PROJECT

SHOOTER GAME USING JAVA

NAME : ANIRBAN HAZRA

TABLE OF CONTENTS

TOPIC : Shooter Game using Java

PROBLEM STATEMENT :

A game created using Java , which involves Firing or Shooting through a nozzle at enemies , and destroying them with a collision .

ABSTRACT :

Java is a popular programming language that can be used to create games. One such game is a shooter game that involves a nozzle that can be rotated through an angle of 180^0 . The objective of the game is to destroy enemies that appear from the top of the screen by firing bullets from the nozzle. The game ends when an enemy hits the nozzle.

To play the game, the player must first launch the application. The game screen will appear with the nozzle positioned at the center of the screen. Enemies will start to appear from the top of the screen, moving towards the nozzle in a random pattern. The player must use the left and right arrow keys to rotate the nozzle, aiming it towards the enemies.

INSTRUCTIONS TO PLAY :

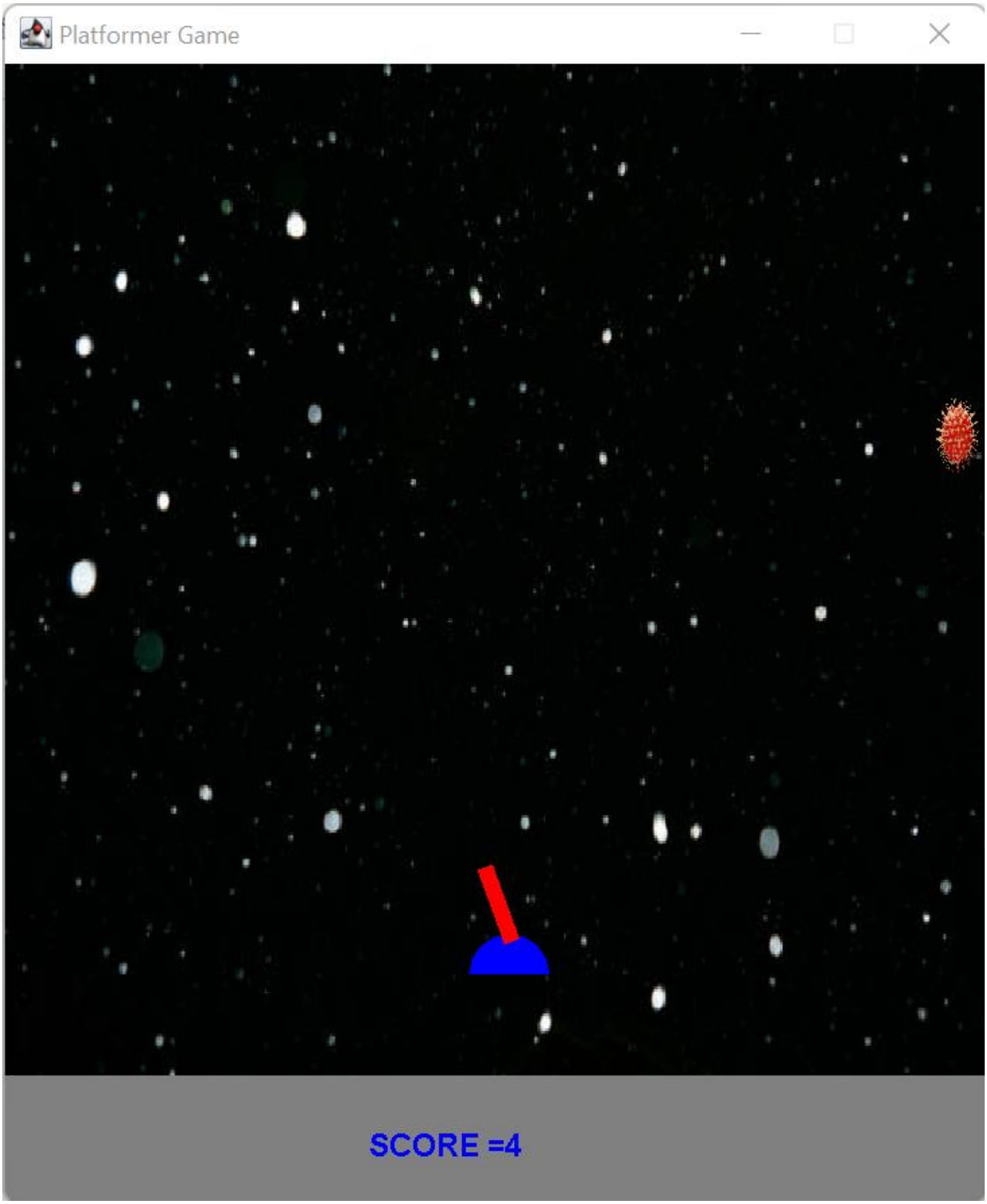
STEP 1 : Once the nozzle is aimed at an enemy, the player can fire a bullet by hitting the space bar. The bullet will travel in a straight line towards the enemy, and if it hits the enemy, the enemy will be destroyed in a burst of flames. The player will receive one point for each enemy destroyed in this way.

