INTERNATIONAL UNIVERSITY – VNU HCMC

FINAL EXAMINATION (2024-2025 S1)

Object-Oriented Programing

ESCAPE DUNGEON

Student name: [Mai Văn Vinh-ITITDK22117]
[Ngô Tùng Chương-ITITDK22116]
[Nguyễn Tiến Anh-ITITDK22128]
[Nguyễn Đăng Minh-ITITIU22103]

Test date: [23 December 2024]

TABLE OF CONTENT

RUN DEMO

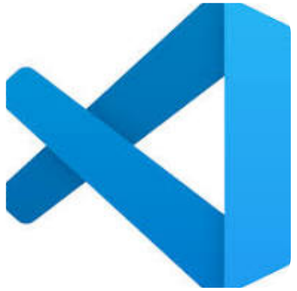
2. INTRODUCTION

- Better understanding of Game Development concepts.
- Utilizing Object-Oriented Programming principles.

2. INTRODUCTION

- Providing hands-on experience in the processes of developing, managing.

TOOL USED



VISUAL STUDIO CODE

ECLIPSE

YOUTUBE

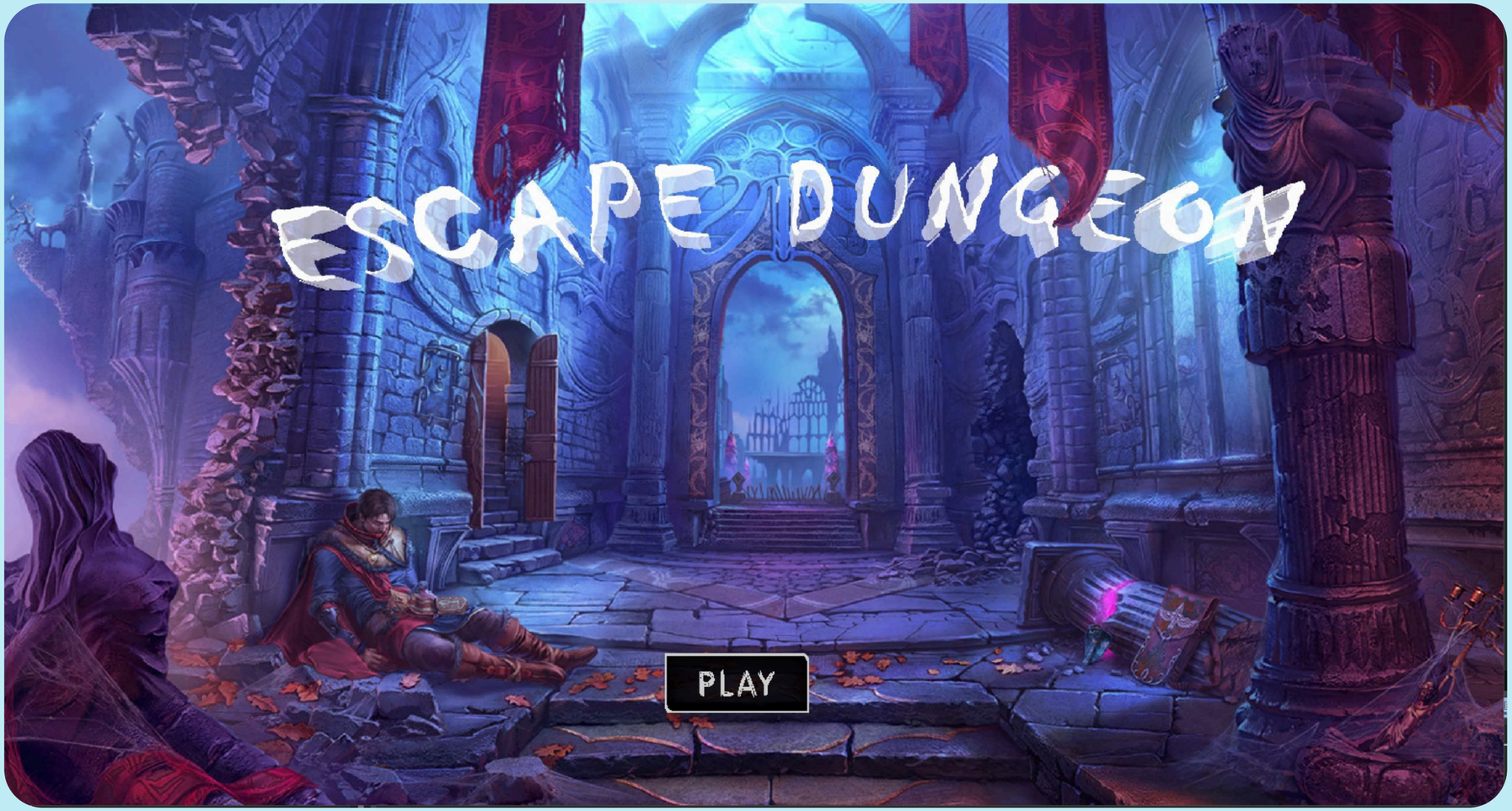
GITHUB

GIMP

03.

GRAPHIC DESGIN

STARTING MENU SCREEN



STARTING MENU SCREEN

```java
public class PlayingMenu extends JPanel {

    private static final long serialVersionUID = 1L;
    private JButton play;
    private Image backgroundImage,playButImage;

    public PlayingMenu() {
        try {
            UIManager.setLookAndFeel(new FlatDarkLaf());
        } catch (UnsupportedLookAndFeelException e) {
            e.printStackTrace();
        }
        setPreferredSize(new Dimension(1248, 700));
        setLayout(null);

        backgroundImage = new ImageIcon(getClass().getResource("/background1.jpg")).getImage();
        playButImage = new ImageIcon(getClass().getResource("/Play Button.png")).getImage();

        play = new JButton() {

            private static final long serialVersionUID = 1L;

            @Override
            protected void paintComponent(Graphics g) {
                g.drawImage(playButImage, 0, 0, getWidth(), getHeight(), this);
                super.paintComponent(g);
            }
        };
        play.setContentAreaFilled(false);
        play.setBorderPainted(false);
        play.setFocusable(false);
        play.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                playGame();
                setVisible(false);
```

```java
            }
        });
        play.setBounds(550, 540, 120, 50);
        add(play);

    }


    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(backgroundImage, 0, 0, 1248, 700, this);
    }

    private void playGame() {
        MainGame.pause = false;
        MapManager.firstCheck = false;
        Game_JPanel.selectionMenu = 0;
    }
```

