# CUTM1031: Java Technologies


Centurion
UNIVERSITY

*Shaping Lives...*
*Empowering Communities!*

---

# PROJECT REPORT
# ON PAC-MAN GAME

~ A game for concentration development

**BY**
**JSK HEMA SHRI**
**201801330033**

---

**Faculty Signature** **Head of Department**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING
CENTURION UNIVERSITYOF TECHNOLOGY AND MANAGEMENT, VIZIANAGARAM, INDIA – 535003

## ABSTRACT:

This project discusses about the popular Pacman game using java. I built a simple Pacman implementation with the maze, a Pac-man and Pac-dots for the Pac-man to eat. For this, I've created a total of eight Pac-man images, one of which is displayed depending on direction with alternating open/closed mouth. The Pac-man moves around legal positions randomly and eats the Pac-dots. I added several ghosts. With checking to see if the game has ended (ghost catches Pacman or Pacman eats all of the dots) this second increment was completed. Although the most noticeable feature of the game is the graphics representation.

***Keywords-*** *Pac-man, attack, ghost, food, maze, human player, loop player, BFS-Finder, Pac-board, ghost data, image helper, move type, messages, powerup food, string helper, teleport tunnel, pac window.*

## INTRODUCTION:

The classic and enormously popular Pac-Man video game came out in Japan on May 21, 1980, and by October of that year it was released in the United States. The yellow, pie-shaped Pac-Man character, who travels around a maze trying to eat dots and avoid four hunting ghosts, quickly became an icon of the 1980s. When we talk about arcade games, without exception we say the Pac-man game. In recent research's it shows that playing Pac-man is able to develop concentration and decision-making speed. This project attempts to emulate human thought by to annotate Pac-man game Pac-man is a game in which Pac-man is an agent that eats the Pac-dots. Ghosts are agents that attack the Pac-man. The whole game is played in maze environment. In this project, I've designed agents for the classic version of Pac-man.

## DESCRIPTION:

Our version of Pac-Man will have several modifications to the original game. Enemy units will be configured to attack and defend human players. Our game is based upon a point system. In a one-person game, if the player gobbles up a designated number of pellets in the game, then the player wins. Our game will host a maximum of two human players competing head-to-head. When the game is in two player mode, a win condition is given to the highest scoring human player, regardless of AI effect. Additionally, our game state will contain "Magic Pellets" which will offer players special powers such as attack capability, invincibility …etc.

### Existing System:

- Less accurate
- Oldest game
- Can crash a lot of time
- Can't save game the game score
- Basic Graphics (Low quality)
- becomes boring overtime

### Proposed System:

- Accurate
- Flexibility productivity
- More graphics with modern consoles
- Addictive and fun
- More interactive and challenging

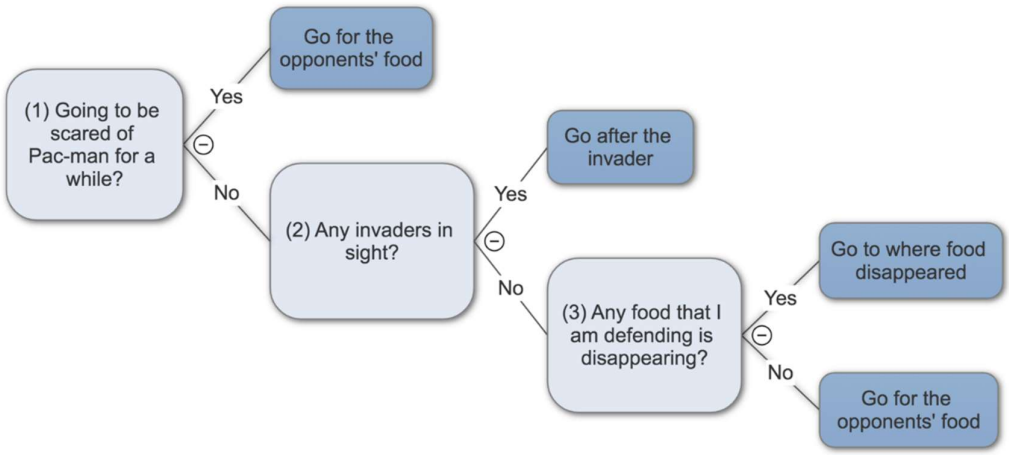
## Software Requirements:

- Windows 7, Windows Vista, Windows 10/11
- Linux (All Versions)
- Mac (All Versions)
- JDK 17
- Any java supported IDE's (This is made in IntelliJ IDEA)

## Hardware Requirements:

- 2 GB RAM minimum, 8GB RAM recommended
- 1 GB of available disk space minimum, 4 GB recommended (500 MB for IDE)
- **Processor:** Intel dual core i3
  - 1280 x 800 minimum screen resolution
  - Sound Card: DirectX sound device

## Block diagram:




## Project Code:

### Loop Player code:

```
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;

// Created by shri on 04/26/2022.
public class LoopPlayer {
   Clip clip;
   AudioInputStream inputStream;
```

```
    public LoopPlayer(String soundname){
        try {
            clip = AudioSystem.getClip();
            inputStream = AudioSystem.getAudioInputStream(
                    Main.class.getResourceAsStream("resources/sounds/"+
soundname));
            clip.open(inputStream);
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
    public void start(){
        try {
            clip.loop(Clip.LOOP_CONTINUOUSLY);
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
    public void stop(){
        try {
            clip.stop();
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
}
```

**Main file:**

```
public class Main {
    public static void main(String[] args) {
        new StartWindow();
    }
}
```