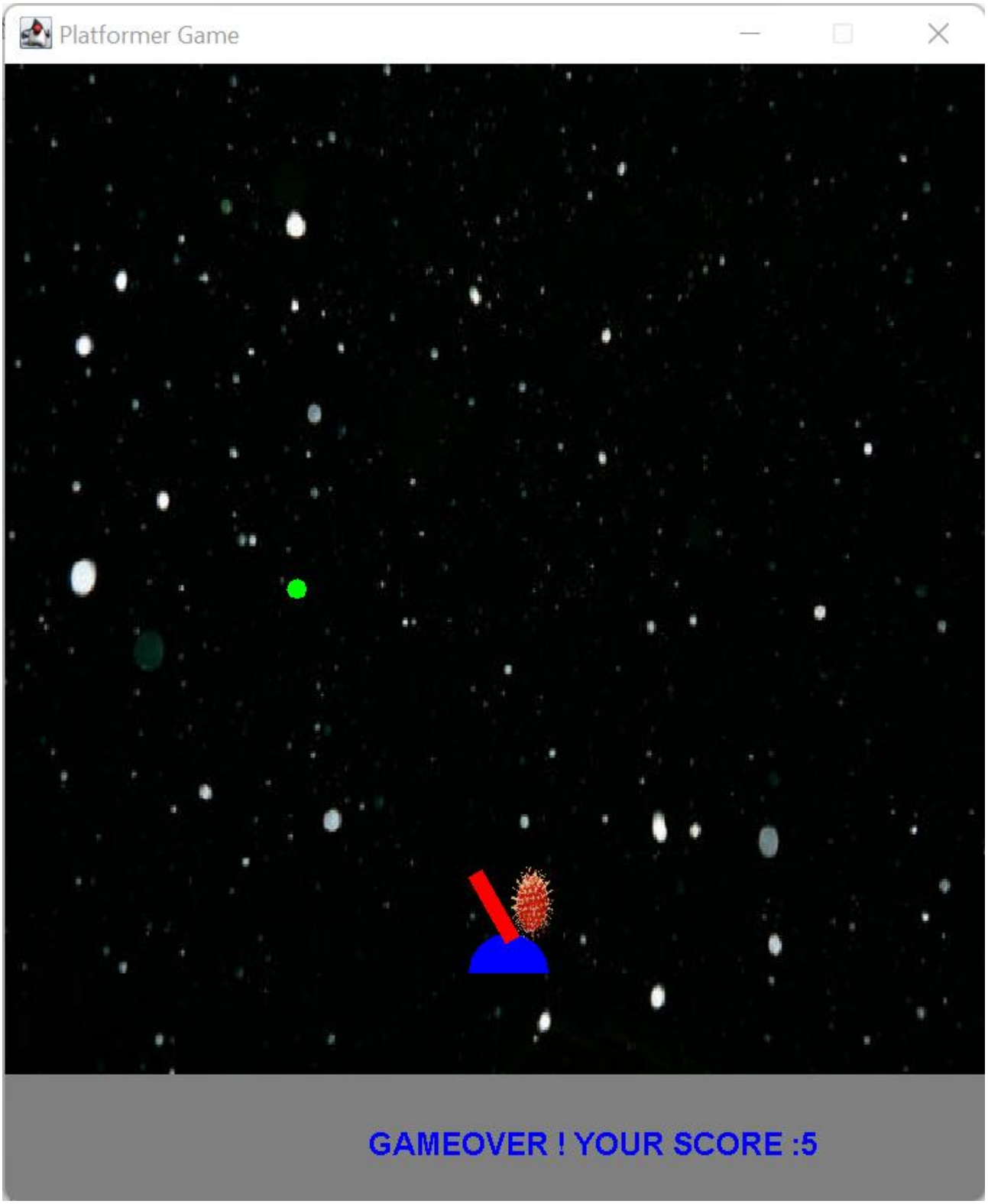
STEP 2 : As the game progresses, the enemies will become faster and more numerous, making it increasingly difficult for the player to avoid getting hit. The player must be quick to react to the movements of the enemies, rotating the nozzle to avoid getting hit.

