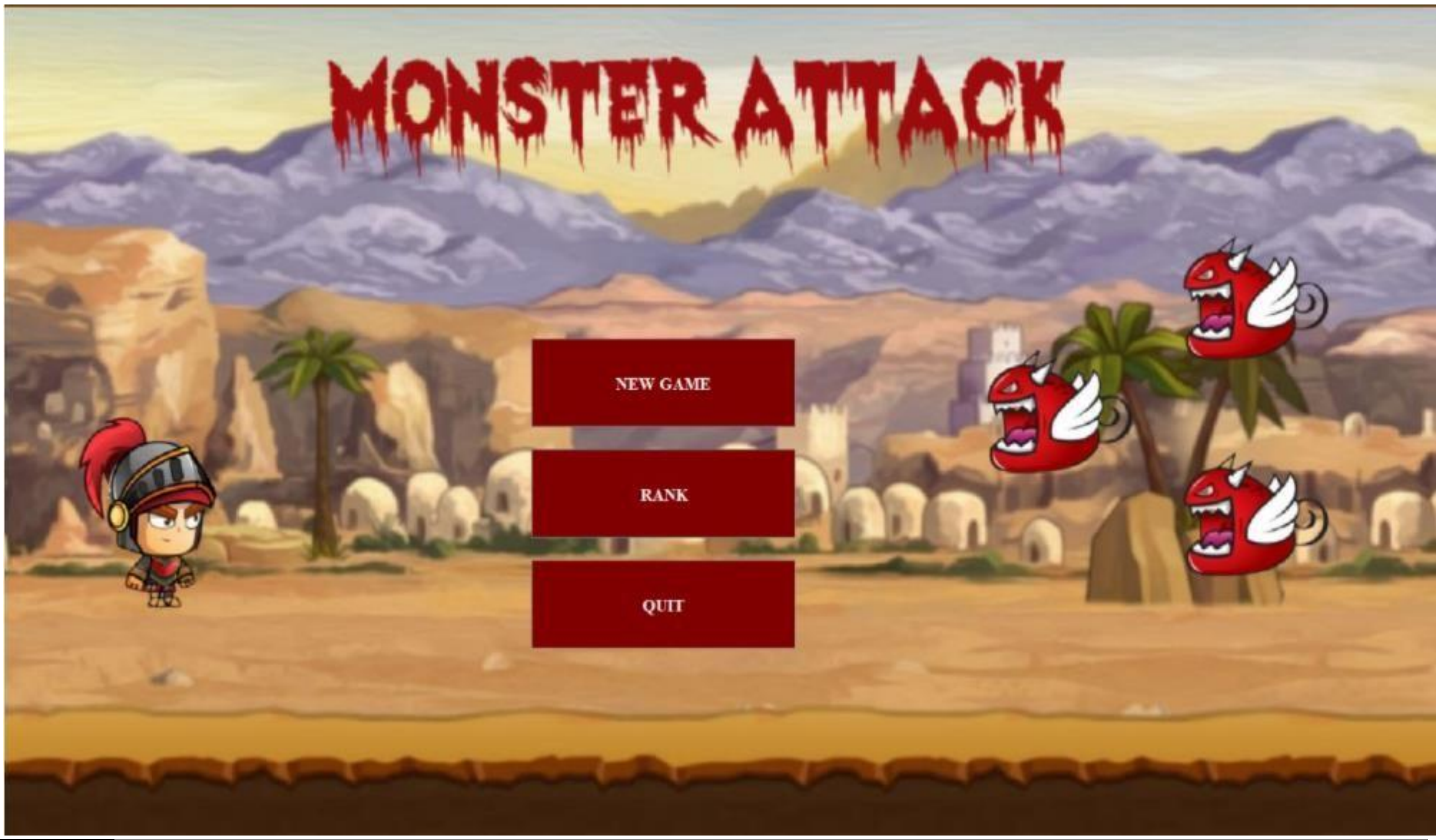


MONSTER ATTACK GAME

REPORT || DSA

FATIMA TUZ ZAHRA || F20CSC07
UZMA FATIMA || F20CSC03
AMBER ASIF || F20CSC02

INTRODUCTION

OBJECTIVES

Monster Attack is an entertaining game which also serves as a brain exercise game where the player must concentrate so that the avatar can survive as long as possible without being attacked by the monsters.
The project objectives are to make a 2D game that uses data structures and GUI implementation to make an easy to understand and entertaining game for the users that could be played on any system that has JVM installed in it.

DESCRIPTION

In this 2D entertaining game, the player has to move the avatar so that it does get caught by any monster. The user will be getting a top view of the whole scene. The player will get 3 lives in each run. The game will not stop till he ran out of all his lives. After his all lives are used then the player will receive the option to either restart the game or just end the game. After the game is over, the avatar's score will be shown on the screen. The lowest the time and highest the distance covered, highest the rank will be.

FUNCTIONAL REQUIREMENT

CRUD FUNCTION

Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform. These functional requirements include the following:

Menu()	Displays options like New Game, Ranks or to quit game.
Table()	Displays all the ranks with the names and the time.
Controller()	Set the locations of enemy and player
AddEnemy()	Add the enemies using the locations set by controller()
SortedInsert()	Insert the player’s name and time taken in a sorted linkedlist.
ReadFromFile()	Reading data from file, storing in a linkedlist.
CallFileStore()	Wrtiting data on file from linkedlkist.
Collision()	Counts the number of time the player collides with an enemy and decreases the lives accordingly.
TimerTask()	Counts the number of seconds passed.
Update()	Update the locations of enemies and player as the game continues to exist and player keeps on pressing keys.
Player()	Controlling player and measuring the distance covered by the player.

