INTERNSHIP PROJECT

ON

Snake Master Game

BACHELOR OF TECHNOLOGY

(COMPUTER SCIENCE & ENGINEERING)

SUBMITTED BY:

Nisha Kumari (19024010166)

UNDER THE GUIDANCE OF:

Mrs. Bhanu Priya

IN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ROORKEE INSTITUTE OF TECHNOLOGY

ROORKEE, UTTRAKHAND, INDIA

(2021-2022)

CERTIFICATE

I hereby certify that the work which is being presented in these entitled "**Snake Master Game**" in partial fulfilment of the requirement for the award of degree of Bachelor of Technology and submitted in Department of Computer Science of Roorkee Institute of Technology, Roorkee, is an authentic record of my own work carried out under the supervision of **Mrs Bhanu Priya.**

The matter presented in this report has not been submitted by me anywhere for the award of any other Degree of this or any other institute.

NISHA KUMARI

This is to clarify that the above statement made by the candidate is correct to the best of our knowledge.

Date: 09 MAY 2022

HOD

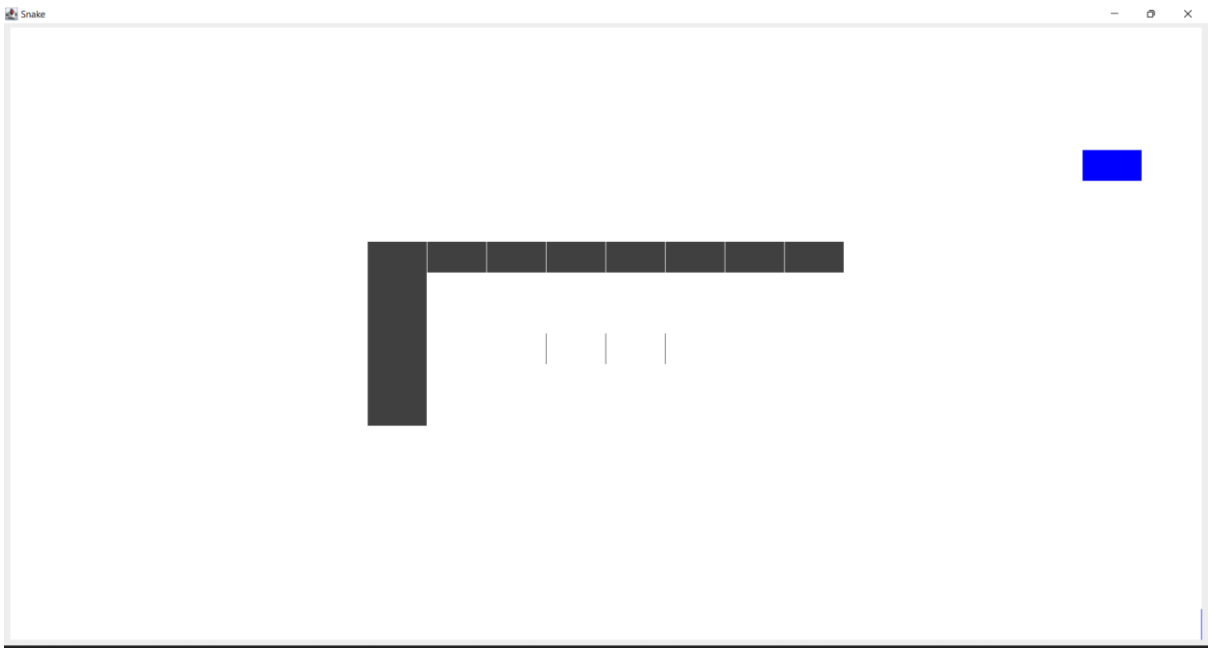
(DR. DEEPAK ARYA)

Project IN charge

(Mrs. Bhanu Priya)

STUDENT INFORMATION

- NAME: ***NISHA KUMARI***
- COURSE: ***BTECH***
- BRANCH: ***COMPUTER SCIENCE AND ENGINEERING***
- UNIVERSITY ROLL NUMBER: ***190240101066***
- YEAR: ***3rd YEAR (2019-23)***
- COLLEGE NAME: ***ROORKEE INSTITUTE OF TECHNOLOGY***
- PROJECT NAME: ***SNAKE MASTER GAME USING JAVA***
- SUBMITTED TO: ***Mrs. Bhanu Priya***