**Map Editor:**

```
import javax.swing.*;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.concurrent.ThreadLocalRandom;
public class MapEditor extends JFrame {
    public MapEditor(){
        setSize(650,400);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        getContentPane().setBackground(Color.black);
        JPanel sideBar = new JPanel();
```

```
        sideBar.setLayout(new BorderLayout());
        sideBar.setBackground(Color.black);
        JPanel ghostSelection = new JPanel();
        ghostSelection.setLayout(new
    BoxLayout(ghostSelection,BoxLayout.Y_AXIS));
        ghostSelection.setBackground(Color.black);
        JLabel l0 = new JLabel("= : Blank Space (without Food)");
        JLabel l1 = new JLabel("_ : Blank Space (with Food)");
        JLabel l2 = new JLabel("X : Wall");
        JLabel l3 = new JLabel("Y : Semi-Wall (Passable by Ghosts)");
        JLabel l4 = new JLabel("P : Pacman Start Position");
        JLabel l5 = new JLabel("1 : Red Ghost (Chaser)");
        JLabel l6 = new JLabel("2 : Pink Ghost (Traveler)");
        JLabel l7 = new JLabel("3 : Cyan Ghost (Patrol)");
        JLabel l8 = new JLabel("F : Fruit");
        JLabel l9 = new JLabel("B : Ghost Base");
        //JLabel l4 = new JLabel("1 : Red Ghost (Chaser)");
        l0.setForeground(Color.yellow);
        l1.setForeground(Color.yellow);
        l2.setForeground(Color.yellow);
        l3.setForeground(Color.yellow);
        l4.setForeground(Color.yellow);
        l5.setForeground(Color.yellow);
        l6.setForeground(Color.yellow);
        l7.setForeground(Color.yellow);
        l8.setForeground(Color.yellow);
        l9.setForeground(Color.yellow);
        ghostSelection.add(l0);
        ghostSelection.add(l1);
        ghostSelection.add(l2);
        ghostSelection.add(l3);
        ghostSelection.add(l4);
        ghostSelection.add(l5);
        ghostSelection.add(l6);
        ghostSelection.add(l7);
        ghostSelection.add(l8);
        ghostSelection.add(l9);
        setLayout(new BorderLayout());
        sideBar.add(ghostSelection,BorderLayout.NORTH);
        getContentPane().add(sideBar,BorderLayout.EAST);
        JTextArea ta = new JTextArea();
        ta.setBackground(Color.black);
        ta.setForeground(Color.yellow);
        ta.setText("XXXXXXXXXX\n"
              + "XP_______X\n"
              + "X________X\n"
              + "X________X\n"
              + "XXXXXXXXXX");
```

```
        ta.setBorder(new CompoundBorder(new CompoundBorder(new
EmptyBorder(20,10,20,10),new LineBorder(Color.yellow)),new
EmptyBorder(10,10,10,10)));
        getContentPane().add(ta);
        FansyButton startButton = new FansyButton("Start Game");
        startButton.addActionListener(new ActionListener() {
           @Override
           public void actionPerformed(ActionEvent e) {
              new PacWindow(compileMap(ta.getText()));
           }
        });
        sideBar.add(startButton,BorderLayout.SOUTH);
        //setLayout(new Grid);

        setVisible(true);
     }
     //Resolve Map
     public static MapData compileMap(String input){
        int mx = input.indexOf('\n');
        int my = StringHelper.countLines(input);
        System.out.println("Making Map "+mx+"x"+my);

        MapData customMap = new MapData(mx,my);
        customMap.setCustom(true);
        int[][] map = new int[mx][my];
        int i=0;
        int j=0;
        for(char c : input.toCharArray()){
           if(c == '1'){
              map[i][j] = 0;
              customMap.getGhostsData().add(new GhostData(i,j,ghostType.RED));
           }
           if(c == '2'){
              map[i][j] = 0;
              customMap.getGhostsData().add(new GhostData(i,j,ghostType.PINK));
           }
           if(c == '3'){
              map[i][j] = 0;
              customMap.getGhostsData().add(new
GhostData(i,j,ghostType.CYAN));
           }
           if(c == 'P'){
              map[i][j] = 0;
              customMap.setPacmanPosition(new Point(i,j));
           }
           if(c == 'X'){
              map[i][j] = 23;
           }
           if(c == 'Y'){
              map[i][j] = 26;
```

```
            }
            if(c == '_'){
               map[i][j] = 0;
               customMap.getFoodPositions().add(new Food(i,j));
            }
            if(c == '='){
               map[i][j] = 0;
            }
            if(c == 'O'){
               map[i][j] = 0;
               customMap.getPufoodPositions().add(new PowerUpFood(i,j,0));
            }
            if(c == 'F'){
               map[i][j] = 0;
               customMap.getPufoodPositions().add(new PowerUpFood(i,j,
ThreadLocalRandom.current().nextInt(4)+1));
            }
            if(c == 'B'){
               map[i][j] = 0;
               customMap.setGhostBasePosition(new Point(i,j));
            }
            i++;
            if(c == '\n'){
               j++;
               i=0;
            }
         }
         customMap.setMap(map);
         customMap.setCustom(true);
         System.out.println("Map Read OK !");
         return customMap;
         //new PacWindow(customMap);
      }
   }
```