NON-FUNCTIONAL REQUIREMENTS

AddEnemy()	Add the enemies using the locations set by controller()
KeyInput()	Takes key presses from user.
KeyRealeased()	Checks when the user release the key.
KeyPressed()	Checks when the user presses the key.
GetName()	Get user’s name.
Disappear()	Dispose the screen.

SOURCE CODE

Menu.java

```
package game;
import java.awt.Font;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import newgame.src.objects.Linked;
import java.awt.event.*;
import java.io.IOException;
import java.util.Vector;
import java.awt.Color;
import java.awt.FlowLayout;
      public class Menu extends JFrame implements ActionListener {
            private static final long serialVersionUID = 1L;
            public String n1;
            Mygame g=new Mygame(1);
            Linked list=new Linked();
            JButton newGame, rank, quit;
            JLabel l1;
            public Menu(){
                  super("MONSTER ATTACK");
            }
                  public void menu() {
                        pack();
                        setSize(1400,800);
                        setContentPane(new JLabel(new 
ImageIcon("C:\\Users\\HP\\Desktop\\Bank\\Monster Attack1.png")));
                        setLayout(new FlowLayout());
                        l1=new JLabel();
                        add(l1);
                        setSize(1400,800);
                        setDefaultCloseOperation(EXIT_ON_CLOSE);

                        setVisible(true);
                        setFocusable(true);
                        newGame = new JButton("New Game");
                        rank = new JButton("Rank");
                        quit = new JButton("Quit");
                        newGame.setFont(new Font(Font.SERIF, Font.BOLD, 16));
                        rank.setFont(new Font(Font.SERIF, Font.BOLD, 16));
                        quit.setFont(new Font(Font.SERIF, Font.BOLD, 16));
                        newGame.setForeground(Color.white);
                        rank.setForeground(Color.white);
                        quit.setForeground(Color.white);
```

```
           Color color=new Color(128,0,0);
           newGame.setBackground(color);
           rank.setBackground(color);
           quit.setBackground(color);
     newGame.setBounds(500,300,250,80);
           rank.setBounds(500,400,250,80);
           quit.setBounds(500,500,250,80);
           add(newGame);
           add(rank);
           add(quit);
           setResizable(true);
           setLayout(null);
           setVisible(true);
           setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
           newGame.addActionListener(this);
           rank.addActionListener(this);
           quit.addActionListener(this);
}
     void table() {
           Color color=new Color(153,0,0);
           JFrame f=new JFrame();
           f.pack();
           DefaultTableModel model = new DefaultTableModel();
           JTable table = new JTable(model);
           table.setBackground(color);
           table.setForeground(Color.white);
           JTableHeader tableHeader = table.getTableHeader();
           tableHeader.setBackground(Color.black);
           tableHeader.setForeground(Color.white);
           tableHeader.setFont(new Font(Font.SERIF, Font.BOLD, 12));
           Vector row = new Vector();
           Linked ll=new Linked();
           Linked l1=new Linked();
           table.setBounds(30,40,200,300);
           JScrollPane sp=new JScrollPane(table);
           f.add(sp);
           f.setSize(300,400);
           f.setVisible(true);
           try {
           int i=0;
           l1=ll.readFromFile();
           Node node;
           Node head = null;
           node=l1.head;
           model.addColumn("RANK");
           model.addColumn("NAME");
           model.addColumn("SCORE");
           model.addColumn("TIME (sec)");
           int rank=1;
           while(node!=null) {
```

```
                      String Name=node.name;
                      int t=node.timetaken;
                      int s=node.totalscore;
                      model.addRow(new Object[]{rank,Name,  t , s});
                      rank++;
                      node=node.next;
                      }
                      } catch (IOException e) {
                      e.printStackTrace();
                      }
                      }
           public void message(int a)
           {

                 if (a==3)
                 {
                      JOptionPane.showMessageDialog (null, "3 LIVES LOST",
"RESULT", JOptionPane.INFORMATION_MESSAGE);
                                           menu();
                      g.disappear();
                 }
           }
public static void main(String[] args) {
                 // TODO Auto-generated method stub
Menu obj=new Menu();
obj.menu();
           }
@Override
public void actionPerformed(ActionEvent a) {
// TODO Auto-generated method stub
if(a.getSource()==newGame) {
n1=JOptionPane.showInputDialog("Name: ");
System.out.println("menu name "+n1);
JOptionPane.showMessageDialog (null, "1.Control avatar with arrow keys . \\n2.
Maximum 3 lives. \n3. The game will end after the player's all lives are lost.
\n4.The rank of the player is calculated on the basis of highest distance and
shortest time.  ", "RULES", JOptionPane.INFORMATION_MESSAGE);
list.setname(n1);
Mygame ob=new Mygame();
dispose();
}
if(a.getSource()==rank) {
table();
}
if(a.getSource()==quit) {
dispose();
}
}
}
```

MyGame.java

```java
package game;
import java.util.Timer;
import java.util.TimerTask;
import java.awt.*;
import javax.swing.*;
public class Mygame {
public static final int WIDTH = 1400,HEIGHT=800;
public static JFrame myFrame;
public String name2;
public Mygame()
{
JFrame myFrame=new JFrame("Monster Attack");
myFrame.pack();
myFrame.setSize(WIDTH,HEIGHT);
myFrame.setLocationRelativeTo(null);
myFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
myFrame.setVisible(true);
myFrame.add(new Game());
}
public Mygame(int n)
{

}
public void disappear()
{
myFrame.dispose();
}
}
```

