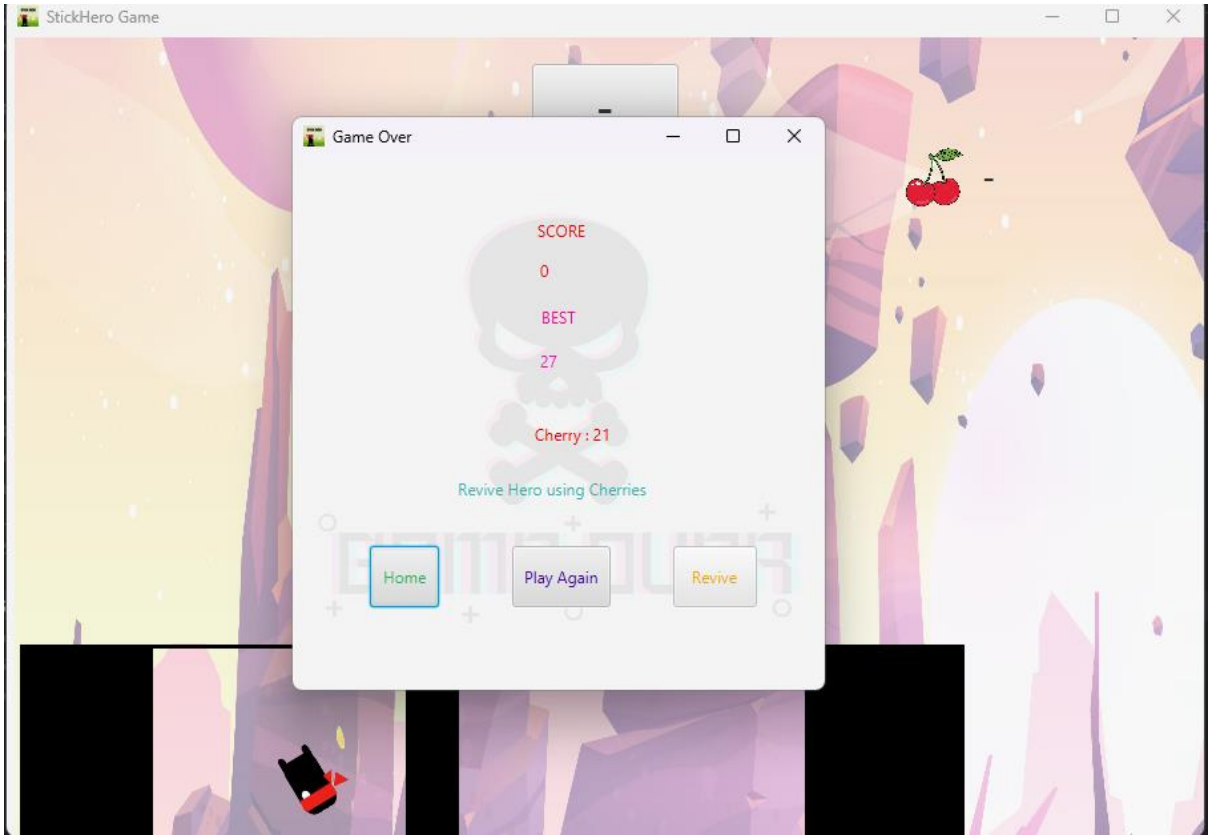
3. Our game is using the reviving feature in which the players can be revived if they have cherry at least equal to 5 . After the player is revived the 5 cherries are deducted from the total cherry count .

   Before Revival :