**Pac Window:**

```
import javax.swing.*;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;
import java.awt.*;
import java.util.Map;
import java.util.Scanner;

public class PacWindow extends JFrame {

   public PacWindow(){
      setTitle("AKP Pacman v1.0");
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
      setLayout(new BorderLayout());
```

```
        getContentPane().setBackground(Color.black);

        setSize(794,884);
        setLocationRelativeTo(null);

        JLabel scoreboard = new JLabel("    Score : 0");
        scoreboard.setForeground(new Color(255, 243, 36));

        MapData map1 = getMapFromResource("resources/maps/map1_c.txt");
        adjustMap(map1);
        PacBoard pb = new PacBoard(scoreboard,map1,this);

        pb.setBorder(new CompoundBorder(new EmptyBorder(10,10,10,10),new
LineBorder(Color.BLUE)));
        addKeyListener(pb.pacman);

        this.getContentPane().add(scoreboard,BorderLayout.SOUTH);
        this.getContentPane().add(pb);
        setVisible(true);
    }

    public PacWindow(MapData md){
        setTitle("AKP Pacman v1.0");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());
        getContentPane().setBackground(Color.black);

        setSize(794,884);
        setLocationRelativeTo(null);

        JLabel scoreboard = new JLabel("    Score : 0");
        scoreboard.setForeground(new Color(255, 243, 36));

        //int[][] mapLoaded = loadMap(27,29,"/maps/map1.txt");
        adjustMap(md);
        PacBoard pb = new PacBoard(scoreboard,md,this);
        pb.setBorder(new CompoundBorder(new EmptyBorder(10,10,10,10),new
LineBorder(Color.BLUE)));
        addKeyListener(pb.pacman);

        this.getContentPane().add(scoreboard,BorderLayout.SOUTH);
        this.getContentPane().add(pb);
        setVisible(true);
    }
    public int[][] loadMap(int mx,int my,String relPath){
        try {
            Scanner scn = new
Scanner(this.getClass().getResourceAsStream(relPath));
            int[][] map;
            map = new int[mx][my];
```

```
            for(int y=0;y<my;y++){
               for(int x=0;x<mx;x++){
                  map[x][y]=scn.nextInt();
               }
            }
            return map;
         }catch(Exception e){
            System.err.println("Error Reading Map File !");
         }
         return null;
      }

      public MapData getMapFromResource(String relPath){
         String mapStr = "";
         try {
            Scanner scn = new
Scanner(this.getClass().getResourceAsStream(relPath));
            StringBuilder sb = new StringBuilder();
            String line;
            while(scn.hasNextLine()){
               line = scn.nextLine();
               sb.append(line).append('\n');
            }
            mapStr = sb.toString();
         }catch(Exception e){
            System.err.println("Error Reading Map File !");
         }
         if("".equals(mapStr)){
            System.err.println("Map is Empty !");
         }
         return MapEditor.compileMap(mapStr);
      }

      //Dynamically Generate Map Segments
      public void adjustMap(MapData mapd){
         int[][] map = mapd.getMap();
         int mx=mapd.getX();
         int my=mapd.getY();
         for(int y=0;y<my;y++){
            for(int x=0;x<mx;x++){
               boolean l = false;
               boolean r = false;
               boolean t = false;
               boolean b = false;
               boolean tl = false;
               boolean tr = false;
               boolean bl = false;
               boolean br = false;
                  if(map[x][y]>0 && map[x][y]<26) {
               int mustSet = 0;
```

```
            //LEFT
            if (x > 0 && map[x - 1][y] > 0 && map[x-1][y]<26) {
               l = true;
            }
            //RIGHT
            if (x < mx - 1 && map[x + 1][y] > 0 && map[x+1][y]<26) {
               r = true;
            }
            //TOP
            if (y > 0 && map[x][y - 1] > 0 && map[x][y-1]<26) {
               t = true;
            }
            //Bottom
            if (y < my - 1 && map[x][y + 1] > 0 && map[x][y+1]<26) {
               b = true;
            }
            //TOP LEFT
            if (x > 0 && y > 0 && map[x - 1][y - 1] > 0 && map[x-1][y-1]<26) {
               tl = true;
            }
            //TOP RIGHT
            if (x < mx - 1 && y > 0 && map[x + 1][y - 1] > 0 && map[x+1][y-
1]<26) {
               tr = true;
            }
            //Bottom LEFT
            if (x > 0 && y < my - 1 && map[x - 1][y + 1] > 0 && map[x-1][y+1]<26)
{
               bl = true;
            }
            //Bottom RIGHT
            if (x < mx - 1 && y < my - 1 && map[x + 1][y + 1] > 0 &&
map[x+1][y+1]<26) {
               br = true;
            }

            //Decide Image to View
            if (!r && !l && !t && !b) {
               mustSet = 23;
            }
            if (r && !l && !t && !b) {
               mustSet = 22;
            }
            if (!r && l && !t && !b) {
               mustSet = 25;
            }
            if (!r && !l && t && !b) {
               mustSet = 21;
            }
            if (!r && !l && !t && b) {
```

```
    mustSet = 19;
}
if (r && l && !t && !b) {
    mustSet = 24;
}
if (!r && !l && t && b) {
    mustSet = 20;
}
if (r && !l && t && !b && !tr) {
    mustSet = 11;
}
if (r && !l && t && !b && tr) {
    mustSet = 2;
}
if (!r && l && t && !b && !tl) {
    mustSet = 12;
}
if (!r && l && t && !b && tl) {
    mustSet = 3;
}
if (r && !l && !t && b && br) {
    mustSet = 1;
}
if (r && !l && !t && b && !br) {
    mustSet = 10;
}
if (!r && l && !t && b && bl) {
    mustSet = 4;
}
if (r && !l && t && b && !tr) {
    mustSet = 15;
}
if (r && !l && t && b && tr) {
    mustSet = 6;
}
if (!r && l && t && b && !tl) {
    mustSet = 17;
}
if (!r && l && t && b && tl) {
    mustSet = 8;
}
if (r && l && !t && b && !br) {
    mustSet = 14;
}
if (r && l && !t && b && br) {
    mustSet = 5;
}
if (r && l && t && !b && !tr) {
    mustSet = 16;
}
```

```
                if (r && l && t && !b && tr) {
                    mustSet = 7;
                }
                if (!r && l && !t && b && !bl) {
                    mustSet = 13;
                }
                if (r && l && t && b && br && tl) {
                    mustSet = 9;
                }
                if (r && l && t && b && !br && !tl) {
                    mustSet = 18;
                }

                //System.out.println("MAP SEGMENT : " + mustSet);
                map[x][y] = mustSet;
            }
            mapd.setMap(map);
        }
    }
    System.out.println("Map Adjust OK !");
  }
}
```

Red ghost:

```
import javax.imageio.ImageIO;
import java.awt.*;
import java.io.IOException;
import java.util.ArrayList;
import java.util.concurrent.ThreadLocalRandom;

public class RedGhost extends Ghost {

  BFSFinder bfs;

  public RedGhost(int x, int y,PacBoard pb){
    super(x,y,pb,12);
  }

  @Override
  public void loadImages(){
    ghostR = new Image[2];
    ghostL = new Image[2];
    ghostU = new Image[2];
    ghostD = new Image[2];
    try {
      ghostR[0]                                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/red/1.png"));
      ghostR[1]                                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/red/3.png"));
```

```
        ghostL[0] =
ImageHelper.flipHor(ImageIO.read(this.getClass().getResource("resources/imag
es/ghost/red/1.png")));
        ghostL[1] =
ImageHelper.flipHor(ImageIO.read(this.getClass().getResource("resources/imag
es/ghost/red/3.png")));
        ghostU[0] =
ImageIO.read(this.getClass().getResource("resources/images/ghost/red/4.png"));
        ghostU[1] =
ImageIO.read(this.getClass().getResource("resources/images/ghost/red/5.png"));
        ghostD[0] =
ImageIO.read(this.getClass().getResource("resources/images/ghost/red/6.png"));
        ghostD[1] =
ImageIO.read(this.getClass().getResource("resources/images/ghost/red/7.png"));
      }catch(IOException e){
        System.err.println("Cannot Read Images !");
      }
    }

    moveType pendMove = moveType.UP;

    //find closest path using BFS
    @Override
    public moveType getMoveAI(){
      if(isPending){
        if(isStuck){
          if(pendMove == moveType.UP){
            pendMove = moveType.DOWN;
          }else if(pendMove == moveType.DOWN){
            pendMove = moveType.UP;
          }
          return pendMove;
        }else{
          return pendMove;
        }
      }
      if(bfs==null)
        bfs = new BFSFinder(parentBoard);
      if(isDead) {
        return baseReturner.getMove(logicalPosition.x,logicalPosition.y,
parentBoard.ghostBase.x,parentBoard.ghostBase.y);
      }else{
        return
bfs.getMove(logicalPosition.x,logicalPosition.y,parentBoard.pacman.logicalPositi
on.x,parentBoard.pacman.logicalPosition.y);
      }
    }
}
```