PAUSE MENU SCREEN

PAUSE MENU SCREEN

```
public class PauseMenu extends JPanel {


    private static final long serialVersionUID = 1L;
    private JButton quit1;
    private JButton restart;
    private JButton continueButton;
    private JLabel gamePause;
    private JLabel music;
    private JLabel sound;
    private JSlider sliderMusic;
    private JSlider sliderSound;
    private JButton nextSong;
    private Image pauseImage,musicButImage,continueButImage,restartButImage,quit1ButImage;


    public PauseMenu() {

         try {
            UIManager.setLookAndFeel(new FlatDarkLaf());
        } catch (UnsupportedLookAndFeelException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        setPreferredSize(new Dimension(500, 600));
        setOpaque(false);
        setBackground(Color.DARK_GRAY);
        setLayout(null);
        pauseImage = new ImageIcon(getClass().getResource("/pause Menu.png")).getImage();
        musicButImage = new ImageIcon(getClass().getResource("/Music Square Button.png")).getImage();
        continueButImage = new ImageIcon(getClass().getResource("/Continue Button.png")).getImage();
        restartButImage = new ImageIcon(getClass().getResource("/New Game Button.png")).getImage();
        quit1ButImage = new ImageIcon(getClass().getResource("/Quit Button.png")).getImage();

        continueButton = new JButton(){
            private static final long serialVersionUID = 1L;

            @Override
            protected void paintComponent(Graphics g) {
```

```
                g.drawImage(continueButImage, 0, 0, getWidth(), getHeight(), this);
                super.paintComponent(g);
            }
        };
        continueButton.setContentAreaFilled(false);
        continueButton.setBorderPainted(false);
        continueButton.setBounds(80, 375, 120, 40);
        continueButton.setFocusable(false);
        continueButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if(!MainGame.pause)
                    MainGame.pause = true;
                else
                    MainGame.pause = false;
                setVisible(false);

            }
        });

        add(continueButton);

        restart = new JButton() {

            private static final long serialVersionUID = 1L;

            protected void paintComponent(Graphics g) {
                    g.drawImage(restartButImage, 0, 0, getWidth(), getHeight(), this);
                    super.paintComponent(g);
                }
        };
        restart.setContentAreaFilled(false);
        restart.setBorderPainted(false);
        restart.setBounds(300, 375, 120, 40);
        restart.setFocusable(false);
        restart.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                MainGame.restart = true;
                if(!MainGame.pause)
                    MainGame.pause = true;
                else
                    MainGame.pause = false;
```

PAUSE MENU SCREEN

```
        public void actionPerformed(ActionEvent e) {
            MainGame.restart = true;
            if(!MainGame.pause)
                MainGame.pause = true;
            else
                MainGame.pause = false;
            MainGame.resetSong = true;
            setVisible(false);
        }});

    add(restart);

    gamePause = new JLabel("Game Paused");
    gamePause.setFont(new Font("Tahoma", Font.BOLD, 30));
    gamePause.setForeground(java.awt.Color.BLACK);
    gamePause.setBounds(163, 58, 213, 43);
    add(gamePause);

    quit1 = new JButton() {

        private static final long serialVersionUID = 1L;

        protected void paintComponent(Graphics g) {
            g.drawImage(quit1ButImage, 0, 0, getWidth(), getHeight(), this);
            super.paintComponent(g);
        }
    };
    quit1.setContentAreaFilled(false);
    quit1.setBorderPainted(false);
    quit1.setBounds(190, 440, 120, 40);
    quit1.setFocusable(false);
    quit1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            promptPage();
        }
    });
    add(quit1);


    music = new JLabel("Music");
    music.setFont(new Font("Tahoma", Font.BOLD, 18));
```

```
    music.setForeground(java.awt.Color.BLACK);
    music.setBounds(70, 130, 62, 23);
    add(music);

    sound = new JLabel("Sound");
    sound.setFont(new Font("Tahoma", Font.BOLD, 18));
    sound.setForeground(java.awt.Color.BLACK);
    sound.setBounds(70, 248, 62, 23);
    add(sound);

    sliderMusic = new JSlider(-60,6,-23);
    sliderMusic.setBounds(145, 130, 200, 26);
    sliderMusic.setFocusable(false);
    sliderMusic.setForeground(java.awt.Color.BLACK);
    sliderMusic.setValue(-23);
    sliderMusic.addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent e) {
            changeVolumeMusic();
        }});
    add(sliderMusic);

    sliderSound = new JSlider(-60,6,-23);
    sliderSound.setBounds(145, 248, 200, 26);
    sliderSound.setFocusable(false);
    sliderSound.setForeground(java.awt.Color.BLACK);
    sliderSound.setValue(-40);
    sliderSound.addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent e) {
            changeVolumeMusic();
        }});
    add(sliderSound);

    nextSong = new JButton() {

        private static final long serialVersionUID = 1L;

        @Override
        protected void paintComponent(Graphics g) {
            g.drawImage(musicButImage, 0, 0, getWidth(), getHeight(), this);
            super.paintComponent(g);
        }
```

```
        nextSong.setContentAreaFilled(false);
        nextSong.setBorderPainted(false);
        nextSong.setBounds(374, 119, 50, 50);
        nextSong.setFocusable(false);
        nextSong.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                nextSong();
            }});
        add(nextSong);
        System.out.println(pauseImage.getClass());

    }

    private void promptPage() {
        int result = JOptionPane.showConfirmDialog(
            null,
            "A chicken loser?\nWant to give up?",
            "Confirmation",
            JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE
        );
        if (result == JOptionPane.YES_OPTION) {
            System.exit(0);
        }
    }

     protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.drawImage(pauseImage, 0, 0, getWidth(), getHeight(), this);

        }

    private void changeVolumeMusic() {
        MainGame.ChangeVolume = sliderMusic.getValue();
        Player.ChangeVolume = sliderSound.getValue();
    }

    private void nextSong() {
        MainGame.nextSong = true;
    }
}
```