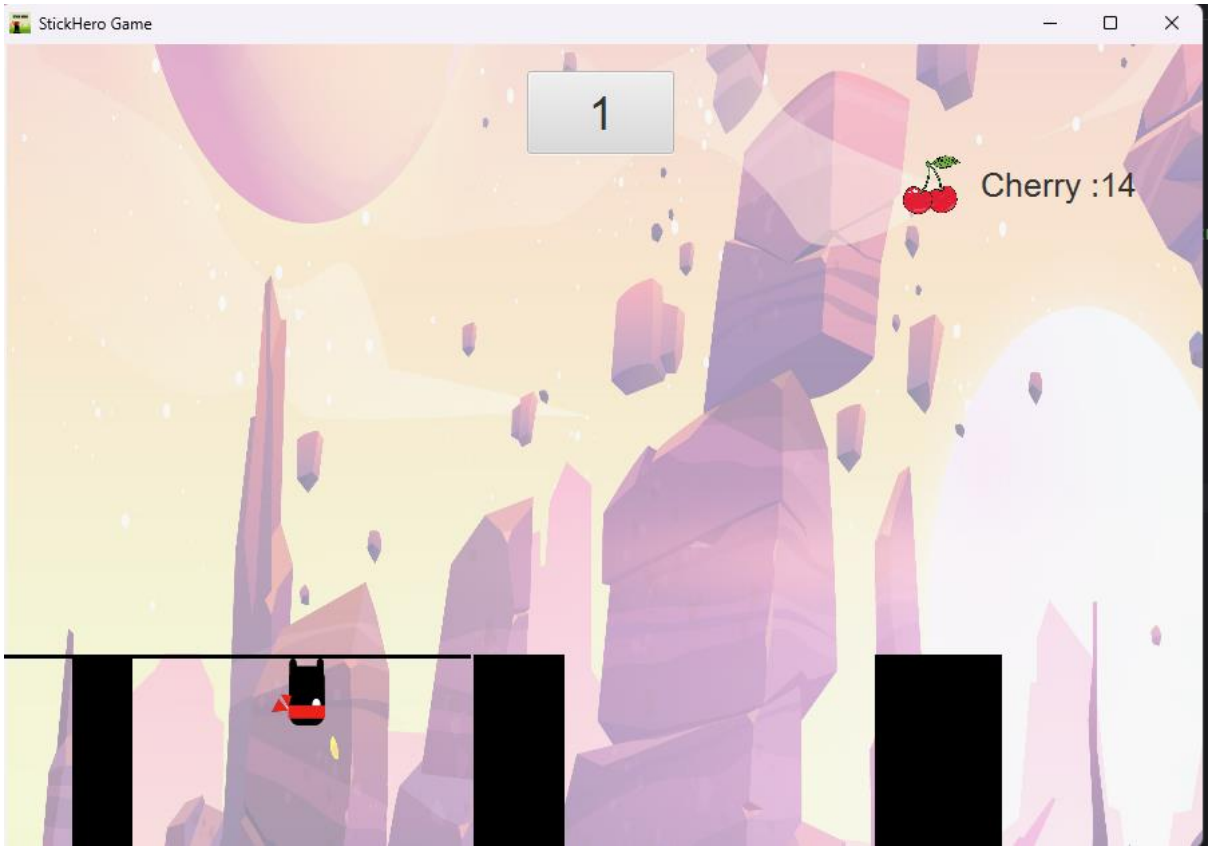


   After Revival :

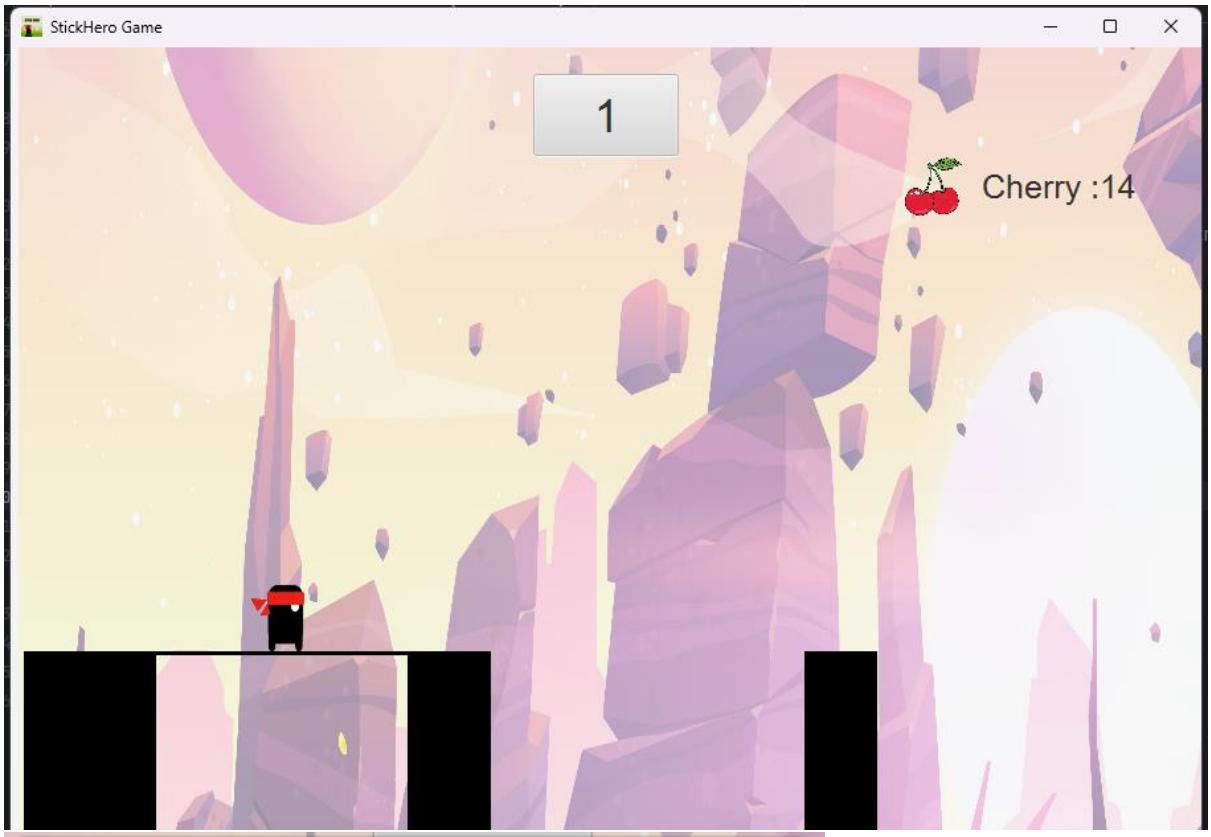
4. Our game is making sure that the length of the stick is such that it lands on the next platform , otherwise the stickman falls in the abyss .