Game.java

```java
package game;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.Timer;
import game.src.input.Controller;
import game.src.input.Keyinput;
import newgame.src.objects.Player;
```

```
public class Game extends JPanel implements ActionListener{
           private static final long serialVersionUID = 1L;
private String background="/images/dumbledore.png";
           Timer gamelooptimer;
Player p;
Controller c;
      public Game()
      {
           setFocusable(true);
           gamelooptimer=new Timer (10,this);
           gamelooptimer.start();
           p=new Player(5,5);
      c=new Controller();
           addKeyListener(new Keyinput(p));
      }
      public void paint(Graphics g)
      {
           super.paint(g);
           Graphics2D g2d=(Graphics2D) g;
           g2d.drawImage(getBackgrondImage(), 0,0, this);
           p.draw(g2d);
           c.draw(g2d);
      }
      public Image getBackgrondImage()
      {
           ImageIcon i=new ImageIcon(getClass().getResource(background));
           return i.getImage();
      }
@Override
      public void actionPerformed(ActionEvent ee) {
      // TODO Auto-generated method stub

      repaint();
      p.update();
c.update();
      }
}
```

KeyInput.java

```
package game.src.input;

import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

import newgame.src.objects.Player;

public class Keyinput extends KeyAdapter  {
```

```
Player p;
public Keyinput(Player p)
{
      this.p=p;
}
public void keyPressed(KeyEvent e)
{
      p.keyPressed(e);
}
public void keyReleased(KeyEvent e)
{
      p.keyReleased(e);
}
}
```

Controller.java

```
package game.src.input;
import java.awt.Graphics2D;
import java.util.LinkedList;
import newgame.src.objects.Enemy;
public class Controller {
static LinkedList<Enemy> e=new LinkedList<Enemy>();
Enemy TempEnemy;
public Controller() {
      addenemy(new Enemy(100,1300));
            addenemy(new Enemy(500,900));
      addenemy(new Enemy(700,600));
      addenemy(new Enemy(900,300));
      addenemy(new Enemy(200,1100));
      addenemy(new Enemy(400,500));
      addenemy(new Enemy(800,1000));
      addenemy(new Enemy(600,600));
            addenemy(new Enemy(100,1700));
      addenemy(new Enemy(100,2000));
}
public void draw(Graphics2D g2d) {
      for(int i=0;i<10;i++)
      {
             TempEnemy=e.get(i);
             TempEnemy.draw(g2d);
      }
}
public void update()
{
      for(int i=0;i<10;i++)
```

```
     {
             TempEnemy=e.get(i);
             TempEnemy.update();
     }
}
public void addenemy(Enemy enemy)
{
     e.add(enemy);
}
public void removeenemy(Enemy enemy)
{
     e.remove(enemy);
}
public static LinkedList<Enemy> getEnemyBounds()
{
     return e;
}
}
```

Player.java

```
package newgame.src.objects;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Rectangle;
import java.awt.event.KeyEvent;
import java.util.LinkedList;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import game.Globalposition;
import game.Menu;
import game.Mygame;
import game.src.input.Controller;
public class Player extends Globalposition  {
     Linked l=new Linked();
     Menu m=new Menu();
Mygame demo=new Mygame(1);
     Time ob= new Time();
     int t1;
     public int count=0;
     public int score=0;
     public int s=0;

     private String playerimage="/images/a.png";
     int velx=0;
```

```
      int vely=0;
        private LinkedList<Enemy> e=Controller.getEnemyBounds();
      public Player(int x,int y)
      {
            super(x,y);
      }
public void update() {
      x+=velx;
      y+=vely;
      //collision with outside
      if (x<0)
      {
            x=0;
      }
      if (y<0)
      {
            y=0;
      }
      if (x>1400)
      {
            x=0;
            ++s;
      }
      if (y>800) {
            y=0;
      }
      Collision();
}
public void Collision()
{
      for(int i=0;i<e.size();++i)
      {
            if(getBounds().intersects(e.get(i).getBounds()))
            {
                  ++count;
                  System.out.println("count "+count);
            }
            if(count>15)
            {
                  ob.mytimer.cancel();
                  score=(s*1400)+x;
                  l.setscore(score);
                  l.settime(ob.c);
                  l.add();
                  String st="3 lives lost";
                  System.out.println(st);
                  m.message(3);
                  }
      }
}
```

```
public void keyPressed(KeyEvent e)
{
      int key=e.getKeyCode();

      if(key==KeyEvent.VK_RIGHT)
      {
            velx=5;
      }
      else if (key==KeyEvent.VK_LEFT)
      {
            velx=-5;
      }
      else if (key==KeyEvent.VK_DOWN)
      {
            vely=5;
      }
      else if (key==KeyEvent.VK_UP)
      {
            vely=-5;
      }
}
public void keyReleased(KeyEvent e)
{
      int key=e.getKeyCode();

      if(key==KeyEvent.VK_RIGHT)
      {
            velx=0;
      }
      else if (key==KeyEvent.VK_LEFT)
      {
            velx=0;
      }
      else if (key==KeyEvent.VK_DOWN)
      {
            vely=0;
      }
      else if (key==KeyEvent.VK_UP)
      {
            vely=0;
      }
      }
public Rectangle getBounds()
{
      return new Rectangle (x,y,50,50);
}
public void draw(Graphics2D g2d)
{
      g2d.drawImage(getPlayerImage(), x,y, null);
```

```
	g2d.draw(getBounds());
}
public Image getPlayerImage()
{
	ImageIcon i=new ImageIcon(getClass().getResource(playerimage));
	return i.getImage();
}
}
```

GlobalPosition.java

```
package game;
public class Globalposition {
	public int x;
	public int y;
	public Globalposition(int x,int y)
	{
		this.x=x;
		this.y=y;
	}
}
```

Time.java

```
package newgame.src.objects;
import java.util.Timer;
import java.util.TimerTask;

import javax.swing.JOptionPane;
public class Time {
	public int Secondspassed=0;
	public int c=0;
	Timer mytimer=new Timer();
	TimerTask task=new TimerTask() {
		public void run()
		{
			Secondspassed++;
			++c;
			System.out.println("Seconds passed  "+Secondspassed);
		}
		};

	public Time(){
		// TODO Auto-generated method stub
		mytimer.scheduleAtFixedRate(task, 1000, 1000);
```

```
     }
}
```