ABOUT PROJECT:

- Basically this is a snake master game (here written in JAVA language) which is designed in such a way that we have a food block and a snake in which the snake is made up of blocks and the food is also made up of block. The snake size increases whenever it eats the food block and this is the prime purpose of the game to move the snake in such a way that it doesn't touch its own body otherwise the snake will die and the game will end.
- In this project, we use JAVA programming language because JAVA is currently being widely used in developing various games due to the vast pool of library it has, is cross-platform, easy to write, etc.
- Here, we used various JAVA components like:
    - AWT
    - Swing
    - Event handling
    - Array list
    - JFrame
    - Grid layout
- This is simple game which can be installed in televisions as well as in smart phones which small kids can play to increase their mind IQ.

Mechanism involved in developing the Snake Master Game using JAVA–

- **Event handling of java**
- **Main Function**
- **Swing component of java**
- **Action Event functions of java**
- **AWT graphics of java**
- **JFrame component of java**
- **Grid layout concept of java**

LANGUAGES USED IN THE PROJECT

In this project, Snake Master Game, JAVA programming language is used in both backend and at frontend. The main language used in both is JAVA in which swing, awt, event handling library, JFrame is also used at great extent.

SOURCE CODE

THE MAIN FUNCTION CODE IS:

```
import javax.swing.JFrame;


public class Main {


    public static void main(String[] args) {


        //Creating the window with all its awesome snaky features
        Window f1= new Window();


        //Setting up the window settings
        f1.setTitle("Snake");
        f1.setSize(300,300);
        f1.setVisible(true);


    f1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
    }
}
```

THE CODE FOR BLOCKS IS:

```
import java.util.ArrayList;
import java.awt.Color;

public class DataOfSquare {


    //ArrayList that'll contain the colors
    ArrayList<Color> C =new ArrayList<Color>();
    int color; //2: snake , 1: food, 0:empty
    SquarePanel square;
    public DataOfSquare(int col){

        //Lets add the color to the arrayList
```

```
            C.add(Color.darkGray);//0
            C.add(Color.BLUE);    //1
            C.add(Color.white);   //2
            color=col;
            square = new SquarePanel(C.get(color));
        }
        public void lightMeUp(int c){
            square.ChangeColor(C.get(c));
        }
}
```

THE CODE FOR KEYBOARD ACTION EVENT HANDLING IS:

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

```
public class KeyboardListener extends KeyAdapter{

        public void keyPressed(KeyEvent e){
           switch(e.getKeyCode()){
              case 39: // -> Right
                                  //if it's not the opposite
direction

    if(ThreadsController.directionSnake!=2)

    ThreadsController.directionSnake=1;
                                  break;
              case 38: // -> Top

    if(ThreadsController.directionSnake!=4)

    ThreadsController.directionSnake=3;
                                  break;

              case 37: // -> Left
```

```
        if(ThreadsController.directionSnake!=1)

        ThreadsController.directionSnake=2;
                                    break;

                    case 40: // -> Bottom

        if(ThreadsController.directionSnake!=3)

        ThreadsController.directionSnake=4;
                                    break;

                    default: break;
                }
            }

}
```

THE CODE FOR BACKGROUND SQAURE PANEL SETTING IS:

```
import java.awt.Color;
import javax.swing.JPanel;

public class SquarePanel extends JPanel{

    private static final long serialVersionUID = 1L;

    public SquarePanel(Color d){
        this.setBackground(d);
    }

    public void ChangeColor(Color d){
        this.setBackground(d);
        this.repaint();
```

```
    }

}
```