DIE MENU SCREEN

DIE MENU SCREEN

```
public class DieMenu extends JPanel {

    private static final long serialVersionUID = 1L;
    private JButton restart;
    private JLabel diePause;
    private JLabel label;
    private JButton quit;
    private Image dieBgImage,restartButImage,quit1ButImage;


    public DieMenu() {

         try {
            UIManager.setLookAndFeel(new FlatDarkLaf());
        } catch (UnsupportedLookAndFeelException e) {
            e.printStackTrace();
        }

        dieBgImage = new ImageIcon(getClass().getResource("/pause Menu.png")).getImage();
        restartButImage = new ImageIcon(getClass().getResource("/New Game Button.png")).getImage();
        quit1ButImage = new ImageIcon(getClass().getResource("/Quit Button.png")).getImage();

        setPreferredSize(new Dimension(500, 600));
        setOpaque(false);
        setBackground(Color.DARK_GRAY);
        setLayout(null);

        restart = new JButton(){
            @Override
             protected void paintComponent(Graphics g) {
                    g.drawImage(restartButImage, 0, 0, getWidth(), getHeight(), this);
                    super.paintComponent(g);
                }
```

```
        restart.setContentAreaFilled(false);
        restart.setBorderPainted(false);
        restart.setBounds(179, 254, 132, 56);
        restart.setFocusable(false);
        restart.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                MainGame.restart = true;
                MainGame.pause = false;
                MainGame.die = false;
                MainGame.resetSong = true;
                setVisible(false);
            }});
        add(restart);

        diePause = new JLabel(" You Died");
        diePause.setFont(new Font("Tahoma", Font.BOLD, 45));
        diePause.setForeground(java.awt.Color.BLACK);
        diePause.setBounds(135, 13, 250, 143);
        add(diePause);

        label = new JLabel("Such a chicken !");
        label.setFont(new Font("Tahoma", Font.BOLD, 45));
        label.setForeground(java.awt.Color.BLACK);
        label.setBounds(80, 129, 388, 85);
        add(label);

        quit = new JButton() {
            protected void paintComponent(Graphics g) {
                g.drawImage(quit1ButImage, 0, 0, getWidth(), getHeight(), this);
                super.paintComponent(g);
            }
        };
```

DIE MENU SCREEN

```
        quit.setContentAreaFilled(false);
        quit.setBorderPainted(false);
        quit.setFocusable(false);
        quit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                promptPage();
            }});
        quit.setBounds(179, 376, 132, 56);
        add(quit);
    }

    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(dieBgImage, 0, 0, getWidth(), getHeight(), this);

    }

    private void promptPage() {
        int result = JOptionPane.showConfirmDialog(null, "A chicken loser ?\nWant to give up ?", "Confirmation",JOptionPane.YI
          if (result == JOptionPane.YES_OPTION) {
              System.exit(0);
          }
    }
```

PLAYING SCREEN

PLAYER CHARACTER

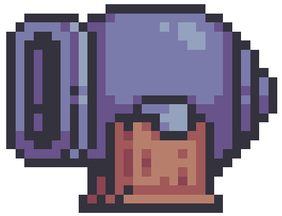
NPC

ENEMY

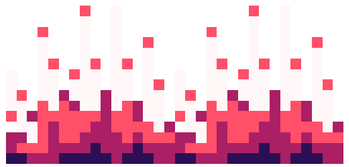
TRAPS

TINY-FACE BALL

CANNON



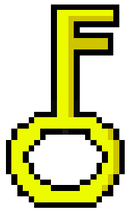
SPIKE TRAP



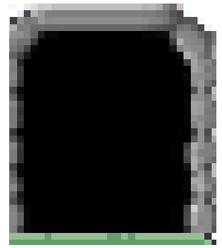
OBJECTS

HEALTH POTION

KEY



DOOR



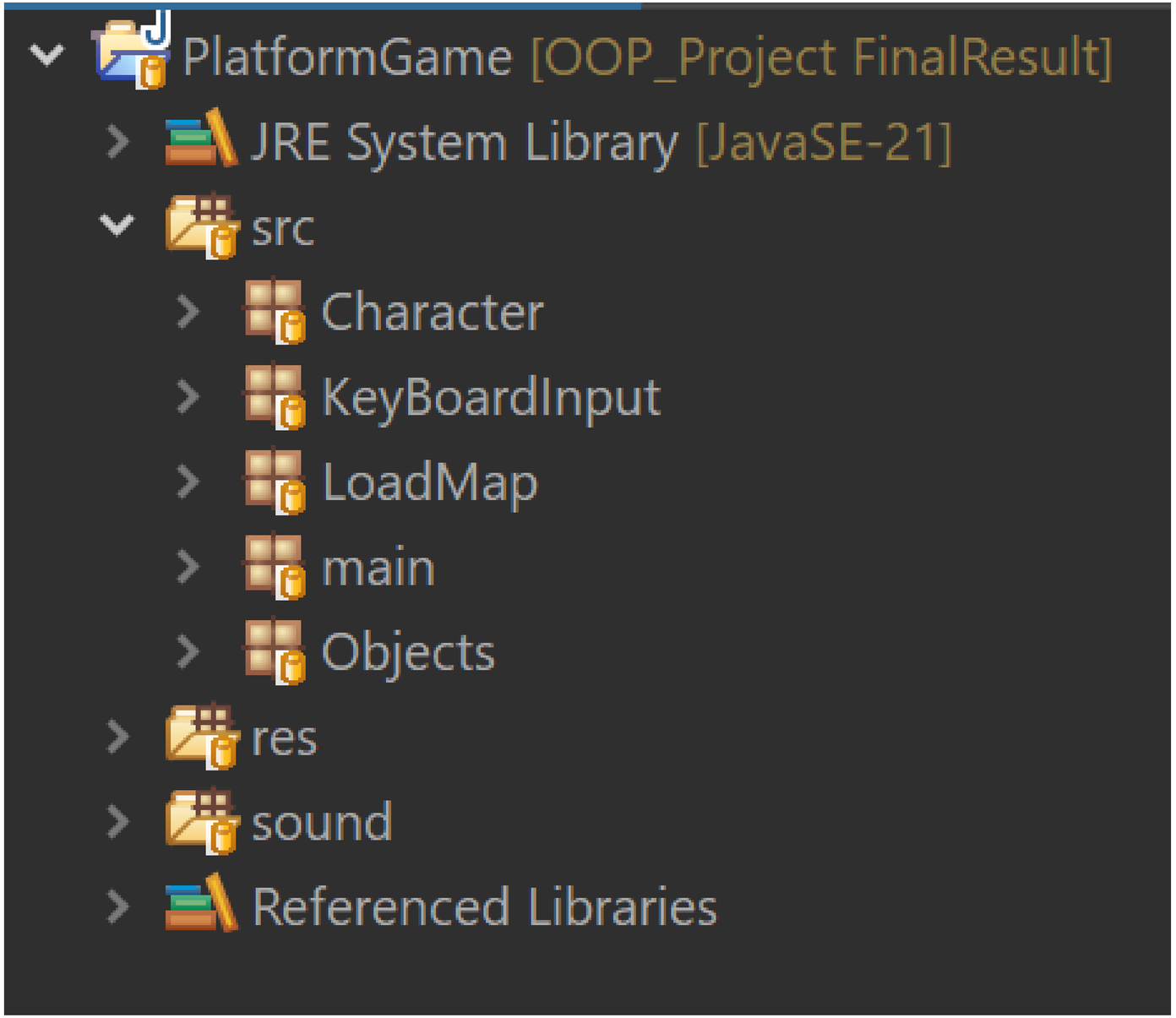
04.