Linked.java

```
package newgame.src.objects;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.Formatter;
import game.Node;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.File;
public class Linked  implements Serializable {
      private static final long serialVersionUID = 1L;
      public Node head;
      public static String namef1;
       public  String namef;
       public int timef;
       public int scoref;
       Linked l;
       Linked l2;
           void sortedInsert(String n,int s,int time)
    {
                System.out.println("passed "+n+time);
           Node newNode= new Node();
           newNode.name=n;
          newNode.totalscore=s;
    newNode.timetaken=time;
        Node current;
        /* Special case for head node */
        if (head == null || (head.timetaken
>= newNode.timetaken && head.totalscore <= newNode.totalscore)) {
            newNode.next = head;
            head = newNode;
        }
        else {
            /* Locate the node before point of insertion. */
            current = head;
            while (current.next != null
&& current.next.timetaken < newNode.timetaken && current.next.totalscore
>newNode.totalscore)
```

```
                current = current.next;
            newNode.next = current.next;
            current.next = newNode;
        }
           System.out.println("passed1 "+namef1+scoref);
    }
     public void printList()
    {
        Node temp = head;
        while (temp != null) {
            System.out.println(temp.name+" "+temp.timetaken +" "
+temp.totalscore  );
            temp = temp.next;
        }
    }
     public void add() {
                Linked l=new Linked();
                 Linked l2=new Linked();
                 System.out.println("passed ss "+namef1+scoref);
                l.sortedInsert(namef1,  scoref,timef);
                l.printList();
                l.callFileStore();
                try {
                l2=l.readFromFile();
                l2.printList();
                }
                catch(Exception ex) {
                       System.out.println("unsuccessful");
                       ex.printStackTrace();
                       }
     }
     public void setname(String n) {
           namef=n;
           System.out.println("namel "+namef);
           System.out.println("namels "+namef1);
           getname();
           }
     public void getname() {
           namef1=namef;
           System.out.println("namels "+namef1);
     }
     public void settime(int t) {
           timef=t;
           System.out.println("timel "+timef);
     }
     public void setscore(int s) {
           scoref=s;
           System.out.println("scoref "+scoref);
     }
       public Linked readFromFile() throws IOException{
             Linked obj= new Linked();
        Node newNode= new Node();
              FileReader file = new FileReader("Newdata.txt");
            try (BufferedReader buffer = new BufferedReader(file)) {
                String line= null;
                   String[] record= new String[3];
System.out.println("reading from file");
                   while((line=buffer.readLine())!=null){
                      record = line.split(",");
                      newNode.name = record[0];
                       newNode.timetaken = Integer.parseInt(record[1]);
                       newNode.totalscore = Integer.parseInt(record[2]);

                       System.out.println("1   "+newNode.name+"
"+newNode.totalscore+" "+newNode.timetaken);
                       obj.sortedInsert(newNode.name,
newNode.totalscore,newNode.timetaken);
                   }
           } catch (NumberFormatException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
           }
            return obj;
        }
      void callFileStore(){
            try {
                 File file= new File("Newdata.txt");
                 try (FileWriter fw = new FileWriter(file, true)) {
                      fw.flush();
                }
                 }
                 catch(Exception ex) {
                       System.out.println("error in filestore");
                 ex.printStackTrace();
                 }
            System.out.println("Storing on a file");
           Node newnode=new Node();
           String record=null;
           Node current= this.head;
            try {
                      FileWriter file= new FileWriter("Newdata.txt",true);
                      BufferedWriter writer= new BufferedWriter(file);
                while(current!=null) {
                       newnode.name= current.name;
                       newnode.timetaken=current.timetaken;
                       newnode.totalscore=current.totalscore;
                record=
current.name+","+current.totalscore+","+current.timetaken;
                System.out.println(record.split(","));
                      writer.write(record);
```

```
                        writer.newLine();
                        current=current.next;
                     }
                  writer.close();
                  }
                  catch(Exception ex) {
                         System.out.println("Stored unsuccessful");
                        ex.printStackTrace();
                  }
              System.out.println("Stored successfully");
        }
}
```