**Pink ghost:**

```
import javax.imageio.ImageIO;
import java.awt.*;
import java.io.IOException;
import java.util.ArrayList;
import java.util.concurrent.ThreadLocalRandom;


public class PinkGhost extends Ghost {

   public PinkGhost(int x, int y,PacBoard pb){
      super(x,y,pb,6);
   }

   @Override
   public void loadImages(){
      ghostR = new Image[2];
      ghostL = new Image[2];
      ghostU = new Image[2];
      ghostD = new Image[2];
      try {
         ghostR[0]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/pink/1.png")
);
         ghostR[1]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/pink/3.png")
);
         ghostL[0]                                                    =
ImageHelper.flipHor(ImageIO.read(this.getClass().getResource("resources/imag
es/ghost/pink/1.png")));
         ghostL[1]                                                    =
ImageHelper.flipHor(ImageIO.read(this.getClass().getResource("resources/imag
es/ghost/pink/3.png")));
         ghostU[0]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/pink/4.png")
);
         ghostU[1]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/pink/5.png")
);
         ghostD[0]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/pink/6.png")
);
         ghostD[1]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/pink/7.png")
);
      }catch(IOException e){
         System.err.println("Cannot Read Images !");
      }
   }

   moveType lastCMove;
```

```
    moveType pendMove = moveType.UP;

    @Override
    public moveType getMoveAI(){
        if(isPending){
            if(isStuck){
                if(pendMove == moveType.UP){
                    pendMove = moveType.DOWN;
                }else if(pendMove == moveType.DOWN){
                    pendMove = moveType.UP;
                }
                return pendMove;
            }else{
                return pendMove;
            }
        }
        if(isDead) {
            return              baseReturner.getMove(logicalPosition.x,logicalPosition.y,
parentBoard.ghostBase.x,parentBoard.ghostBase.y);
        }else {
            if (lastCMove == null || isStuck) {
                ArrayList<moveType> pm = getPossibleMoves();
                int i = ThreadLocalRandom.current().nextInt(pm.size());
                lastCMove = pm.get(i);
                return lastCMove;
            } else {
                return lastCMove;
            }
        }
    }
}
```

Ghost data:

```
public class GhostData {
    private int x;
    private int y;
    private ghostType type;

    public GhostData(int x,int y,ghostType type){
        this.x = x;
        this.y = y;
        this.type = type;
    }

    public int getX() {
        return x;
    }

    public void setX(int x) {
```

```
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public ghostType getType() {
        return type;
    }

    public void setType(ghostType type) {
        this.type = type;
    }
}
```

**Buttons:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.io.IOException;
public class FansyButton extends JLabel implements MouseListener {

    ActionListener myAL;

    public FansyButton(String str){
        super(str);
        Font customFont;
        try {
            customFont = Font.createFont(Font.TRUETYPE_FONT,
this.getClass().getResourceAsStream("resources/fonts/crackman.ttf")).deriveFo
nt(30f);
            this.setFont(customFont);
        } catch (FontFormatException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        this.setForeground(Color.yellow);
        this.setOpaque(false);
        this.addMouseListener(this);
    }
```

```
    public void addActionListener(ActionListener al){
        myAL = al;
    }

    @Override
    public void mouseClicked(MouseEvent e) {
        myAL.actionPerformed(new ActionEvent(this,501,""));
    }

    @Override
    public void mousePressed(MouseEvent e) {

    }

    @Override
    public void mouseReleased(MouseEvent e) {

    }

    @Override
    public void mouseEntered(MouseEvent e) {
        this.setForeground(new Color(243, 105, 66));
    }

    @Override
    public void mouseExited(MouseEvent e) {
        this.setForeground(Color.yellow);
    }
}
```

**Teleport Tunnel:**

```
import java.awt.*;

public class TeleportTunnel {

    private Point from;
    private Point to;
    private moveType reqMove;