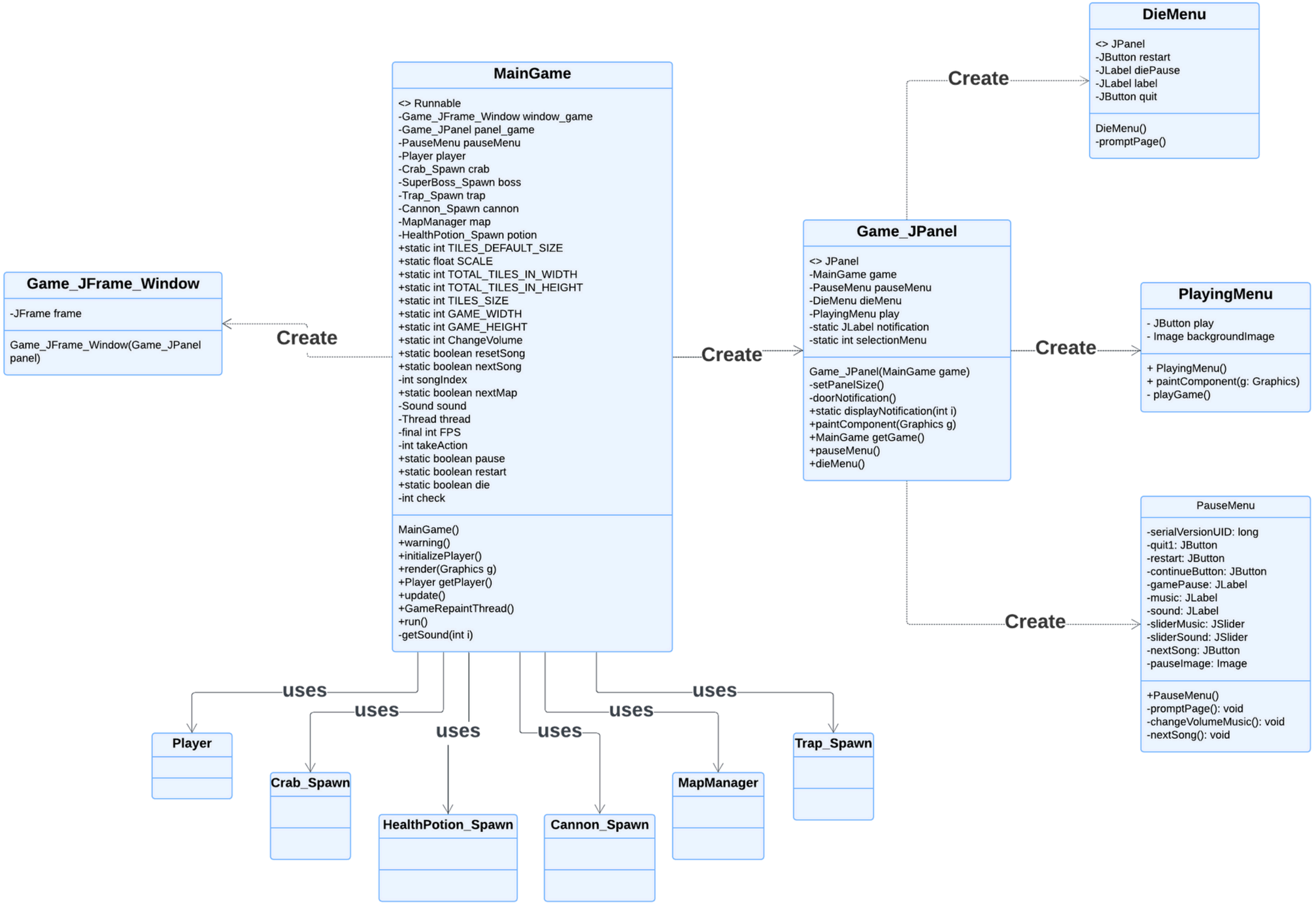
SYSTEM DESGIN

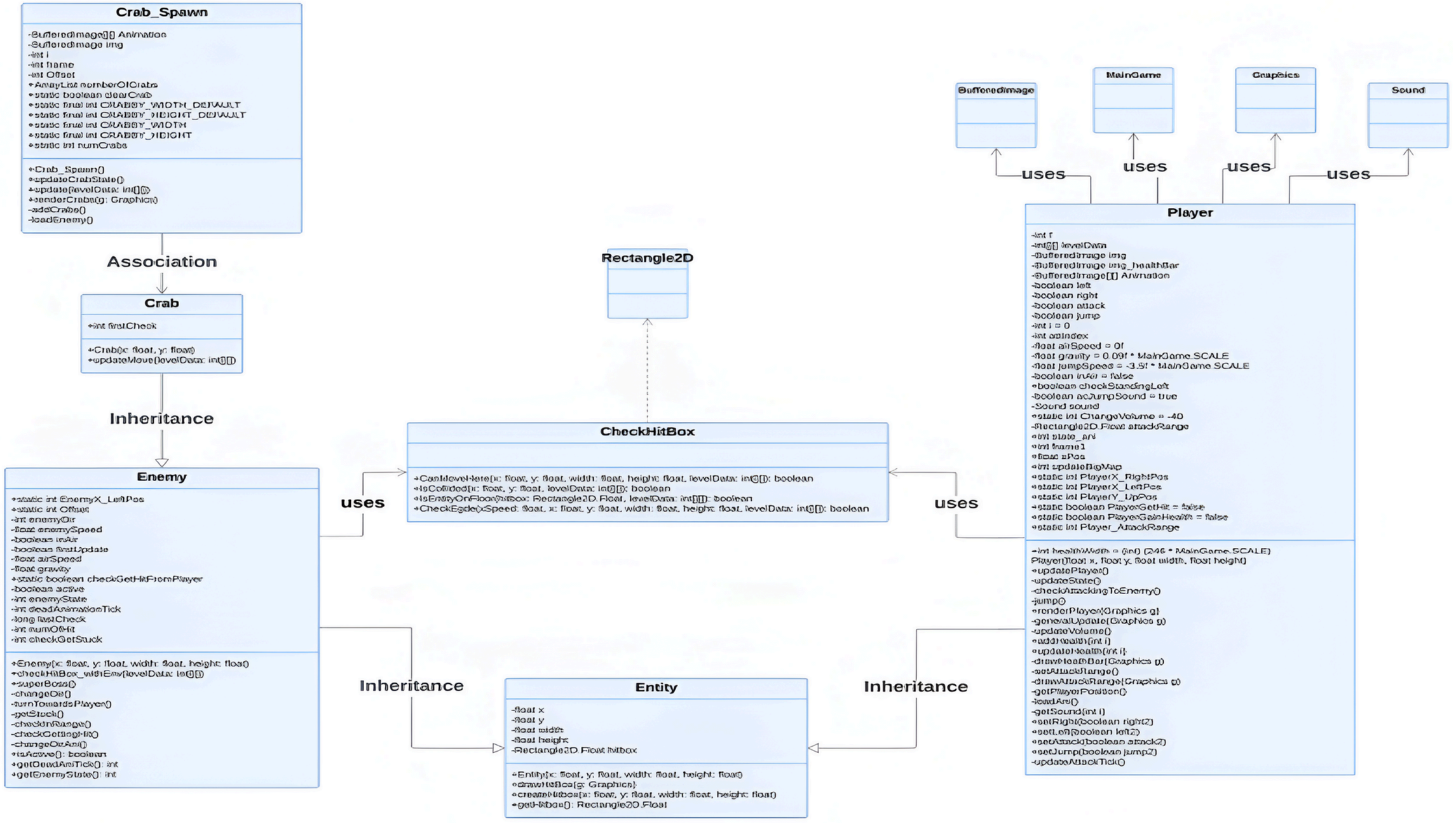
PACKAGE STRUCTURE