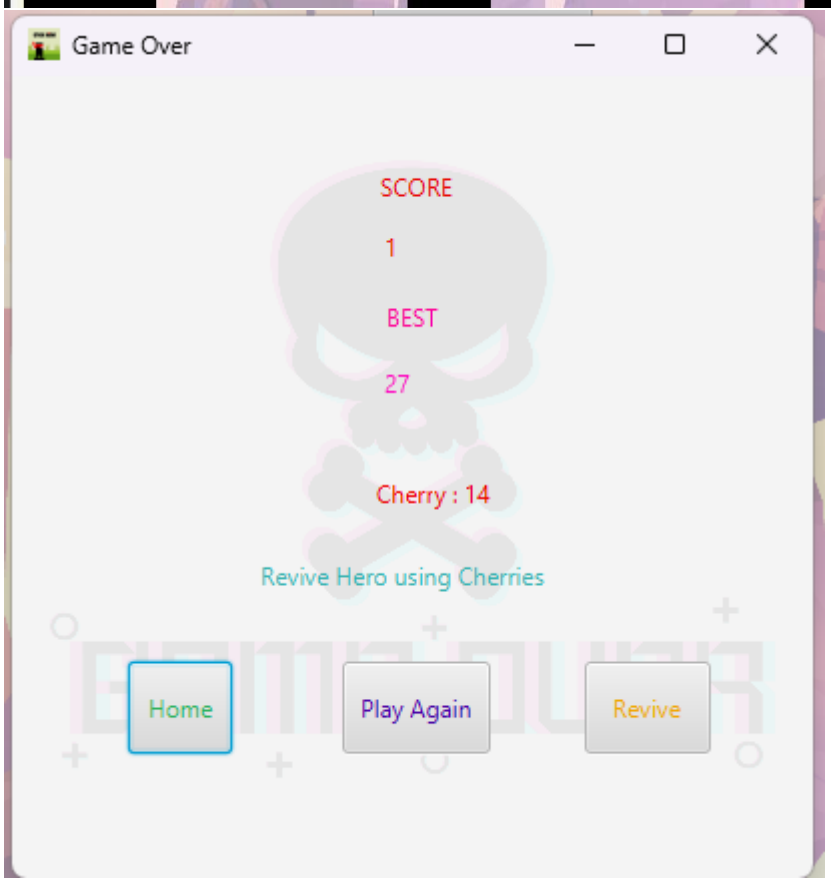


5. Our game is making sure that the hero is able to collect cherry ‘ s similar to that like in the original game . This is also added to the player’s score .



6. In our game we have the scoring system above .





Last score , highest score and number of cherries .

7. We have added the animations and sound to increase the overall gaming experience .

The Design Pattern used are the following :

1) **Singleton Design Pattern :**
   We have used the singleton design in the Hero class in which we have only made 1 instance of the hero in the whole game .
   The code for it is :

```
        private Hero() {
    // Load the image during the first instantiation
    heroImage = new Image("hero_style1.png");
}

// Public method to get the single instance of Hero
public static Hero getInstance() {
    if (instance == null) {
        instance = new Hero();
    }
    return instance;
}

// Public method to get the ImageView for the Hero
public ImageView getImageView() {
    return new ImageView(heroImage);
}

@Override
public void makeStick() {
    System.out.println("Hero is Making stick");
}

@Override
public void run() {
    System.out.println("Hero is Running");
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;

    Hero hero = (Hero) o;

    return Objects.equals(heroImage, hero.heroImage);
}
```