    public Point getTo() {
        return to;
    }

    public void setTo(Point to) {
        this.to = to;
    }

    public Point getFrom() {
        return from;
    }
```

```
    public void setFrom(Point from) {
        this.from = from;
    }

    public moveType getReqMove() {
        return reqMove;
    }

    public void setReqMove(moveType reqMove) {
        this.reqMove = reqMove;
    }

    public TeleportTunnel(int x1,int y1,int x2,int y2,moveType reqMove){
        from = new Point(x1,y1);
        to = new Point(x2,y2);
        this.reqMove = reqMove;
    }
}
```

**Ghost:**

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.util.ArrayList;
import java.util.concurrent.ThreadLocalRandom;


public abstract class Ghost {
    //Anim Vars
    Timer animTimer;
    ActionListener animAL;

    //Pending Vars
    Timer pendingTimer;
    ActionListener pendingAL;

    //Move Vars
    Timer moveTimer;
    ActionListener moveAL;
    public moveType activeMove;
    protected boolean isStuck = true;
    boolean isPending = false;

    Timer unWeakenTimer1;
    Timer unWeakenTimer2;
    ActionListener unweak1;
    ActionListener unweak2;
```

```
int unweakBlinks;
boolean isWhite = false;

protected boolean isWeak = false;
protected boolean isDead = false;

public boolean isWeak() {
   return isWeak;
}

public boolean isDead() {
   return isDead;
}

//Image[] pac;
Image ghostImg;
int activeImage = 0;
int addFactor = 1;

public Point pixelPosition;
public Point logicalPosition;

Image[] ghostR;
Image[] ghostL;
Image[] ghostU;
Image[] ghostD;

Image[] ghostW;
Image[] ghostWW;
Image ghostEye;

int ghostNormalDelay;
int ghostWeakDelay = 30;
int ghostDeadDelay = 5;

BFSFinder baseReturner;

protected PacBoard parentBoard;

public Ghost (int x, int y,PacBoard pb,int ghostDelay) {

   logicalPosition = new Point(x,y);
   pixelPosition = new Point(28*x,28*y);

   parentBoard = pb;

   activeMove = moveType.RIGHT;

   ghostNormalDelay = ghostDelay;
```

```
        loadImages();

        //load weak Image
        ghostW = new Image[2];
        try {
            ghostW[0]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/blue/1.png"))
;
            ghostW[1]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/ghost/blue/3.png"))
;
        } catch (IOException e) {
            e.printStackTrace();
        }

        ghostWW = new Image[2];
        try {
            ghostWW[0]                                                   =
ImageIO.read(this.getClass().getResource("resources/images/ghost/white/1.png"
));
            ghostWW[1]                                                   =
ImageIO.read(this.getClass().getResource("resources/images/ghost/white/3.png"
));
        } catch (IOException e) {
            e.printStackTrace();
        }

        try {
            ghostEye                                                     =
ImageIO.read(this.getClass().getResource("resources/images/eye.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }

        //animation timer
        animAL = new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                activeImage = (activeImage + 1) % 2;
            }
        };
        animTimer = new Timer(100,animAL);
        animTimer.start();

        moveAL = new ActionListener() {
            public void actionPerformed(ActionEvent evt) {

                if((pixelPosition.x % 28 == 0) && (pixelPosition.y % 28 == 0)){
                    if(!isStuck) {
                        switch (activeMove) {
                            case RIGHT:
```

```
                    logicalPosition.x++;
                    break;
                case LEFT:
                    logicalPosition.x--;
                    break;
                case UP:
                    logicalPosition.y--;
                    break;
                case DOWN:
                    logicalPosition.y++;
                    break;
            }
            parentBoard.dispatchEvent(new
ActionEvent(this,Messeges.UPDATE,null));
        }


        activeMove = getMoveAI();
        isStuck = true;

        //animTimer.stop();
        //System.out.println("LOGICAL POS :" + logicalPosition.x + " , " +
logicalPosition.y);
        //if(todoMove != moveType.NONE) {
        //    activeMove = todoMove;
        //    todoMove = moveType.NONE;
        //}
      }else{
        isStuck = false;
        //animTimer.start();
      }
      // }
      //TODO : fix ghost movements
      switch(activeMove){
        case RIGHT:
          if(pixelPosition.x >= (parentBoard.m_x-1) * 28){
            return;
          }
          if((logicalPosition.x+1 < parentBoard.m_x) &&
(parentBoard.map[logicalPosition.x+1][logicalPosition.y]>0) &&
((parentBoard.map[logicalPosition.x+1][logicalPosition.y]<26)||isPending)){
            return;
          }
          pixelPosition.x ++;
          break;
        case LEFT:
          if(pixelPosition.x <= 0){
            return;
          }
```

```
                if((logicalPosition.x-1 >= 0) &&
(parentBoard.map[logicalPosition.x-1][logicalPosition.y]>0) &&
((parentBoard.map[logicalPosition.x-1][logicalPosition.y]<26)||isPending)){
                    return;
                }
                pixelPosition.x --;
                break;
             case UP:
                if(pixelPosition.y <= 0){
                    return;
                }
                if((logicalPosition.y-1 >= 0) &&
(parentBoard.map[logicalPosition.x][logicalPosition.y-1]>0) &&
((parentBoard.map[logicalPosition.x][logicalPosition.y-1]<26)||isPending)){
                    return;
                }
                pixelPosition.y--;
                break;
             case DOWN:
                if(pixelPosition.y >= (parentBoard.m_y-1) * 28){
                    return;
                }
                if((logicalPosition.y+1 < parentBoard.m_y) &&
(parentBoard.map[logicalPosition.x][logicalPosition.y+1]>0) &&
((parentBoard.map[logicalPosition.x][logicalPosition.y+1]<26)||isPending)){
                    return;
                }
                pixelPosition.y ++;
                break;
          }

          parentBoard.dispatchEvent(new
ActionEvent(this,Messeges.COLTEST,null));
       }
    };
    moveTimer = new Timer(ghostDelay,moveAL);
    moveTimer.start();

    unweak1 = new ActionListener() {
       @Override
       public void actionPerformed(ActionEvent e) {
          unWeakenTimer2.start();
          unWeakenTimer1.stop();
       }
    };
    unWeakenTimer1 = new Timer(7000,unweak1);

    unweak2 = new ActionListener() {
       @Override
       public void actionPerformed(ActionEvent e) {
```

```
            if(unweakBlinks == 10){
               unweaken();
               unWeakenTimer2.stop();
            }
            if(unweakBlinks % 2 == 0){
               isWhite = true;
            }else{
               isWhite = false;
            }
            unweakBlinks++;
         }
      };
      unWeakenTimer2 = new Timer(250,unweak2);


      pendingAL = new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            isPending = false;
            pendingTimer.stop();
         }
      };
      pendingTimer = new Timer(7000,pendingAL);

      baseReturner = new BFSFinder(pb);
      //start AI
      activeMove = getMoveAI();

   }

   //load Images from Resource
   public abstract void loadImages();

   //get Move Based on AI
   public abstract moveType getMoveAI();

   //get possible Moves
   public ArrayList<moveType> getPossibleMoves(){
      ArrayList<moveType> possibleMoves = new ArrayList<>();

      if(logicalPosition.x >= 0 && logicalPosition.x < parentBoard.m_x-1 &&
logicalPosition.y >= 0 && logicalPosition.y < parentBoard.m_y-1 ) {
         //System.out.println(this.toString());
         if (!(parentBoard.map[logicalPosition.x + 1][logicalPosition.y] > 0)) {
            possibleMoves.add(moveType.RIGHT);
         }

         if (!(parentBoard.map[logicalPosition.x - 1][logicalPosition.y] > 0)) {
            possibleMoves.add(moveType.LEFT);
         }
```

```
        if(!(parentBoard.map[logicalPosition.x][logicalPosition.y-1]>0)){
            possibleMoves.add(moveType.UP);
        }

        if(!(parentBoard.map[logicalPosition.x][logicalPosition.y+1]>0)){
            possibleMoves.add(moveType.DOWN);
        }
    }

    return possibleMoves;
}

public Image getGhostImage(){
    if(!isDead) {
        if (!isWeak) {
            switch (activeMove) {
                case RIGHT:
                    return ghostR[activeImage];
                case LEFT:
                    return ghostL[activeImage];
                case UP:
                    return ghostU[activeImage];
                case DOWN:
                    return ghostD[activeImage];
            }
            return ghostR[activeImage];
        } else {
            if (isWhite) {
                return ghostWW[activeImage];
            } else {
                return ghostW[activeImage];
            }
        }
    }else{
        return ghostEye;
    }
}


public void weaken(){
    isWeak = true;
    moveTimer.setDelay(ghostWeakDelay);
    unweakBlinks = 0;
    isWhite = false;
    unWeakenTimer1.start();
}

public void unweaken(){
    isWeak = false;
```

```
        moveTimer.setDelay(ghostNormalDelay);
    }

    public void die(){
        isDead = true;
        moveTimer.setDelay(ghostDeadDelay);
    }

    public void undie(){
        //Shift Left Or Right
        int r = ThreadLocalRandom.current().nextInt(3);
        if (r == 0) {
            //Do nothing
        }
        if(r==1){
            logicalPosition.x += 1;
            pixelPosition.x += 28;
        }
        if(r==2){
            logicalPosition.x -= 1;
            pixelPosition.x -= 28;
        }
        isPending = true;
        pendingTimer.start();

        isDead = false;
        isWeak = false;
        moveTimer.setDelay(ghostNormalDelay);
    }
}
Start window:
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;

public class StartWindow extends JFrame {

    public StartWindow(){
        setSize(600,300);
        getContentPane().setBackground(Color.black);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        ImageIcon logo = new ImageIcon();
        try {
```

```
        logo = new
ImageIcon(ImageIO.read(this.getClass().getResource("/resources/images/pacma
n_logo.png")));
      } catch (IOException e) {
         e.printStackTrace();
      }

      //Register Custom fonts
      try {
         GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
         ge.registerFont(Font.createFont(Font.TRUETYPE_FONT,
this.getClass().getResourceAsStream("/resources/fonts/crackman.ttf")));
      } catch (IOException|FontFormatException e) {
         e.printStackTrace();
      }

      setLayout(new BorderLayout());
      getContentPane().add(new JLabel(logo),BorderLayout.NORTH);

      JPanel buttonsC = new JPanel();
      buttonsC.setBackground(Color.black);
      //buttonsC.setLayout(new FlowLayout(FlowLayout.LEADING,20,10));
      buttonsC.setLayout(new BoxLayout(buttonsC,BoxLayout.Y_AXIS));
      FansyButton startButton = new FansyButton("Start Game");
      FansyButton customButton = new FansyButton("Customize Game");

      startButton.setAlignmentX(Component.CENTER_ALIGNMENT);
      customButton.setAlignmentX(Component.CENTER_ALIGNMENT);

      startButton.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            PacWindow pw = new PacWindow();
            //new PacWindow();
            dispose();
         }
      });

      customButton.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            MapEditor me = new MapEditor();
            dispose();
         }
      });

      buttonsC.add(startButton);
      buttonsC.add(customButton);
```

```
        getContentPane().add(buttonsC);

        System.out.print('\n');
        System.out.println("PacMan v1.0   Developed By : Armin Kazemi");
        System.out.println("----------------------------------------");
        setVisible(true);
    }
}
```