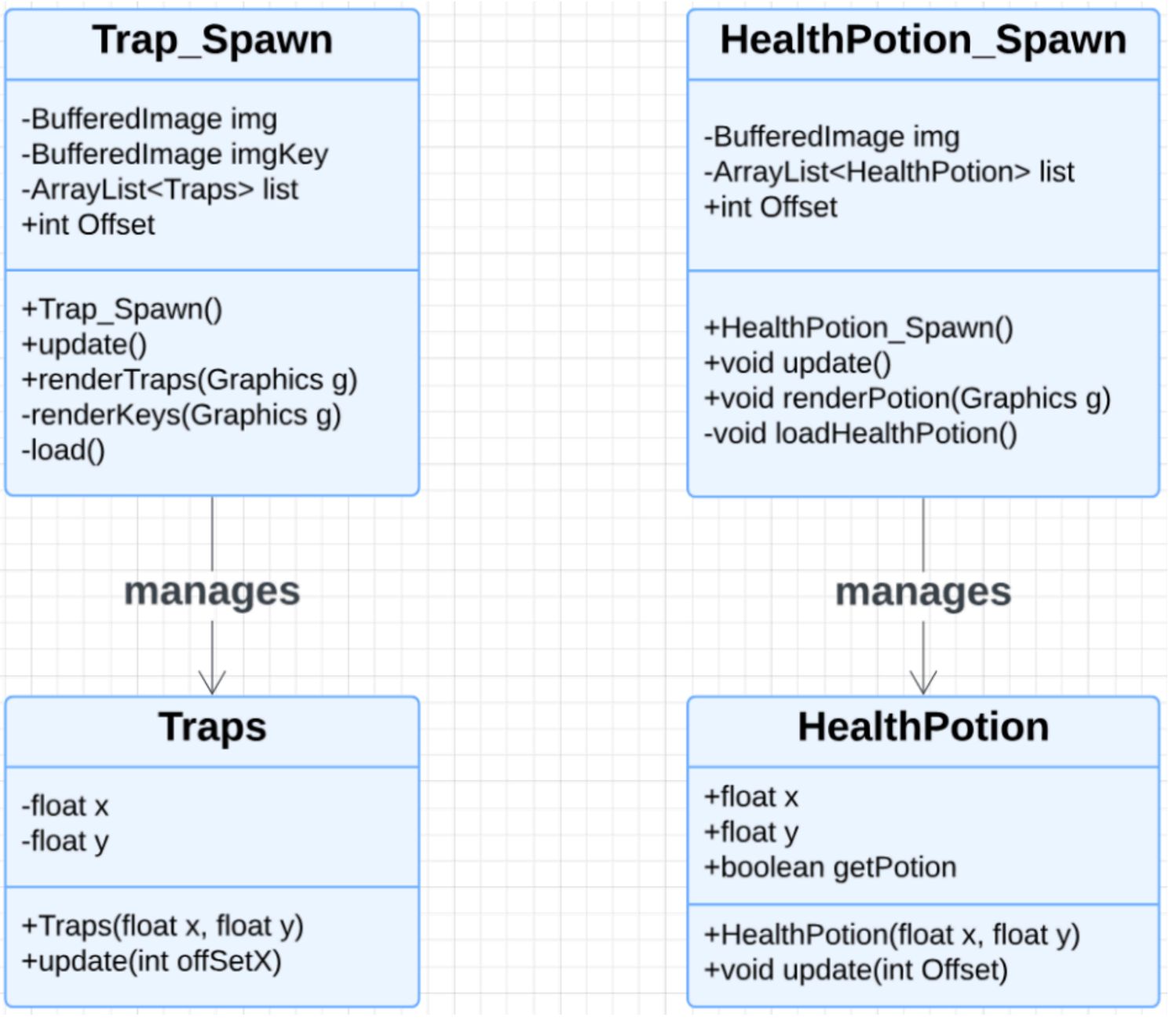
Organizing multiple classes into distinct groups for better management, including:

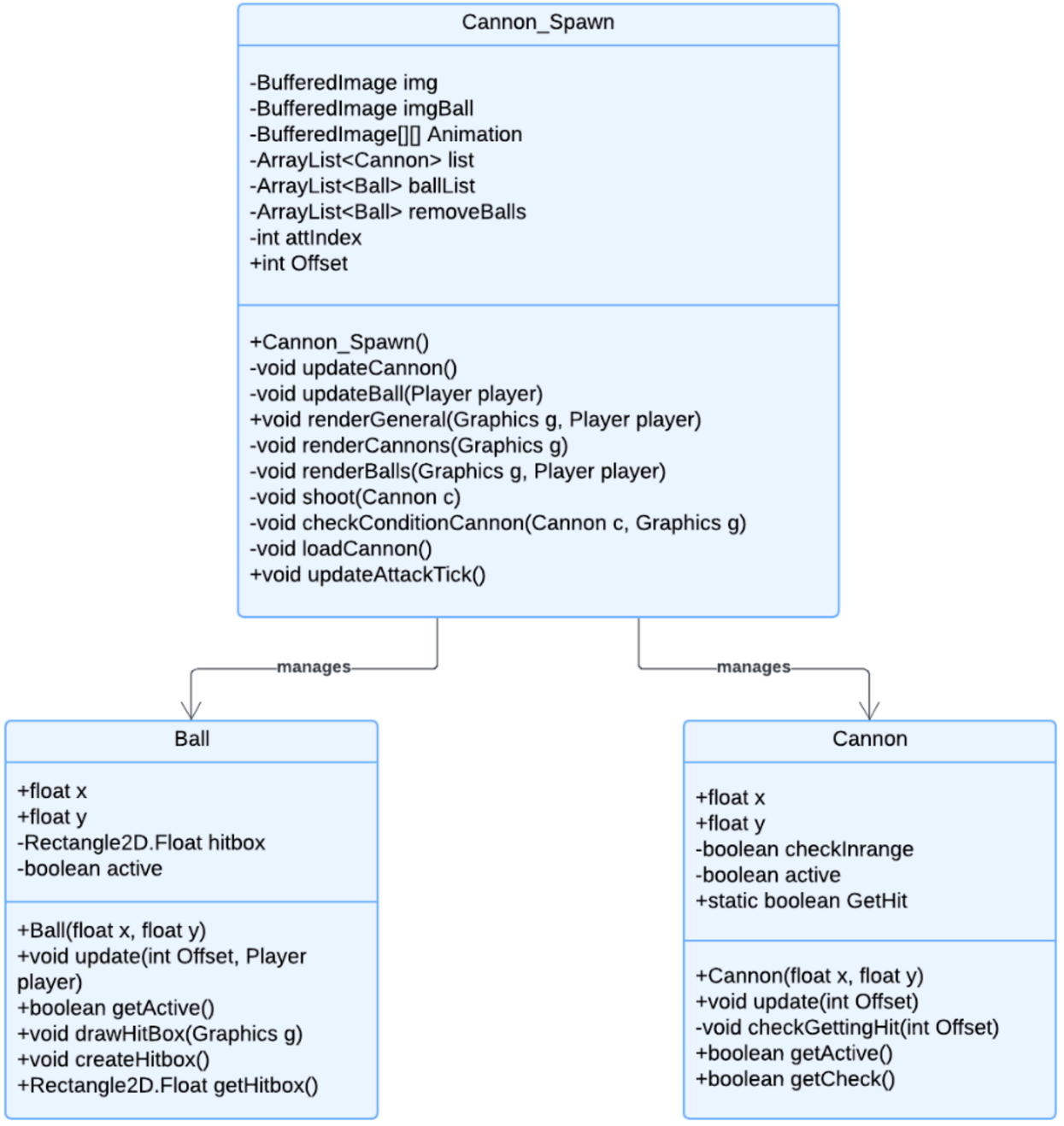
- **5 PACKAGES**
- **29 CLASSES**

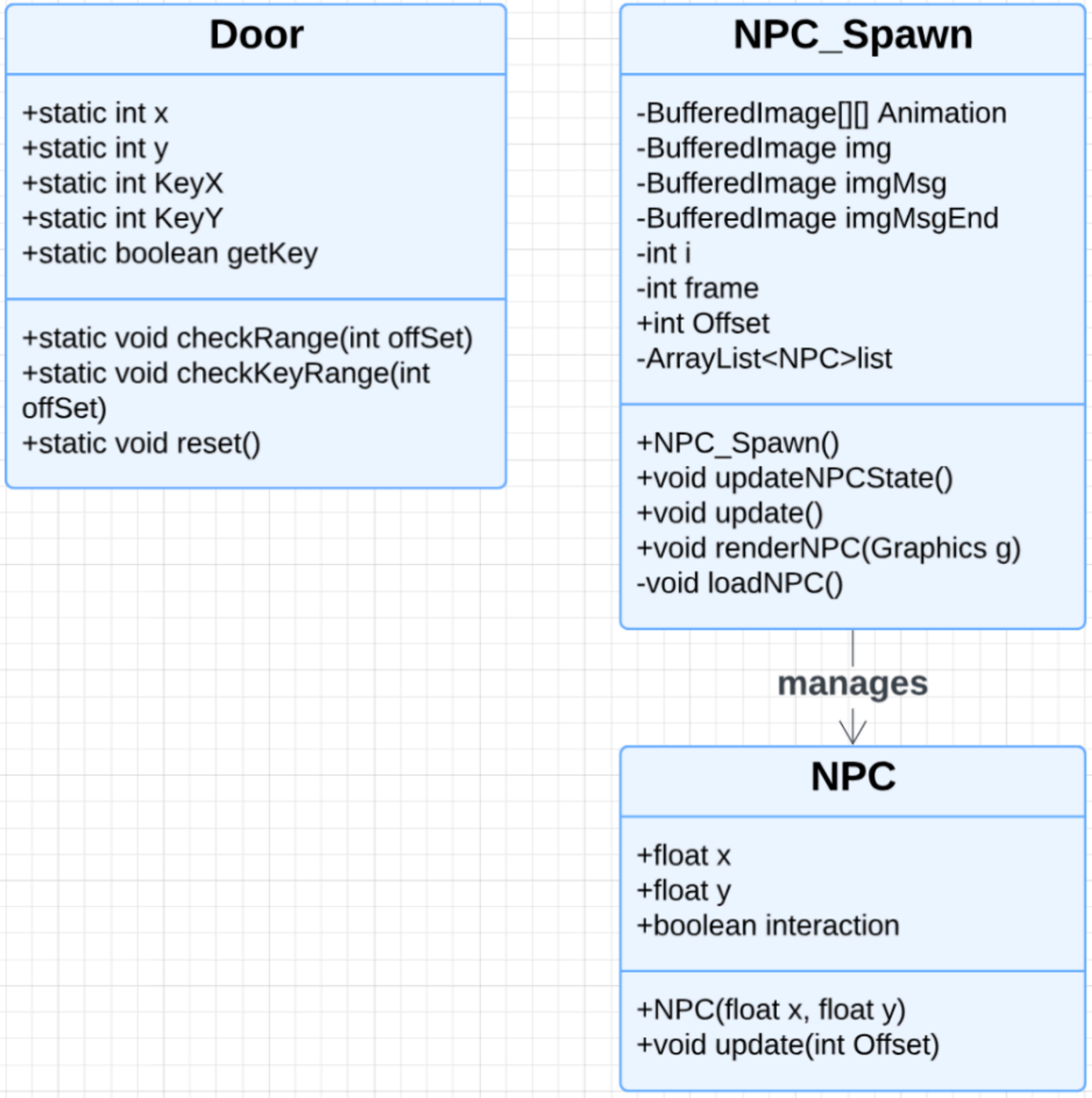



DieMenu
<> JPanel
-JButton restart
-JLabel diePause
-JLabel label
-JButton quit
DieMenu()
-promptPage()
Create
MainGame
<> Runnable
-Game_JFrame_Window window_game
-Game_JPanel panel_game
-PauseMenu pauseMenu
-Player player
-Crab_Spawn crab
-SuperBoss_Spawn boss
-Trap_Spawn trap
-Cannon_Spawn cannon
-MapManager map
-HealthPotion_Spawn potion
+static int TILES_DEFAULT_SIZE
+static float SCALE
+static int TOTAL_TILES_IN_WIDTH
+static int TOTAL_TILES_IN_HEIGHT
+static int TILES_SIZE
+static int GAME_WIDTH
+static int GAME_HEIGHT
+static int ChangeVolume
+static boolean resetSong
+static boolean nextSong
-int songIndex
+static boolean nextMap
-Sound sound
-Thread thread
-final int FPS
-int takeAction
+static boolean pause
+static boolean restart
+static boolean die
-int check
MainGame()
+warning()
+initializePlayer()
+render(Graphics g)
+Player getPlayer()