CODE FOR THE MAIN LOGIC OF THE PROGRAM (THE MOST IMPORTANT CODE):

```
import java.util.ArrayList;



//Controls all the game logic .. most important class in this project.
public class ThreadsController extends Thread {
    ArrayList<ArrayList<DataOfSquare>> Squares= new ArrayList<ArrayList<DataOfSquare>>();
    Tuple headSnakePos;
    int sizeSnake=3;
    long speed = 80;
    public static int directionSnake ;
```

```
        ArrayList<Tuple> positions = new
ArrayList<Tuple>();
        Tuple foodPosition;

        //Constructor of ControlleurThread
        ThreadsController(Tuple positionDepart){
            //Get all the threads
            Squares=Window.Grid;

            headSnakePos=new
Tuple(positionDepart.x,positionDepart.y);
            directionSnake = 1;

            //!!! Pointer !!!!
            Tuple headPos = new
Tuple(headSnakePos.getX(),headSnakePos.getY());
            positions.add(headPos);
```

```
            foodPosition= new Tuple(Window.height-
1,Window.width-1);
            spawnFood(foodPosition);


        }


        //Important part :
        public void run() {
            while(true){
                moveInterne(directionSnake);
                checkCollision();
                moveExterne();
                deleteTail();
                pauser();
            }
        }


        //delay between each move of the snake
```

```
        private void pauser(){
            try {
                    sleep(speed);
            } catch (InterruptedException e) {
                    e.printStackTrace();
            }
        }

        //Checking if the snake bites itself or is eating
        private void checkCollision() {
            Tuple posCritique =
positions.get(positions.size()-1);
            for(int i = 0;i<=positions.size()-2;i++){
                boolean biteItself =
posCritique.getX()==positions.get(i).getX() &&
posCritique.getY()==positions.get(i).getY();
                if(biteItself){
                    stopTheGame();
                }
```

```
            }


            boolean eatingFood =
posCritique.getX()==foodPosition.y &&
posCritique.getY()==foodPosition.x;
            if(eatingFood){
                System.out.println("Yummy food!");
                sizeSnake=sizeSnake+1;
                    foodPosition =
getValAleaNotInSnake();


                spawnFood(foodPosition);
            }
        }


        //Stops The Game
        private void stopTheGame(){
            System.out.println("COLISION! YOU LOST
\n");
```

```
            while(true){
                pauser();
            }
        }

        //Put food in a position and displays it
        private void spawnFood(Tuple foodPositionIn){

       Squares.get(foodPositionIn.x).get(foodPositionIn.
y).lightMeUp(1);
        }

        //return a position not occupied by the snake
        private Tuple getValAleaNotInSnake(){
            Tuple p ;
            int ranX= 0 + (int)(Math.random()*19);
            int ranY= 0 + (int)(Math.random()*19);
            p=new Tuple(ranX,ranY);
            for(int i = 0;i<=positions.size()-1;i++){
```

```
                if(p.getY()==positions.get(i).getX() &&
p.getX()==positions.get(i).getY()){
                    ranX= 0 + (int)(Math.random()*19);
                    ranY= 0 + (int)(Math.random()*19);
                    p=new Tuple(ranX,ranY);
                    i=0;
                }
            }
            return p;
        }

        //Moves the head of the snake and refreshes the
positions in the arraylist
        //1:right 2:left 3:top 4:bottom 0:nothing
        private void moveInterne(int dir){
            switch(dir){
                case 4:
```

```
headSnakePos.ChangeData(headSnakePos.x,(headSnakePos.y+1)%20);
                    positions.add(new Tuple(headSnakePos.x,headSnakePos.y));
                    break;
                case 3:
                    if(headSnakePos.y-1<0){

headSnakePos.ChangeData(headSnakePos.x,19);
                    }
                    else{

headSnakePos.ChangeData(headSnakePos.x,Math.abs(headSnakePos.y-1)%20);
                    }
                    positions.add(new Tuple(headSnakePos.x,headSnakePos.y));
                    break;
                case 2:
```

```
                    if(headSnakePos.x-1<0){

headSnakePos.ChangeData(19,headSnakePos.y);
                    }
                    else{

headSnakePos.ChangeData(Math.abs(headSnakePos.
x-1)%20,headSnakePos.y);
                    }
                    positions.add(new
Tuple(headSnakePos.x,headSnakePos.y));

                    break;
                case 1:

headSnakePos.ChangeData(Math.abs(headSnakePos.
x+1)%20,headSnakePos.y);
                    positions.add(new
Tuple(headSnakePos.x,headSnakePos.y));
                    break;
```

```
        }
    }

    //Refresh the squares that needs to be
    private void moveExterne(){
        for(Tuple t : positions){
            int y = t.getX();
            int x = t.getY();
            Squares.get(x).get(y).lightMeUp(0);

        }
    }

    //Refreshes the tail of the snake, by removing the superfluous data in positions arraylist
    //and refreshing the display of the things that is removed
    private void deleteTail(){
        int cmpt = sizeSnake;
```

```
            for(int i = positions.size()-1;i>=0;i--){
                if(cmpt==0){
                    Tuple t = positions.get(i);

Squares.get(t.y).get(t.x).lightMeUp(2);
                }
                else{
                    cmpt--;
                }
            }
            cmpt = sizeSnake;
            for(int i = positions.size()-1;i>=0;i--){
                if(cmpt==0){
                    positions.remove(i);
                }
                else{
                    cmpt--;
                }
```

```
            }
        }
}
```