**Image helper:**

```
import java.awt.*;
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;

public class ImageHelper {

    public static Image rotate90(Image i) {
        BufferedImage bi = (BufferedImage)i;
        AffineTransform tx = new AffineTransform();
        tx.rotate(0.5*Math.PI, bi.getWidth() / 2, bi.getHeight() / 2);
        AffineTransformOp op = new
AffineTransformOp(tx,AffineTransformOp.TYPE_BILINEAR);
        return op.filter(bi, null);
    }

    public static Image flipHor(Image i){
        BufferedImage bi = (BufferedImage)i;
        AffineTransform tx = AffineTransform.getScaleInstance(-1, 1);
        tx.translate(-i.getWidth(null), 0);
        AffineTransformOp op = new AffineTransformOp(tx,
AffineTransformOp.TYPE_NEAREST_NEIGHBOR);
        return op.filter(bi, null);
    }

    public static Image flipVer(Image i){
        BufferedImage bi = (BufferedImage)i;
        AffineTransform tx = AffineTransform.getScaleInstance(1, -1);
        tx.translate(0,-i.getWidth(null));
        AffineTransformOp op = new AffineTransformOp(tx,
AffineTransformOp.TYPE_NEAREST_NEIGHBOR);
        return op.filter(bi, null);
    }

}
```

**Animation:**

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.io.IOException;
import java.util.ArrayList;

public class Pacman implements KeyListener{

   //Move Vars
   Timer moveTimer;
   ActionListener moveAL;
   public moveType activeMove;
   moveType todoMove;
   boolean isStuck = true;

   //Animation Vars
   Timer animTimer;
   ActionListener animAL;
   Image[] pac;
   int activeImage = 0;
   int addFactor = 1;

   public Point pixelPosition;
   public Point logicalPosition;

   private PacBoard parentBoard;


   public Pacman (int x, int y,PacBoard pb) {

      logicalPosition = new Point(x,y);
      pixelPosition = new Point(28*x,28*y);

      parentBoard = pb;

      pac = new Image[5];

      activeMove = moveType.NONE;
      todoMove = moveType.NONE;

      try {
         pac[0]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/pac/pac0.png"));
         pac[1]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/pac/pac1.png"));
         pac[2]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/pac/pac2.png"));
         pac[3]                                                    =
ImageIO.read(this.getClass().getResource("resources/images/pac/pac3.png"));
```

```
            pac[4]                                                                    =
ImageIO.read(this.getClass().getResource("resources/images/pac/pac4.png"));
        }catch(IOException e){
            System.err.println("Cannot Read Images !");
        }

        //animation timer
        animAL = new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                activeImage = activeImage + addFactor;
                if(activeImage==4 || activeImage==0){
                    addFactor *= -1;
                }
            }
        };
        animTimer = new Timer(40,animAL);
        animTimer.start();


        moveAL = new ActionListener() {
            public void actionPerformed(ActionEvent evt) {

                //update logical position
                if((pixelPosition.x % 28 == 0) && (pixelPosition.y % 28 == 0)){
                    if(!isStuck) {
                        switch (activeMove) {
                            case RIGHT:
                                logicalPosition.x++;
                                break;
                            case LEFT:
                                logicalPosition.x--;
                                break;
                            case UP:
                                logicalPosition.y--;
                                break;
                            case DOWN:
                                logicalPosition.y++;
                                break;
                        }
                        //send update message
                        parentBoard.dispatchEvent(new
ActionEvent(this,Messeges.UPDATE,null));
                    }
                    isStuck = true;
                    animTimer.stop();

                    if(todoMove != moveType.NONE && isPossibleMove(todoMove) ) {
                        activeMove = todoMove;
                        todoMove = moveType.NONE;
                    }
```