Node.java

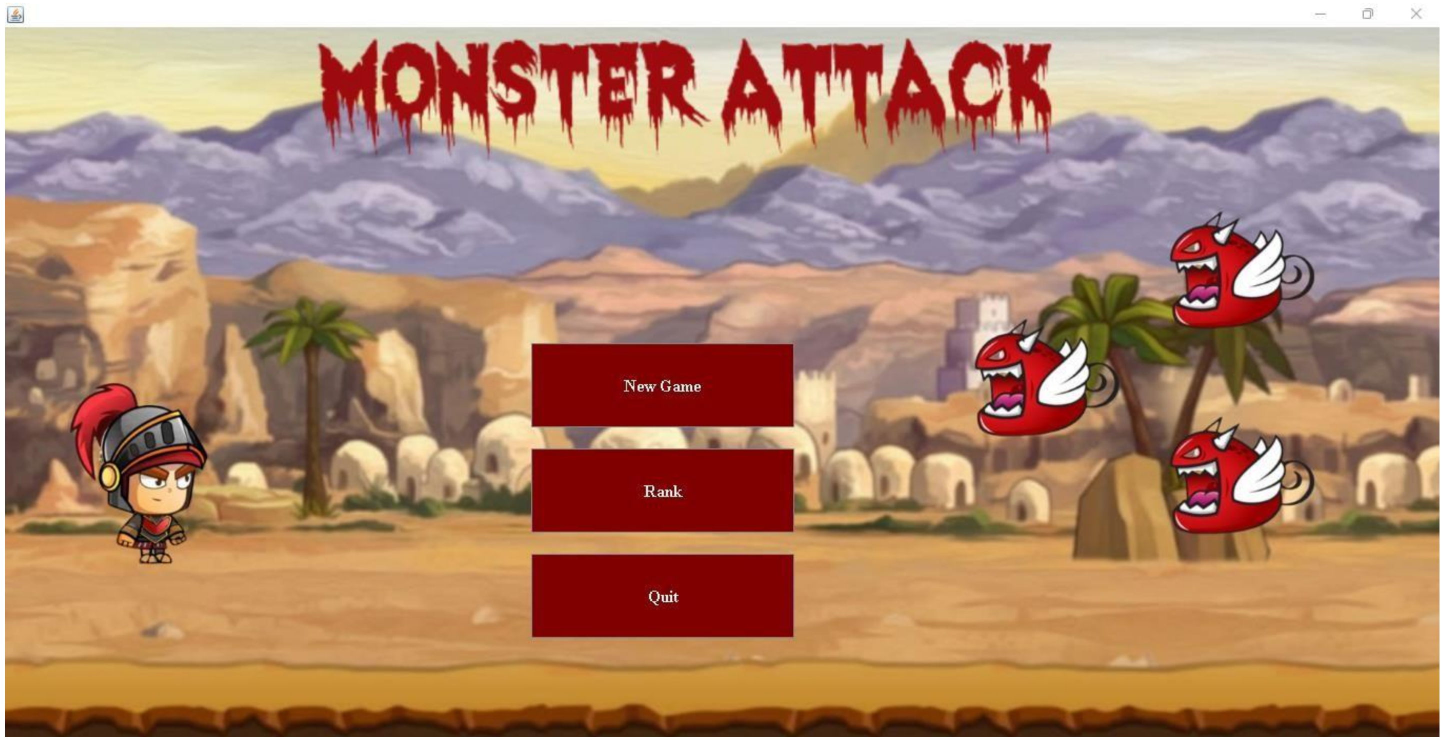
```
package game;
public class Node {
           public int timetaken;
           public int totalscore;
           public String name;
           public Node next;

}
```

Enemy.java

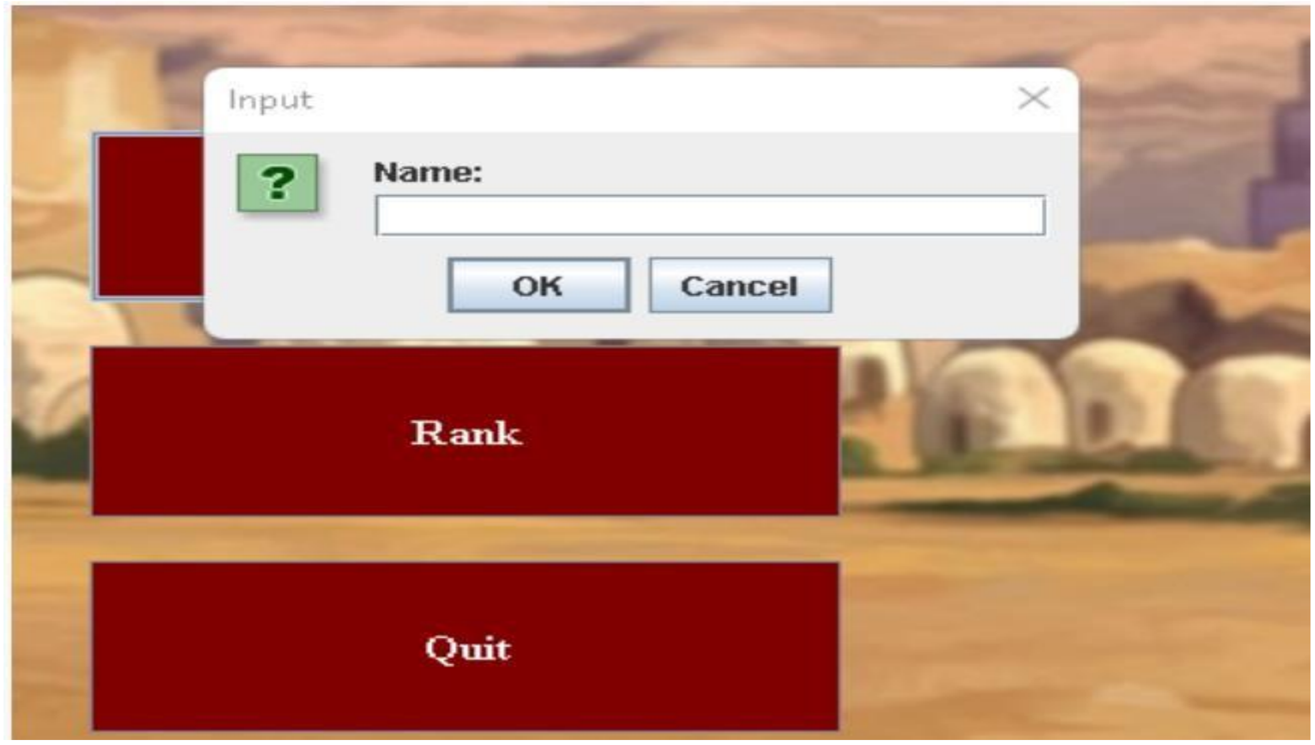
```
package newgame.src.objects;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Rectangle;
import javax.swing.ImageIcon;
import game.Globalposition;
import game.Mygame;
public class Enemy extends Globalposition {
private String image="/images/ene.png";
int speed=8;
int speed1=8;
     public Enemy(int x,int y)
{
     super(x,y);
}
```

```
	public void update()
	{
		x+=speed;
		y+=speed1;
		if(x>Mygame.WIDTH-32)
		{
			speed=-8;
		}
		if(x<0)
		{
			speed=8;
		}
		if(y>Mygame.HEIGHT)
		{
			speed1=-8;
		}
		if(y<0)
		{
			speed1=8;
		}
	}
public void draw (Graphics2D g2d)
{
	g2d.drawImage(getEnemyImage(),x,y,null);
}
public Rectangle getBounds()
{
	return new Rectangle (x,y,25,25);
}
public Image getEnemyImage()
{
	ImageIcon i=new ImageIcon(getClass().getResource(image));
	return i.getImage();
}
}
```