THE TUPLE CODE IS:

```
public class Tuple {
        public  int x;
        public  int y;
        public int xf;
        public int yf;

        public Tuple(int x, int y) {
          this.x = x;
          this.y = y;
        }
        public void ChangeData(int x, int y){
                this.x = x;
```

```
            this.y = y;
        }
        public int getX(){
            return x;
        }
        public int getY(){
            return y;
        }
        public int getXf(){
            return xf;
        }
        public int getYf(){
            return yf;
        }


}
```

THE WINDOW OF THE GAME CODE IS:

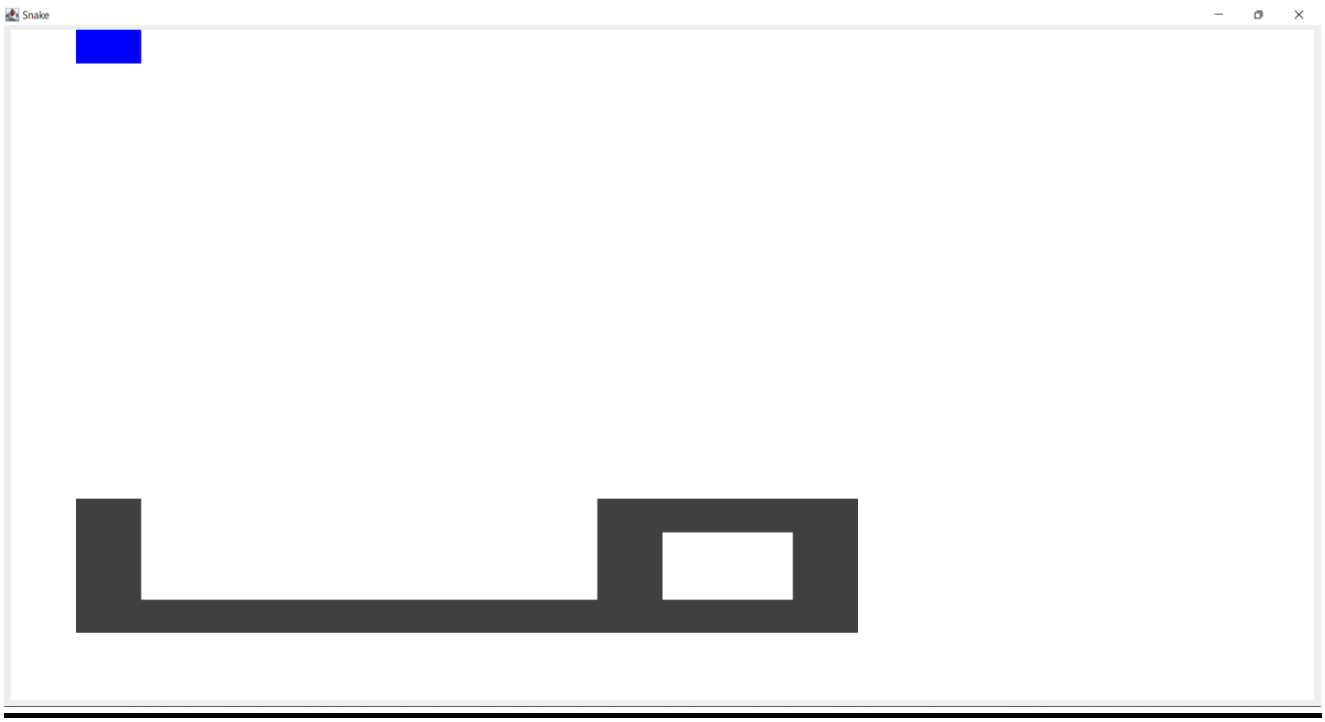
```
import java.awt.GridLayout;
import java.awt.event.KeyListener;
import java.util.ArrayList;

import javax.swing.JFrame;


class Window extends JFrame{
	private static final long serialVersionUID = -2542001418764869760L;
	public static ArrayList<ArrayList<DataOfSquare>> Grid;
	public static int width = 20;
	public static int height = 20;
	public Window(){
```