STEP 3 : If an enemy collides with the nozzle, the game is over, and the player will have to start again from the beginning. The game over screen will appear, showing the player's score and offering them the option to play again or quit.

STEP 4 : To maximize their score, the player must be strategic in their use of bullets, aiming carefully and conserving their ammunition. They must also be quick to react to the movements of the enemies, rotating the nozzle to avoid getting hit.

SCREENSHOTS :




Platformer Game
GAMEOVER ! YOUR SCORE :5

PROBABLE SOFTWARE MODEL :

If this project is later developed into a full scale Software or Application then , various Software Models can be used . One software development life cycle (SDLC) model that could be used for developing the Java game involving firing bullets through a nozzle and destroying enemies which appear from above is the Agile methodology. The Agile methodology is a flexible and iterative approach to software development that emphasizes collaboration, communication, and delivering working software frequently and consistently.

The Agile methodology consists of a series of sprints or iterations, each of which involves planning, executing, and reviewing a set of tasks or features. The process begins with a planning phase, during which the team identifies the goals and objectives of the project, as well as the features or tasks to be completed in the upcoming sprint.

Once the planning phase is complete, the team moves on to the execution phase, during which they develop and test the features or tasks identified in the planning phase. The Agile methodology emphasizes collaboration and communication, with team members working closely together to ensure that the project is progressing according to plan.

At the end of each sprint, the team holds a review and retrospective, during which they evaluate the work completed during the sprint and identify areas for improvement. This feedback is used to guide the planning phase of the next sprint, with the goal of continuously improving the development process and delivering high-quality software on a regular basis.

There are several reasons why the Agile methodology may be a good choice for developing the Java game involving firing bullets through a nozzle and destroying enemies which appear from above. First, the Agile methodology is well-suited for projects that require flexibility and adaptability, which is important for a game development project that may require frequent changes and updates as the game progresses.

Second, the Agile methodology emphasizes collaboration and communication, which can help ensure that all team members are on the same page and working towards a common goal. This can be particularly important for a game development project, which may involve multiple team members with different areas of expertise.

Third, the Agile methodology emphasizes delivering working software frequently and consistently, which can help ensure that the game is developed in an iterative and incremental manner. This can be important for a game development project, as it allows the team to test and refine the game as it is being developed, rather than waiting until the end of the project to identify and fix issues.

Overall, the Agile methodology can be a good choice for developing the Java game involving firing bullets through a nozzle and destroying enemies . Its flexibility, emphasis on collaboration and communication, and focus on delivering working software frequently and consistently can help ensure that the game is developed in a timely and effective manner.

PROGRAM CODE :

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.Rectangle;
class Gun
{   private int x1, x2, y1, y2;
    private int angle, radius;
    Gun()
    {
        x1 = 250;
        y1 = 430;
        angle = 270;
        radius = 30;
        update();
    }

    private void update()
    {
        x2 = (int) (radius * Math.cos(Math.toRadians(angle)) + x1);
        y2 = (int) (radius * Math.sin(Math.toRadians(angle)) + y1);
    }

    public int getX()
    { return x2; }
    public int getY()
    { return y2; }
    public int getAngle()
```

```
{ return angle ; }
public void moveleft()
{
   if (angle > 190)
   {
      angle -= 10;
      update();
   }
}

public void moveright()
{
   if (angle < 350)
   {
      angle += 10;
      update();
   }
}

public void draw(Graphics2D g)
{
   g.setColor(Color.blue);
   g.fillArc(230, 430, 40, 40, 0, 180);
   g.setColor(Color.red);
   BasicStroke wideStroke = new BasicStroke(8.0f);
   g.setStroke( wideStroke );
   g.drawLine(x1, y1, x2 , y2);
}

public Rectangle getObject()
{
   Rectangle r = new Rectangle(250,430,40,40);
```