```
            }else{
               isStuck = false;
               animTimer.start();
            }

            switch(activeMove){
               case RIGHT:
                  if((pixelPosition.x >= (parentBoard.m_x-1) *
28)&&parentBoard.isCustom){
                     return;
                  }
                  if(logicalPosition.x >= 0 && logicalPosition.x < parentBoard.m_x-1
&& logicalPosition.y >= 0 && logicalPosition.y < parentBoard.m_y-1 ) {
                     if (parentBoard.map[logicalPosition.x + 1][logicalPosition.y] > 0)
{
                        return;
                     }
                  }
                  pixelPosition.x ++;
                  break;
               case LEFT:
                  if((pixelPosition.x <= 0)&&parentBoard.isCustom){
                     return;
                  }
                   if(logicalPosition.x > 0 && logicalPosition.x < parentBoard.m_x-1
&& logicalPosition.y >= 0 && logicalPosition.y < parentBoard.m_y-1 ) {
                     if (parentBoard.map[logicalPosition.x - 1][logicalPosition.y] > 0)
{
                        return;
                     }
                  }
                  pixelPosition.x--;
                  break;
               case UP:
                  if((pixelPosition.y <= 0)&&parentBoard.isCustom){
                     return;
                  }
                  if(logicalPosition.x >= 0 && logicalPosition.x < parentBoard.m_x-1
&& logicalPosition.y >= 0 && logicalPosition.y < parentBoard.m_y-1 ) {
                     if(parentBoard.map[logicalPosition.x][logicalPosition.y-1]>0){
                        return;
                     }
                  }
                  pixelPosition.y--;
                  break;
               case DOWN:
                  if((pixelPosition.y >= (parentBoard.m_y-1) *
28)&&parentBoard.isCustom){
                     return;
                  }
```

```
                if(logicalPosition.x >= 0 && logicalPosition.x < parentBoard.m_x-1
&& logicalPosition.y >= 0 && logicalPosition.y < parentBoard.m_y-1 ) {
                    if(parentBoard.map[logicalPosition.x][logicalPosition.y+1]>0){
                        return;
                    }
                }
                pixelPosition.y ++;
                break;
            }
            parentBoard.dispatchEvent(new
ActionEvent(this,Messeges.COLTEST,null));

        }
      };
      moveTimer = new Timer(9,moveAL);
      moveTimer.start();
   }
   public boolean isPossibleMove(moveType move){
      if(logicalPosition.x >= 0 && logicalPosition.x < parentBoard.m_x-1 &&
logicalPosition.y >= 0 && logicalPosition.y < parentBoard.m_y-1 ) {
         switch(move){
            case RIGHT:
               return !(parentBoard.map[logicalPosition.x + 1][logicalPosition.y] >
0);
            case LEFT:
               return !(parentBoard.map[logicalPosition.x - 1][logicalPosition.y] >
0);
            case UP:
               return !(parentBoard.map[logicalPosition.x][logicalPosition.y - 1] >
0);
            case DOWN:
               return  !(parentBoard.map[logicalPosition.x][logicalPosition.y+1]  >
0);
         }
      }
      return false;
   }
   public Image getPacmanImage(){
      return pac[activeImage];
   }
   @Override
   public void keyReleased(KeyEvent ke){
   }
   @Override
   public void keyTyped(KeyEvent ke){
   }
   @Override
   public void keyPressed(KeyEvent ke){
      switch(ke.getKeyCode()){
         case 37:
```

```
            todoMove = moveType.LEFT;
            break;
         case 38:
            todoMove = moveType.UP;
            break;
         case 39:
            todoMove = moveType.RIGHT;
            break;
         case 40:
            todoMove = moveType.DOWN;
            break;
         case 82:
            parentBoard.dispatchEvent(new
ActionEvent(this,Messeges.RESET,null));
            break;
      }
      //System.out.println(ke.getKeyCode());
   }
}
```

**Food:**

```
import java.awt.*;
public class Food {

   public Point position;

   public Food(int x,int y){
      position = new Point(x,y);
   }
}
```

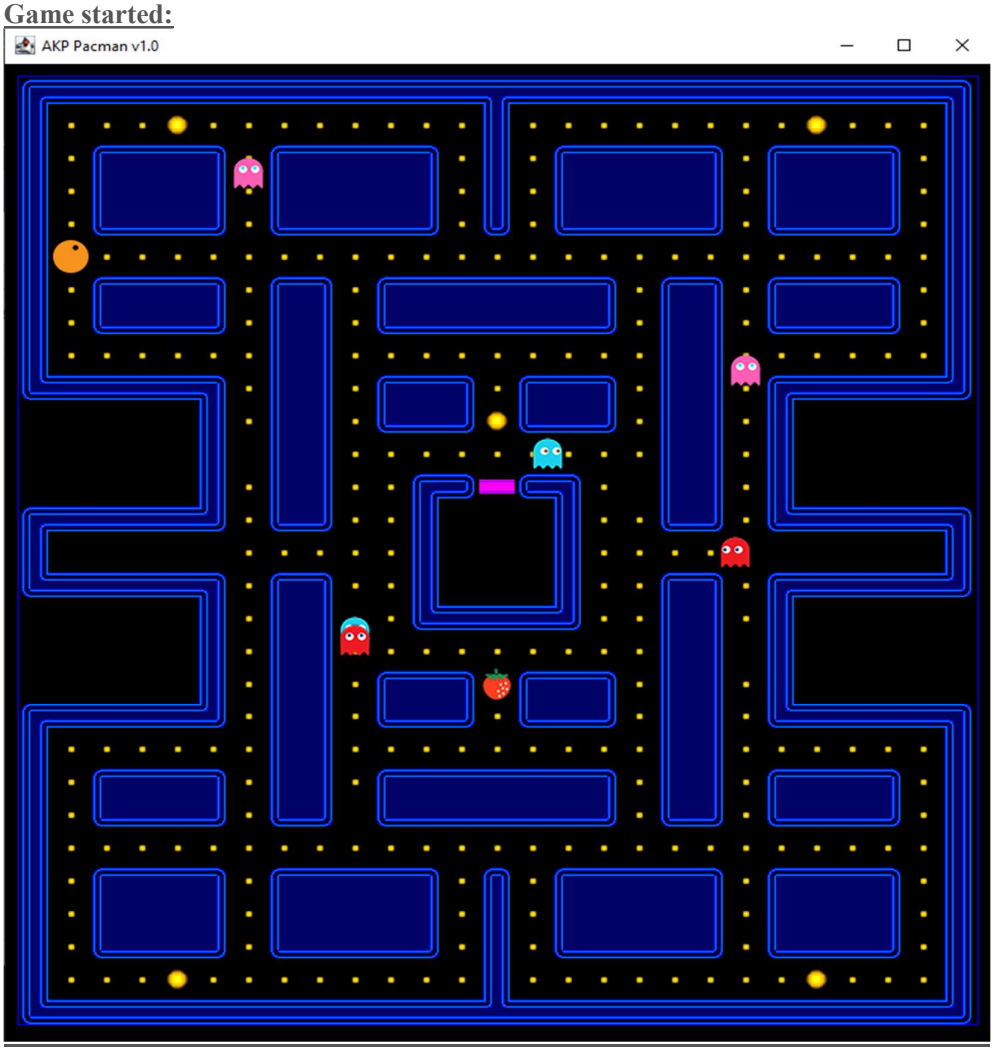
**String helper:**