+update()
+GameRepaintThread()
+run()
-getSound(int i)
Game_JFrame_Window
-JFrame frame
Game_JFrame_Window(Game_JPanel panel)
Create
Create
Game_JPanel
<> JPanel
-MainGame game
-PauseMenu pauseMenu
-DieMenu dieMenu
-PlayingMenu play
-static JLabel notification
-static int selectionMenu
Game_JPanel(MainGame game)
-setPanelSize()
-doorNotification()
+static displayNotification(int i)
+paintComponent(Graphics g)
+MainGame getGame()
+pauseMenu()
+dieMenu()
Create
PlayingMenu
- JButton play
- Image backgroundImage
+ PlayingMenu()
+ paintComponent(g: Graphics)
- playGame()
Create
PauseMenu
-serialVersionUID: long
-quit1: JButton
-restart: JButton
-continueButton: JButton
-gamePause: JLabel
-music: JLabel
-sound: JLabel
-sliderMusic: JSlider
-sliderSound: JSlider
-nextSong: JButton
-pauseImage: Image
+PauseMenu()
-promptPage(): void
-changeVolumeMusic(): void
-nextSong(): void
uses
uses
uses
uses
uses
uses
Player
Crab_Spawn
HealthPotion_Spawn
Cannon_Spawn
MapManager
Trap_Spawn