```
        return r;
    }

    public boolean collides(Enemy e)
    {
        Rectangle r1= this.getObject();
        Rectangle r2= e.getObject();
        return (r1.intersects(r2));
    }
}
class Enemy
{
    int x ,y , dy;
    Image pic;
    Enemy(int x)
    {
        this.x = x;
        y = 0;
        dy = 4;
        pic = Toolkit.getDefaultToolkit().getImage("virus1.png");
      }
     public void update()
    {
        y += dy;
    }

    public void drawEnemy(Graphics2D g )
    {
     g.drawImage(pic, x, y, 40 , 40 , null);

    }
```

```
    public boolean outOfBounds()
    {
        if ((y + 40) > 470)
                return true;
        return false;
    }

    public Rectangle getObject()
    {
        Rectangle r = new Rectangle(x,y,40,40);
        return r;
    }
}
class Bullet
{
    int x, y , dx , dy, radius ;
    boolean fired ;

    public Bullet( )
    {
      x = y = dx = dy = 0 ; radius = 10 ;
      fired = false ;
    }

    public void fire(int i , int j, int a)
    {
     int angle ;
     x = i ; y = j ; angle = a ;
     double r = Math.toRadians(angle) ;
     dx = (int) ( radius*Math.cos(r) ) ;
     dy = (int) ( radius*Math.sin(r) ) ;
    }
```

```
    public void update()
    {
      x = x+dx ; y = y+dy ;
    }

    public void draw(Graphics2D g)
    {
       g.setColor(Color.green);
       g.fillOval(x,y,radius,radius);
    }

    public Rectangle getObject()
    {
       Rectangle r = new Rectangle(x,y,radius,radius);
       return r;
    }

    public boolean collides(Enemy e)
    {
       Rectangle r1= this.getObject();
       Rectangle r2= e.getObject();
       return (r1.intersects(r2));
    }

    public int getX()
    { return x;
    }

    public int getY()
    { return y;
    }
}
```

```
class Explosion
 {
  private int x ,y,radius , maxradius;
  private Color color;
  boolean done , expand ;
  public Explosion( int x, int y , Color c)
  {
     this.x = x;
     this.y = y;
     maxradius = 40 ;
     radius = 0;
     color = c ;
     done = false; expand = true;
  }

  public void update( )
  {
     if(done)
       { return ; }
     if(expand )
      {
      radius++;
      if(radius>maxradius)
        { expand =false ; }
       }
      else
       {
          radius--;
          if ( radius <0)
           done = true ;
       }
```

```
  }
 public void draw(Graphics2D g)
  {
     g.setColor(color);
     BasicStroke wideStroke = new BasicStroke(4.0f);
     g.setStroke( wideStroke );
     g.fillOval(x, y, radius, radius);
  }
 public boolean status( )
 { return done ; }

}

class GamePanel extends JPanel implements  ActionListener , KeyListener
{
   Timer t;
   Gun gun;
   Label l;
   Image pic ;
   Enemy e;
   Bullet bullet;
   Explosion ex;
   int score;
   String str;
   boolean gameover;
   public GamePanel()
   {
      setSize(500,600);
      setLayout(null);
      setFocusable(true);
      addKeyListener(this);
```

```
    setBackground(Color.gray);
    pic = Toolkit.getDefaultToolkit().getImage("night_sky1.jpg");
    score = 0;
    str = "SCORE = "+score ;
    gameover= false;
    gun = new Gun ();
    t=new Timer(7,this);
    t.start();
}

public void actionPerformed(ActionEvent ae)
{
    if (gameover)
    {
       str="GAMEOVER ! YOUR SCORE :"+score;
       t.stop();
    }
      if(e==null || e.outOfBounds())
       {
        int x = ((int)(Math.random() * 460)) ;
        e = new Enemy(x) ;
       }
       e.update();
       if(e!=null)
       {
          if(gun.collides(e))
          {
             gameover = true;
          }
       }
      if(ex!=null)
      {
```