```
public class StringHelper {
   public static int countLines(String str){
      String[] lines = str.split("\r\n|\r|\n");
      return  lines.length;
   }
}
```

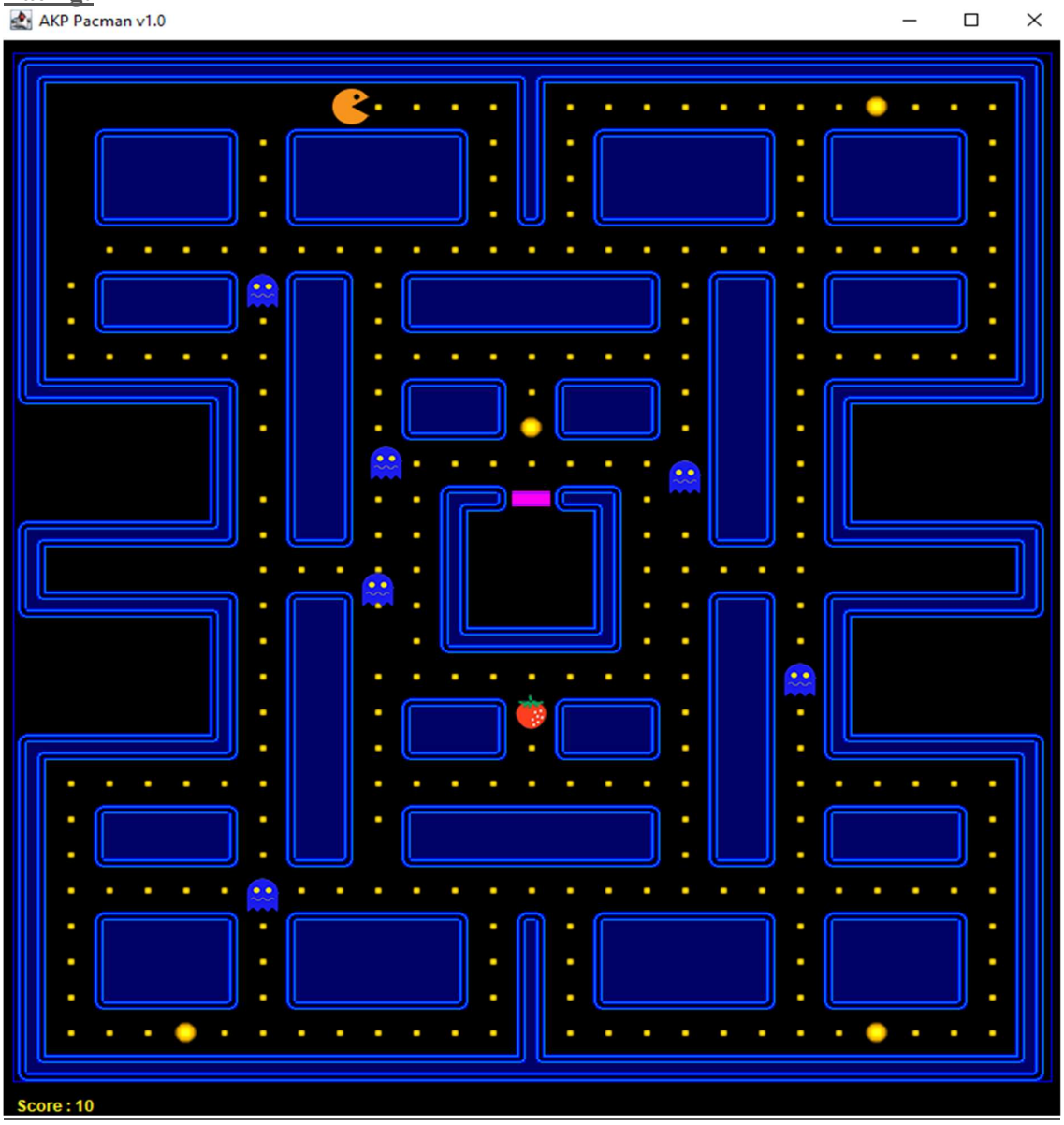
## Output Screens:

**Home Screen:**


**Game started:**



## Eating:




## Game Controls:

⬆ Or num-pad 8 => to move up

⬇ Or num-pad 2 => to move down

⬅ Or num-pad 4 => to move left

➡ Or num-pad 8 => to move up

If the pac-man eats a magic-pallets (point), the ghosts will slow down, So that we can capture them and we finish all the dots and foot. After a few seconds the ghosts will be released and they will haunt the pac-man.

**Game over:**


## GOALS AND OBJECTIVES:

The main goal of pac-man is to eat all the dots. It may seem easy, but it is not as simple as it seems. The task is made difficult because of the ghosts. They are trying to catch pac-man, and when one of them touches the Pac-man, the game is over.

## Bibliography:

Java Programming Advanced Topics Third Edition - Joe Wigglesworth & Paula McMillan (IBM Toronto Software Lab)

## References:

- www.stackoverflow.com
- www.wikipedia.com
- www.tutorialpoint.com

## Conclusion:

The player who succeeds in eating more dots by avoiding ghosts scores much points. The code I wrote is for desktop application. This can be extended and can be used in mobile applications so that it will be very flexible for the user to play the game. The project which I undertaken has helped me gain a better perspective on various aspects related to my course of study as well as particular knowledge of particular web-based applications.