2) **Composite Design Pattern**

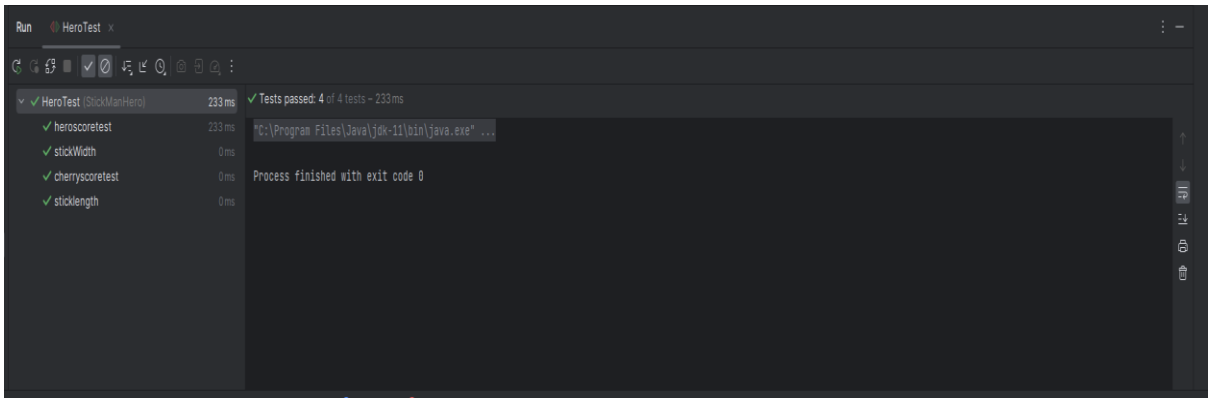
For the second Design Pattern we have used Composite Design Pattern in which we are composing the objects in a tree like structure.

```
public void onStartButtonClick(ActionEvent event) throws IOException {
        Scanner in = null;
        try{
            in = new Scanner(new BufferedReader(new FileReader("AP-
Project\\src\\main\\java\\StickManHero\\GameState.txt")));
            if (in.hasNext()){
//                System.out.println("File read");
                highScore = Integer.parseInt(in.next());
                cherryScore = Integer.parseInt(in.next());
            }
        }catch(IOException e){
            System.out.println(e.getMessage());
            e.printStackTrace();
        }finally {
            if (in!=null) in.close();
        }


        String path = "AP-
Project\\src\\main\\java\\StickManHero\\sound_1.mp3";

        // Instantiating Media class
        media = new Media(new File(path).toURI().toString());

        // Instantiating MediaPlayer class
        mediaPlayer = new MediaPlayer(media);

        // by setting this property to true, the audio will be played
        mediaPlayer.setAutoPlay(true);

        mediaPlayer.setCycleCount(MediaPlayer.INDEFINITE);

        // Load the new scene
        FXMLLoader loader = new
FXMLLoader(Objects.requireNonNull(getClass().getResource("Scene-1.fxml")));
//        newSceneRoot =
FXMLLoader.load(Objects.requireNonNull(getClass().getResource("Scene-
1.fxml")));
        newSceneRoot = loader.load();
        // set-up controller for scene-1
        controller = (StickHeroController) controller;
        controller = loader.getController();


        // Get the current stage
        Stage stage = (Stage) ((Node)
event.getSource()).getScene().getWindow();

        // Create a new scene for fade-out transition
        Scene oldScene = stage.getScene();

        // Set up a fade-out transition for the old scene
        FadeTransition fadeOutTransition = new
FadeTransition(Duration.millis(500), oldScene.getRoot());
        fadeOutTransition.setFromValue(1.0);
        fadeOutTransition.setToValue(0.0);

        // Set up a fade-in transition for the new scene
```

```
        FadeTransition fadeInTransition = new
FadeTransition(Duration.millis(500), newSceneRoot);
        fadeInTransition.setFromValue(0.0);
        fadeInTransition.setToValue(1.0);

        // Combine fade-out and fade-in transitions
        SequentialTransition sequentialTransition = new
SequentialTransition(fadeOutTransition, fadeInTransition);

        // Set the new scene with a fade-in transition
        sequentialTransition.setOnFinished(e -> {
            stage.setScene(new Scene(newSceneRoot));
            stage.show();

            // Polymorphism is used here
            ((StickHeroController) controller).anchorPane.requestFocus();
        });
        // Start the combined fade-out and fade-in transition
        sequentialTransition.play();
        game_maker();
//        myCherry.setText("" + cherryScore);

        // Set separate event handlers for mouse pressed and released

    }
```

JUNIT TESTING

WE have done junit testing with 3 tests . The output of junit testing is as follows :



Bonus Marks :

We have also used multithreading in moving hero and the towers . (moveAll thread)

```java
StickHeroController myRunnable = new StickHeroController();

Thread moveAll = new Thread(myRunnable, name: "move");

if (x2 > x1+w1+(l-3) || x2 + w2 < x1+w1+(l-3)){
```

```java
    }
}else{
    onTower = 1;

    // moveAll is a thread which calls transitions
    moveAll.start();
    try{
        moveAll.join();
    }catch(InterruptedException e){
        e.printStackTrace();
    }
    Platform.runLater(() ->{
        PauseTransition pause = new PauseTransition(Duration.millis( ms: 300));
```

Note -

Please run the game to see the animations and the sound effects .