Object Oriented Programming

Semester Project Report

A 2D Java Game

Group Members

Muhammad Saad Hussaini
Takreem Masood
Muhammad Ubaid Ullah

CMS IDs

211190
208720
211321

The BooMBaby Team

LOST

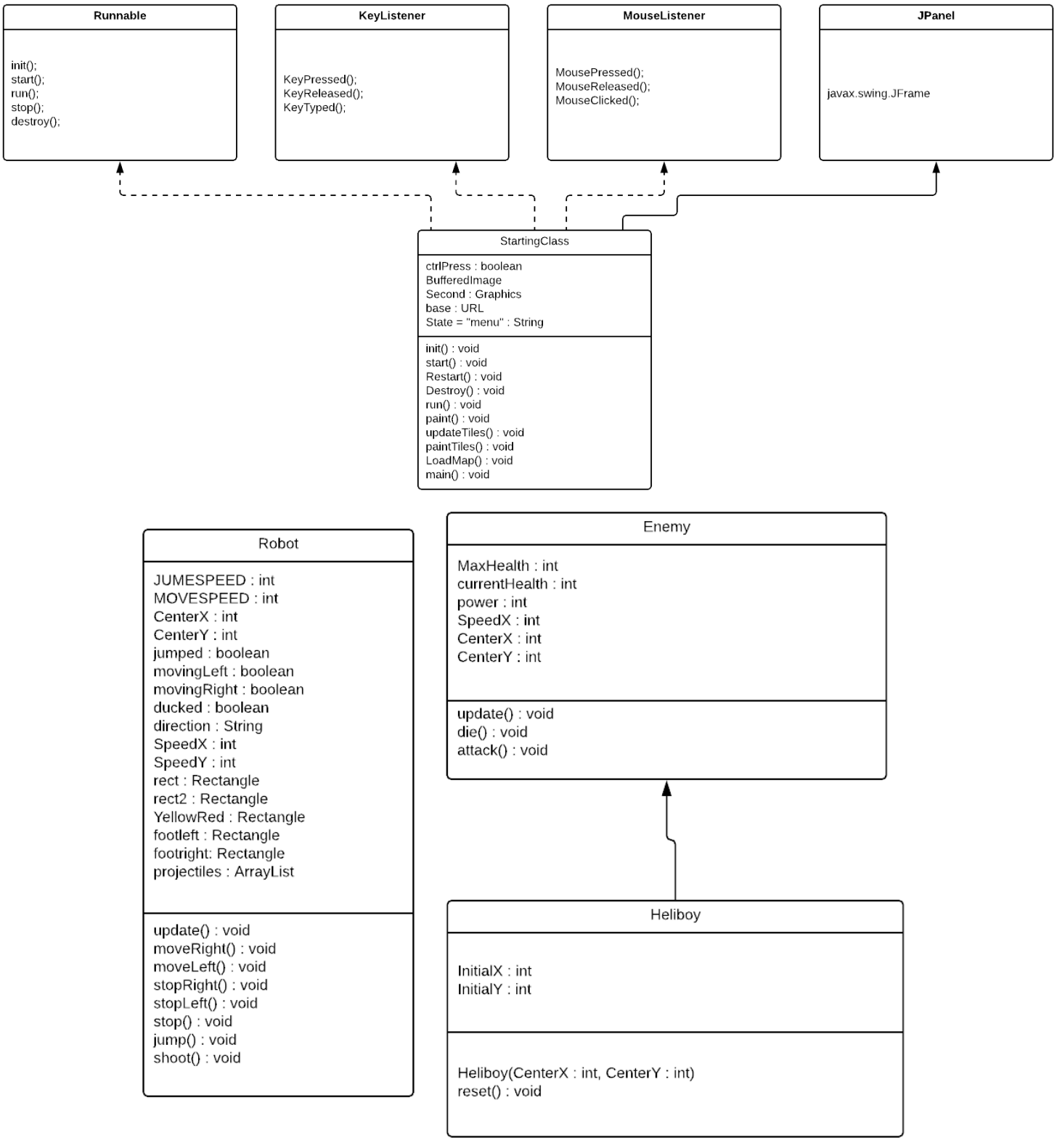
A 2D java Game

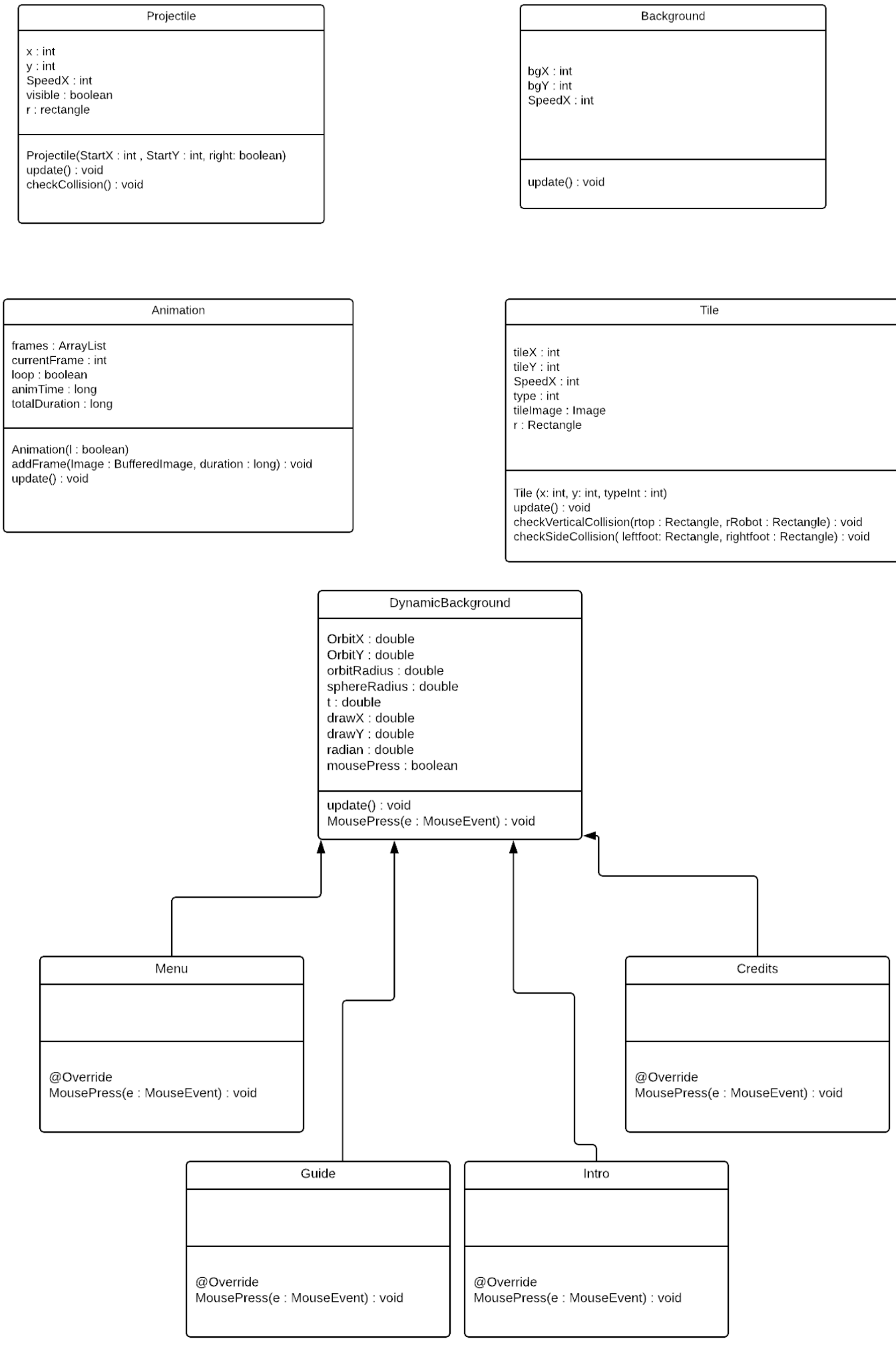
Idea

Idea of the game LOST is inspired by games like “Metal Slug” but with a gloomy and dark theme. The game is meant for entertainment purpose as the player will encounter different challenges and enemies. The character is lost in the mountains and he needs to find his way out but the path isn’t an easy one.

Functional Requirements:

-Java OOP Concepts
-GUI Liberaries SWING, AWT.
-File IO
-Graphic Designs




Projectile
x : int
y : int
SpeedX : int
visible : boolean
r : rectangle
Projectile(StartX : int , StartY : int, right: boolean)
update() : void
checkCollision() : void
Background
bgX : int
bgY : int
SpeedX : int
update() : void
Animation
frames : ArrayList
currentFrame : int
loop : boolean
animTime : long
totalDuration : long
Animation(l : boolean)
addFrame(Image : BufferedImage, duration : long) : void
update() : void
Tile
tileX : int
tileY : int
SpeedX : int
type : int
tileImage : Image
r : Rectangle
Tile (x: int, y: int, typeInt : int)
update() : void
checkVerticalCollision(rtop : Rectangle, rRobot : Rectangle) : void
checkSideCollision(leftfoot: Rectangle, rightfoot : Rectangle) : void
DynamicBackground
OrbitX : double
OrbitY : double
orbitRadius : double
sphereRadius : double
t : double
drawX : double
drawY : double
radian : double
mousePress : boolean
update() : void
MousePress(e : MouseEvent) : void
Menu
@Override
MousePress(e : MouseEvent) : void
Credits
@Override
MousePress(e : MouseEvent) : void
Guide
@Override
MousePress(e : MouseEvent) : void
Intro
@Override
MousePress(e : MouseEvent) : void

OOP Concepts Implemented

-Inheriteance (different Dynamic backgrounds).
-Overriding paint(), repaint(), EventListener and update() methods.
-Implementing runnable interface.
-Running Threads.
-File Handling (To create maps based on text file).
-Exception Handling (To ensure correct image and file loading).
-Creating and running Game Loop.
-Implementing GUI using SWING and AWT.