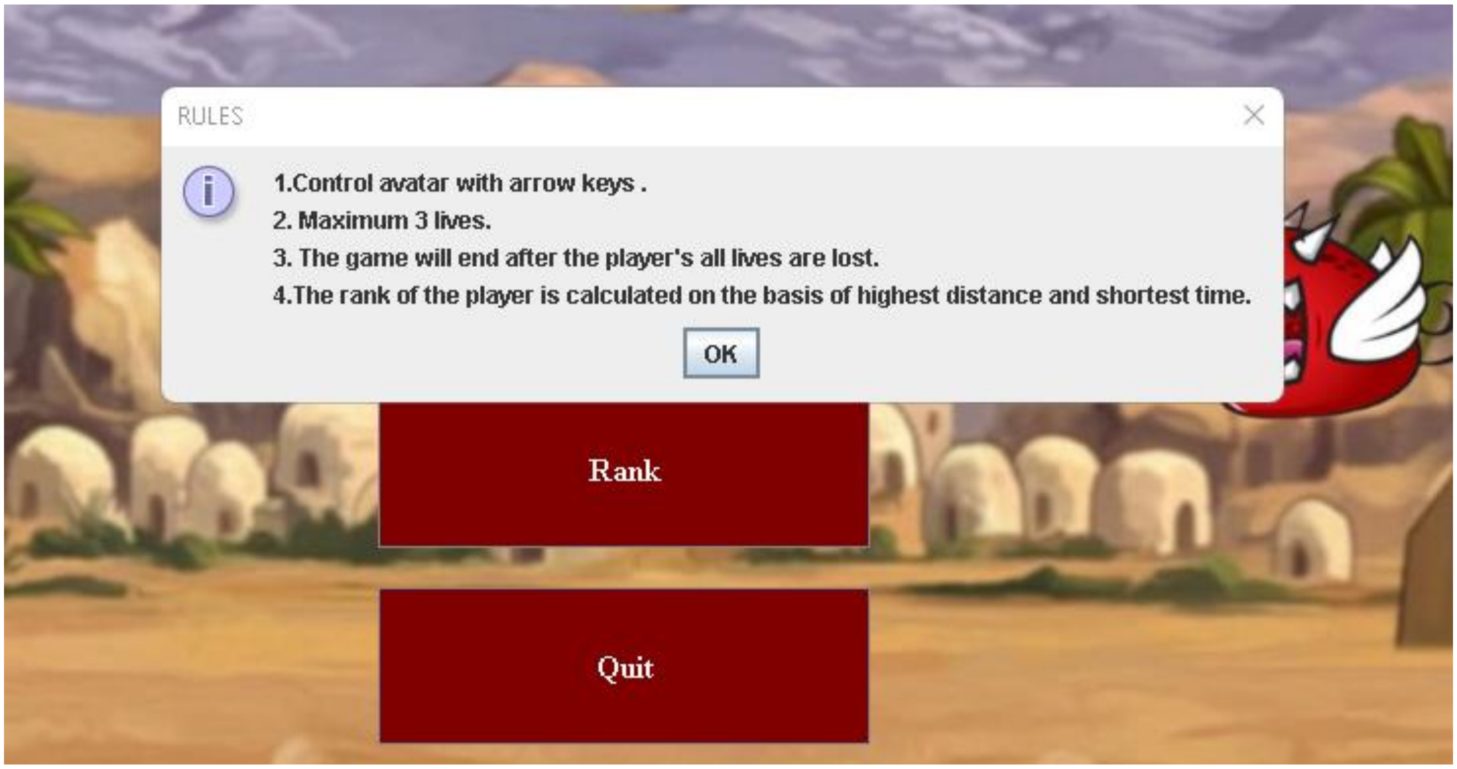
OUTPUT



This is the initial screen of our game.