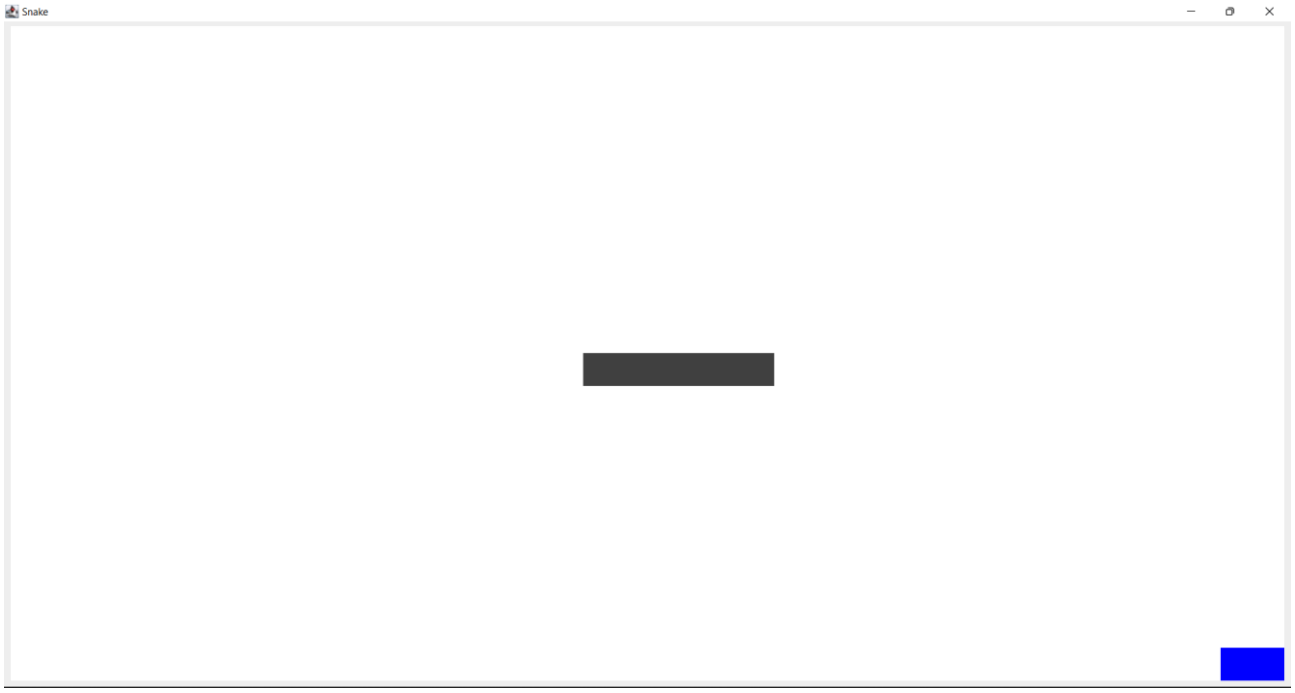
```
        // Creates the arraylist that'll contain the
threads
        Grid = new
ArrayList<ArrayList<DataOfSquare>>();
        ArrayList<DataOfSquare> data;

        // Creates Threads and its data and adds it to
the arrayList
        for(int i=0;i<width;i++){
            data= new ArrayList<DataOfSquare>();
            for(int j=0;j<height;j++){
                DataOfSquare c = new
DataOfSquare(2);
                data.add(c);
            }
            Grid.add(data);
        }
```