Major GUIs:

Major GUIs used in project are SWING and AWT


LOST
PLAY
INTRO
GUIDE
CREDITS
EXIT

Dynamic Background

We coded the menu logics so that the background continuously revolves in a small circle with very low speed, to create a dynamic effect with respect to the static logos and buttons

```
public void update(){
                    radian = orbitSpeed * t;
                    drawX = orbitX + orbitRadius * Math.cos(radian);
                    drawY = orbitY + orbitRadius * Math.sin(radian);
                    t+=1;
}
```

Event Handling

For mouse clicks and keyboard control we implemented Event handling using their respective interfaces.

```
switch (e.getKeyCode()) {

    case KeyEvent.VK_ESCAPE:
            State="menu";
            break;

    case KeyEvent.VK_UP:
            System.out.println("Move up");
            break;

    case KeyEvent.VK_DOWN:
            robot.setDucked( true);
            break;

    case KeyEvent.VK_LEFT:
            robot.moveLeft();
            break;

    case KeyEvent.VK_RIGHT:
            robot.moveRight();
            break;
```

Animations:

We also included animations for instance walking animation, crouch animation, enemy flying animation etc in our game so that it looks more dynamic and attractive to the player.

```
public synchronized void update(long elapsedTime) {

        if (frames.size() > 1) {
                animTime += elapsedTime;



                if (!(loop && currentFrame == frames.size()-1)){
                    if (animTime >= totalDuration) {
                            animTime = animTime % totalDuration;
                            currentFrame = 0;

                    }

                    while (animTime > getFrame(currentFrame).endTime) {
                            currentFrame++;

                    }
```

Painting Graphics

We implemented the concept of method overriding to override the paint method to paint the graphics

```
@Override
public void paint(Graphics g) {

    //super.paint(g);

    switch (State) {

        case "start":
            g.drawImage(start, 0, 0, this);
            break;

        case "menu":

            try{
            g.drawImage(menu, (int) MENU.drawX,(int)MENU.drawY,this);
            g.drawImage(logo, 510, 32, this);
            g.drawImage(play,60,250,this);
            g.drawImage(introButton, 60, 300, this);
            g.drawImage(guideButton,60,350, this);
            g.drawImage(creditsButton,60,400,this);
            g.drawImage(exit,60,450,this);
            g.drawImage(teamlogo,60,680,this);
            }
            catch(Exception e){ }
            break;

        case "credits":
```

Main Charcter Movement:

For the movement of our main character we used the eventhandling along with other logic to shift the x-coordinate of background and the character.

```
public void update() {
    if (speedX == 0){
        bg1.setSpeedX(0);
        bg2.setSpeedX(0);
        bg3.setSpeedX(0);
        bg4.setSpeedX(0);
    }

    if (centerX <= 800 && speedX > 0) {
        centerX += speedX;
    }
    if (centerX >=400 && speedX<0){
        centerX += speedX;
    }

    if (speedX > 0 && centerX > 800) {
        bg1.setSpeedX(-MOVESPEED / 5);
        bg2.setSpeedX(-MOVESPEED / 5);
        bg3.setSpeedX(-MOVESPEED / 5);
        bg4.setSpeedX(-MOVESPEED / 5);
    }
     if (speedX < 0 && centerX < 400){
        bg1.setSpeedX(MOVESPEED / 5);
```

Loading Map:

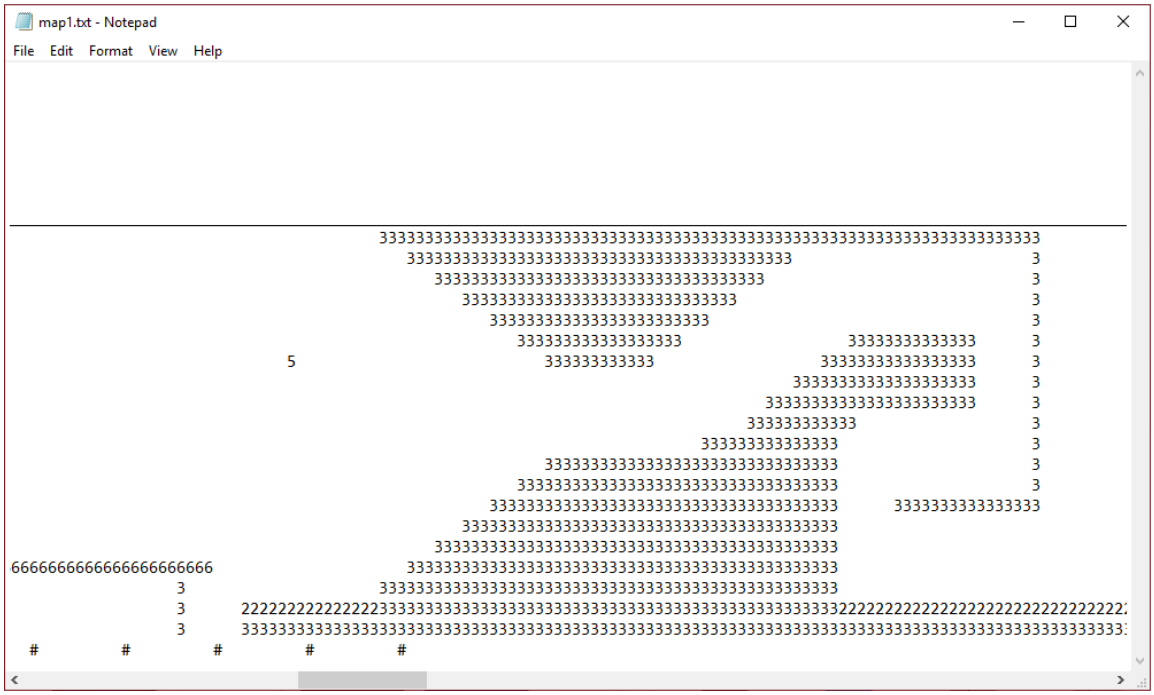
We created entire levels by mapping them out in a text file and then reading it into the Game. For every single digit, we associated an individual tile and all the tiles were different for different digits, giving us more flexibility to create more in-depth or simply put, more interesting levels.

```java
private void loadMap(String filename) throws IOException {
ArrayList lines = new ArrayList();
int width = 0;
int height = 0;

BufferedReader reader = new BufferedReader(new FileReader(filename));
while (true) {
    String line = reader.readLine();
    // no more lines to read
    if (line == null) {
        reader.close();
        break;
    }

    if (!line.startsWith("!")) {
        lines.add(line);
        width = Math.max(width, line.length());

    }
```



Collisions and Physics

We created separate logics for testing collisions of the characters with the level-tiles and other physical objects. Different algorithm had to be used for vertical and bottom collision with different resulting actions to deal with jumping and interaction with the ground.

```
public void checkVerticalCollision(Rectangle rtop, Rectangle rbot) {
    if ( type == 2 || type == 3 || type == 6 ){
        if (rtop.intersects(r)) {
            robot.setCenterY(tileY+100);
            robot.setSpeedY(-robot.JUMPSPEED);
        }

        if (rbot.intersects(r)) {
            robot.setJumped(false);
            robot.setSpeedY(0);
            robot.setCenterY(tileY - 75);
        }
    }
}

 public void checkSideCollision(Rectangle leftfoot, Rectangle rightfoot) {
    if (type == 2 || type == 3 || type == 6 ){

       if (leftfoot.intersects(r)) {
            robot.setCenterX(tileX + 100);
            robot.setSpeedX(0);
        }

        else if (rightfoot.intersects(r)) {
            robot.setCenterX(tileX + 15);
            robot.setSpeedX(0);
        }
```

Main Method

We created a separate thread for the game by implementing the Runnable Interface. We started the thread in a separate start() method and called it within the main() method. Also, we did our best to keep the main function of the game as small as possible by putting everything like thread running and game loops and repaint processes in separate methods. This made it easy for us to debug or further upgrade the Game.

```
public static void main(String[] args){

        JFrame frame = new JFrame("LOST");
        frame.setSize(1366, 768);
        frame.setBackground(Color.black);
        frame.add(starter);
        frame.setUndecorated(true);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        try{
            starter.init();
            starter.start();
        }
        catch (Exception e){
            e.printStackTrace();
        }
}
```

Game Loop

```
@Override
public void run() {
        while (true) {

            switch (State){

                case "start":
                    break;

                case "menu":
                    MENU.update();
                    break;

                case "credits":
                    CREDITS.update();
                    break;

                case "intro":
                    INTRO.update();
                    break;

                case "guide":
                    GUIDE.update();
                    break;
```

The BooM Baby Team

Muhammad Saad Hussaini - 211190
Takreem Masood - 208720
Muhammad Ubaid - 211321

SEECS – BESE 8A
National University of Science and Technology