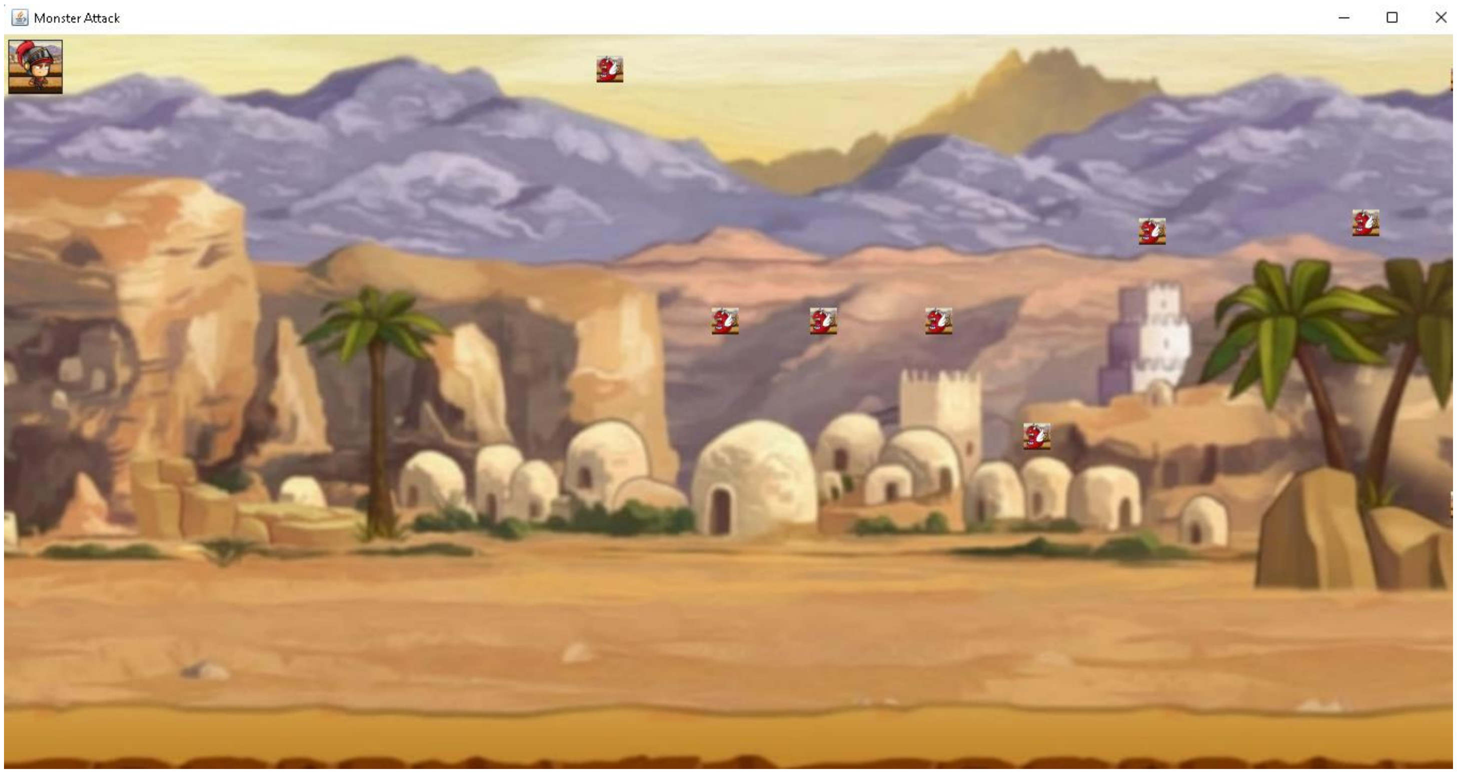


Once you clicked **New Game,** a dialog box will appear where you have to enter your name.

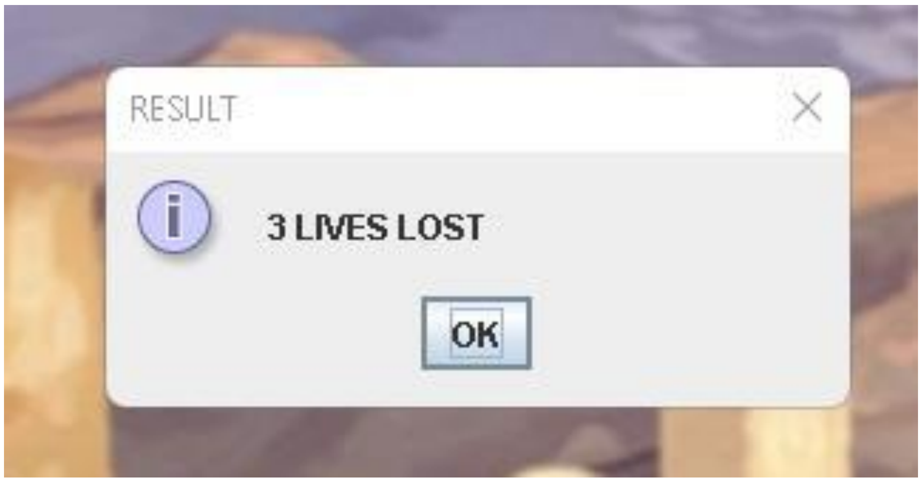

RULES

1.Control avatar with arrow keys .
2. Maximum 3 lives.
3. The game will end after the player's all lives are lost.
4.The rank of the player is calculated on the basis of highest distance and shortest time.

After entering the name, a dialog box will appear that will display the rules of the game.



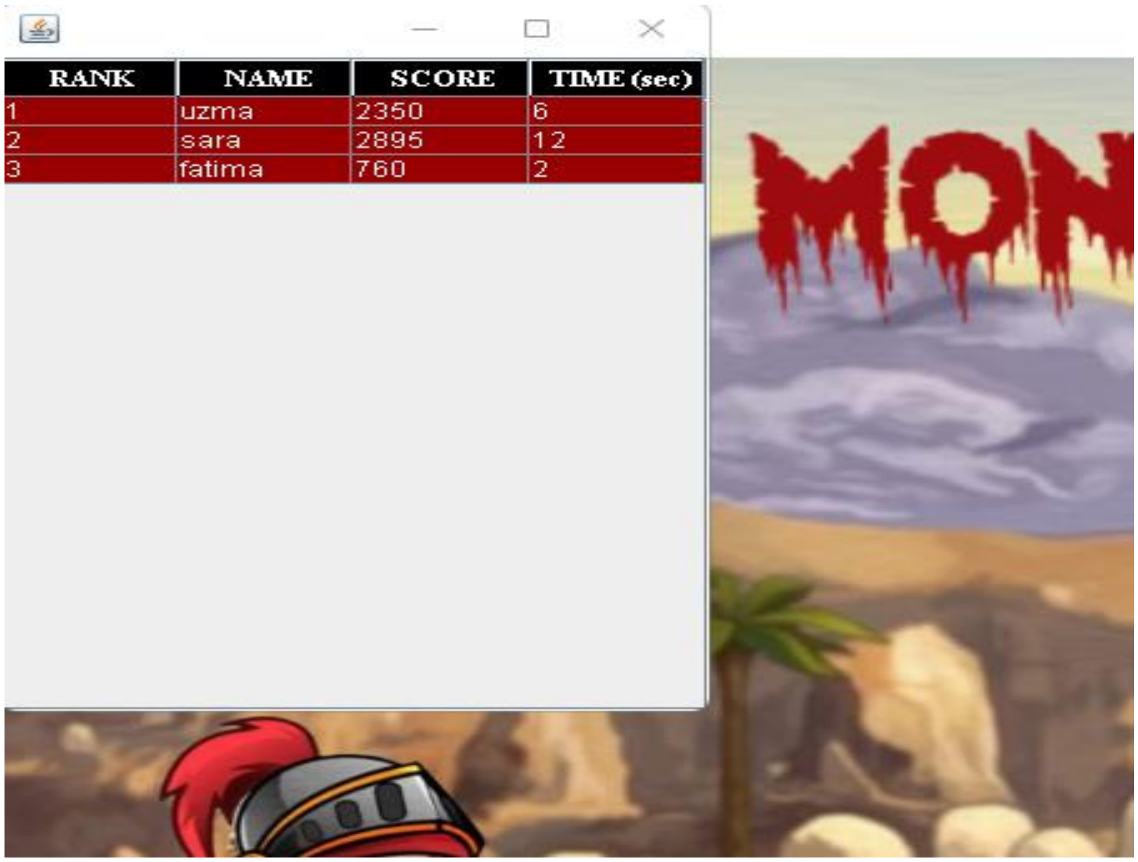
After that, the game will start. The player have to move the maximum distance possible without bumping into the enemies. Each collision will cost one life.