```
        ex.update();
      }
    if(bullet!=null &&  e!=null)
    {
      if(bullet.collides(e))
      {
         score++;
         str = "SCORE =" +score;
         e=null;
         int x1,y1;
         x1=bullet.getX();
         y1=bullet.getY();
         ex= new Explosion(x1,y1,Color.yellow);

      }
    }
      repaint();
}

public void paint(Graphics g)
{
    super.paintComponent(g);
    g.drawImage(pic, 0, 0, 500, 500, this);
    Graphics2D  g2d = (Graphics2D)g;
    gun.draw(g2d);
    if(e!=null)
    e.drawEnemy(g2d);
    if(bullet!=null)
    {
       bullet.draw(g2d);
       bullet.update();
    }
```

```
        if(ex!=null)
        {
            ex.draw(g2d);
        }
        g.setColor(Color.blue);
        g.setFont(new Font("arial",Font.BOLD ,15));
        g.drawString(str,180,540);
    }
    public void keyTyped(KeyEvent ke)
    {
    }
    public void keyReleased(KeyEvent ke)
    {
    }
    public void keyPressed(KeyEvent ke)
    {int a=ke.getKeyCode();
        switch(a)
        {
            case KeyEvent.VK_LEFT :
            gun.moveleft();
            break;
            case KeyEvent.VK_RIGHT :
            gun.moveright();
            break;
            case KeyEvent.VK_SPACE :
            bullet = new Bullet();
            bullet.fire(gun.getX()-5,gun.getY()-5,gun.getAngle());
            break;
        }
    }
}
class MyGameWindow extends JFrame
```

```
{
   public MyGameWindow()
   {
      getContentPane().add(new GamePanel());
      pack();
   }
}
public class Firing
{
   public static void main(String args[])
   {
      MyGameWindow obj=new MyGameWindow();
      obj.setSize(500,600);
      obj.setBackground(Color.orange);
      obj.setTitle("Platformer Game");
      obj.setVisible(true);
      obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      obj.setResizable(false);
   }
}
```

DATA FLOW DIAGRAM :

This DFD shows three main components of the game:

1. User Interface: This component is responsible for providing the game interface to the user, including drawing the game panel, updating the game, displaying the score, restarting the game, and quitting the game.

2. Game Logic: This component is responsible for managing the game logic, including managing the gun, moving the gun left or right, firing bullets, managing enemies, updating enemies, checking for collisions, and managing explosions.

3. Resources: This component is responsible for managing the game resources, including images and sound effects.

The arrows in the DFD represent the flow of data and not the actual flow of control.

EXPLANATION OF THE CODE :

This is a Java program for a simple game where the user controls a gun to shoot at moving enemy objects. The game panel is a JPanel object, which is a container for all the game elements. The game has three main classes: Gun, Enemy, and Bullet, which represent the player's gun, the enemy objects, and the bullets fired by the gun, respectively.

The Gun class has methods for moving the gun left and right and for drawing the gun on the game panel. The Enemy class has methods for updating the position of the enemy object, drawing the enemy object, and checking if the enemy object has gone out of bounds.

The Bullet class has methods for firing the bullet, updating the position of the bullet, drawing the bullet on the game panel, and checking if the bullet has collided with an enemy object. The game panel has a timer that updates the game at a fixed interval.

The panel also has a label that displays the player's score. The score increases each time the player hits an enemy object. The game ends when an enemy object goes out of bounds, and a message is displayed to the player.

There is also an Explosion class that represents an explosion that occurs when an enemy object is hit by a bullet. The explosion is drawn on the game panel for a brief period before disappearing. Overall, this is a simple game that can be easily extended and customized to add more features and levels.

FUTURE PROSPECTIVE WORK :

The game can be enhanced with additional features, such as power-ups that provide the player with temporary advantages, such as increased bullet speed or a temporary shield that protects the nozzle from enemy fire.

The game can also be expanded to include different types of enemies with unique behaviors and abilities, such as enemies that move faster or enemies that require multiple hits to be destroyed.

CONCLUSION :

In conclusion, this shooter game created through Java is a fun and challenging experience that tests the player's reflexes and strategic thinking. With its simple yet engaging gameplay mechanics and intuitive controls, it is sure to provide hours of entertainment for players of all ages and skill levels. The game can be enhanced and expanded to keep players engaged and coming back for more.