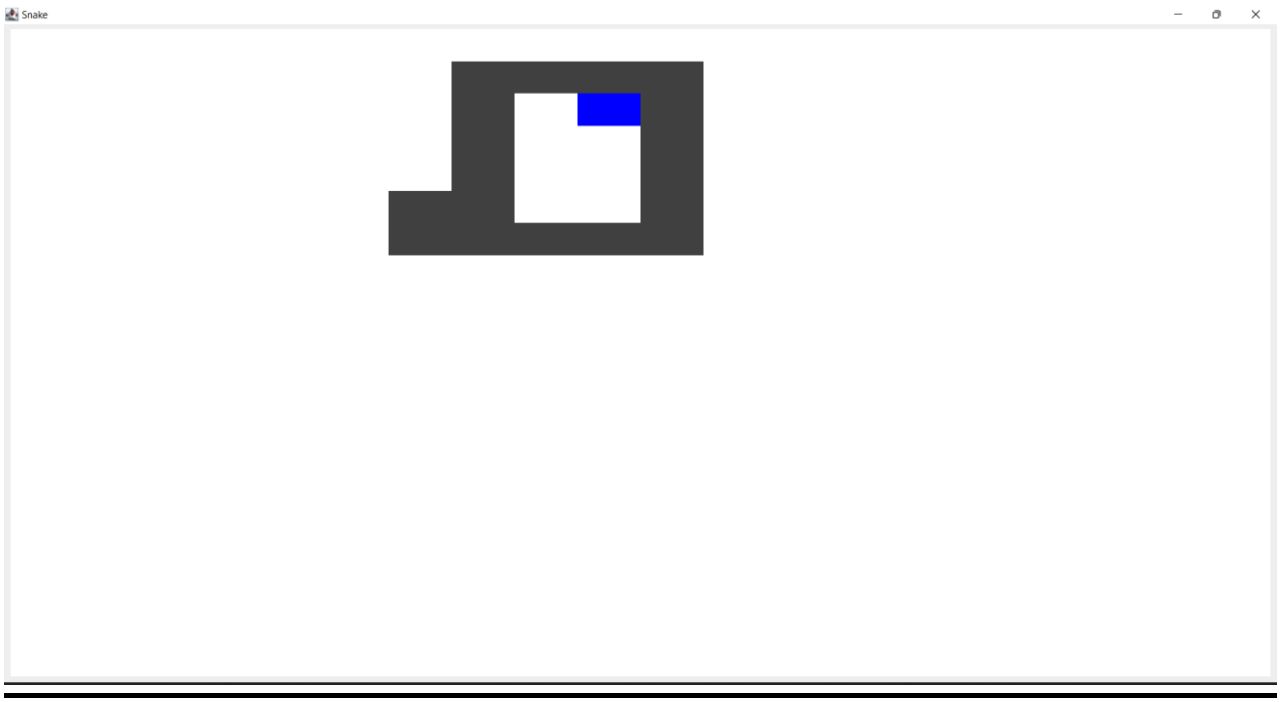
```
        // Setting up the layout of the panel
        getContentPane().setLayout(new
GridLayout(20,20,0,0));

        // Start & pauses all threads, then adds every
square of each thread to the panel
        for(int i=0;i<width;i++){
            for(int j=0;j<height;j++){

    getContentPane().add(Grid.get(i).get(j).square);
            }
        }

        // initial position of the snake
        Tuple position = new Tuple(10,10);
        // passing this value to the controller
        ThreadsController c = new
ThreadsController(position);
        //Let's start the game now..
```

```
            c.start();


            // Links the window to the
keyboardlistenner.
            this.addKeyListener((KeyListener) new
KeyboardListener());


            //To do : handle multiplayers .. The above
works, test it and see what happens


            //Tuple position2 = new Tuple(13,13);
            //ControlleurThreads c2 = new
ControlleurThreads(position2);
            //c2.start();


      }
}
```

OUTPUTS FROM THE GAME




Snake


Snake

FUTURE SCOPE OF THE PROJECT

1. This project is quite interesting and has a good future. This game is in market from almost 20 years but still the game holds strong position in market but this game has been made in JAVA programming language which makes this game quite light weighted, faster than before, better user interaction, and so on. This game can still be installed on various multimedia devices and televisions in order to give customers free game. This is very simple game and cheap too and can be easily be circulated in market.

2. The most eye-catching feature of this game is that it is non-addictive and can help small kids to increase their IQ which surely gives it a huge scope in market.

SUMMARY

Snake Master Game can be used widely because though it is a simple game but still it is highly beneficial for small kids as it will help in increasing their IQ. These games can be pre-installed on Television and Smartphones in order to give users free access to at least one game. We have used various libraries of Java in this project to make it user friendly and interesting for users.