RESULT

3 LIVES LOST

The player will be given 3 lives and if the player loses all of them, the game will end and a message box will be shown.

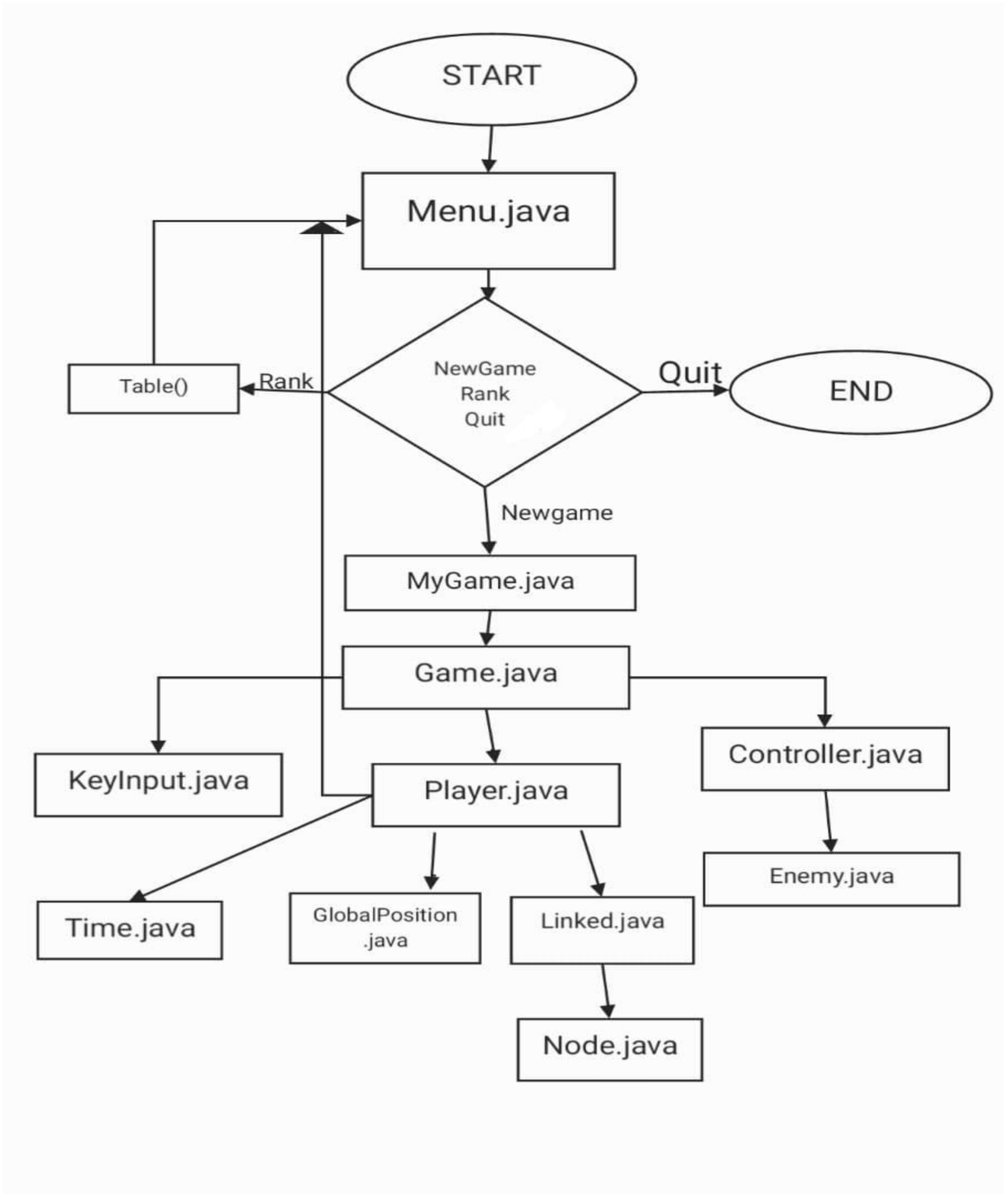


RANK	NAME	SCORE	TIME (sec)
1	uzma	2350	6
2	sara	2895	12
3	fatima	760	2

After the game is over, if the player wants to see his/her rank , then the player can see it via the **rank** option.

To exit the game, the player will click the **quit** option.

FLOWCHART


START
Menu.java
Table()
Rank
NewGame
Rank
Quit
Quit
END
Newgame
MyGame.java
Game.java
KeyInput.java
Player.java
Controller.java
Enemy.java
Time.java
GlobalPosition
.java
Linked.java
Node.java