Crab_Spawn
-BufferedImage[][] Animation
-BufferedImage img
-int i
-int name
-int Offset
+ArrayList numberOfCrabs
+static boolean deadCrab
+static final int CRABBY_WIDTH_DEFAULT
+static final int CRABBY_HEIGHT_DEFAULT
+static final int CRABBY_WIDTH
+static final int CRABBY_HEIGHT
+static int numCrabs
+Crab_Spawn()
+updateCrabState()
+updateRevelData: int[][]
+renderCrabs(g: Graphics)
-addCrabs()
-loadEnemy()
Association
Crab
+int firstCheck
+Crab(x: float, y: float)
+updateMove(levelData: int[][]
Inheritance
Enemy
+static int EnemyX_LeftPos
+static int Offset
-int enemyDir
-float enemySpeed
-boolean isAir
-boolean firstUpdate
-float airSpeed
-float gravity
+static boolean checkGetHitFromPlayer
-boolean active
-int enemyState
-int deadAnimationTick
-long firstCheck
-int numOfHit
-int checkGetStuck
+Enemy(x: float, y: float, width: float, height: float)
+checkHitBox_withEnv(levelData: int[][]
+superBoss()
-changeDir()
-turnTowardsPlayer()
-getStuck()
-checkInRange()
-checkGetHigHit()
-changeDead()
+isActive(): boolean
+getDeadAniTick(): int
+getEnemyState(): int
Rectangle2D
CheckHitBox
+CanMoveHere(x: float, y: float, width: float, height: float, levelData: int[][]: boolean
+isCollided(x: float, y: float, levelData: int[][]: boolean
+isEntityOnFloor(hitbox: Rectangle2D.Float, levelData: int[][]: boolean
+CheckEdge(xSpeed: float, x: float, y: float, width: float, height: float, levelData: int[][]: boolean
uses
uses
Inheritance
Entity
-float x
-float y
-float width
-float height
-Rectangle2D.Float hitbox
+Entity(x: float, y: float, width: float, height: float)
+drawHitbox(g: Graphics)
+createHitbox(x: float, y: float, width: float, height: float)
+getHitbox(): Rectangle2D.Float
Inheritance
BufferedImage
MainGame
Graphics
Sound
uses
uses
uses
uses
Player
-int i
-int[][] levelData
-BufferedImage img
-BufferedImage img_healthBar
-BufferedImage[][] Animation
-boolean left
-boolean right
-boolean attack
-boolean jump
-int i = 0
-int aniIndex
-float airSpeed = 0f
-float gravity = 0.09f * MainGame.SCALE
-float jumpSpeed = -3.5f * MainGame.SCALE
-boolean inAir = false
+boolean checkStandingLeft
-boolean playJumpSound = true
-Sound sound
+static int ChangeVolume = -40
-Rectangle2D.Float attackRange
+int state_ani
+int frame1
+float xPos
+int updateTickMap
+static int PlayerX_RightPos
+static int PlayerX_LeftPos
+static int PlayerY_UpPos
+static boolean PlayerGetHit = false
+static boolean PlayerGainHealth = false
+static int Player_AttackRange
+int healthWidth = (int) (246 * MainGame.SCALE)
Player(float x, float y, float width, float height)
+updatePlayer()
-updateState()
-checkAttackingToEnemy()
-jump()
+renderPlayer(Graphics g)
-generalUpdate(Graphics g)
-updateVolume()
+addHealth(int i)
+updateHealth(int i)
-drawHealthBar(Graphics g)
-setAttackRange()
-drawAttackRange(Graphics g)
-getPlayerPosition()
-loadAni()
-getSound(int i)
+setRight(boolean right)
+setLeft(boolean left)
+setAttack(boolean attack)
+setJump(boolean jump)
-updateAttackTick()


Trap_Spawn
-BufferedImage img
-BufferedImage imgKey
-ArrayList<Traps> list
+int Offset
+Trap_Spawn()
+update()
+renderTraps(Graphics g)
-renderKeys(Graphics g)
-load()
manages
Traps
-float x
-float y
+Traps(float x, float y)
+update(int offSetX)
HealthPotion_Spawn
-BufferedImage img
-ArrayList<HealthPotion> list
+int Offset
+HealthPotion_Spawn()
+void update()
+void renderPotion(Graphics g)
-void loadHealthPotion()
manages
HealthPotion
+float x
+float y
+boolean getPotion
+HealthPotion(float x, float y)
+void update(int Offset)


Cannon_Spawn
-BufferedImage img
-BufferedImage imgBall
-BufferedImage[][] Animation
-ArrayList<Cannon> list
-ArrayList<Ball> ballList
-ArrayList<Ball> removeBalls
-int attIndex
+int Offset
+Cannon_Spawn()
-void updateCannon()
-void updateBall(Player player)
+void renderGeneral(Graphics g, Player player)
-void renderCannons(Graphics g)
-void renderBalls(Graphics g, Player player)
-void shoot(Cannon c)
-void checkConditionCannon(Cannon c, Graphics g)
-void loadCannon()
+void updateAttackTick()
manages
manages
Ball
+float x
+float y
-Rectangle2D.Float hitbox
-boolean active
+Ball(float x, float y)
+void update(int Offset, Player player)
+boolean getActive()
+void drawHitBox(Graphics g)
+void createHitbox()
+Rectangle2D.Float getHitbox()
Cannon
+float x
+float y
-boolean checkInrange
-boolean active
+static boolean GetHit
+Cannon(float x, float y)
+void update(int Offset)
-void checkGettingHit(int Offset)
+boolean getActive()
+boolean getCheck()


Door
+static int x
+static int y
+static int KeyX
+static int KeyY
+static boolean getKey
+static void checkRange(int offSet)
+static void checkKeyRange(int offSet)
+static void reset()
NPC_Spawn
-BufferedImage[][] Animation
-BufferedImage img
-BufferedImage imgMsg
-BufferedImage imgMsgEnd
-int i
-int frame
+int Offset
-ArrayList<NPC>list
+NPC_Spawn()
+void updateNPCState()
+void update()
+void renderNPC(Graphics g)
-void loadNPC()
manages
NPC
+float x
+float y
+boolean interaction
+NPC(float x, float y)
+void update(int Offset)

05.

CODE FLOW

MAINGAME FOR STARTING THE GAME

```
65    public MainGame() {
66
67        sound = new Sound();
68        getSound(songIndex);
69
70        initializePlayer();
71
72        panel_game = new Game_JPanel(this);
73        window_game = new Game_JFrame_Window(panel_game);
74        GameRepaintThread();
75
76    }
77
78    public void warning() {
79
80    }
81
82    public void initializePlayer() {
83        Load.mapControler();
84        map = new MapManager();
85        if(check == 1) {
86        potion = new HealthPotion_Spawn();
87        npc = new NPC_Spawn();
88        trap = new Trap_Spawn();
89        cannon = new Cannon_Spawn();
90        player = new Player(30,300, (int) (96*SCALE), (int) (84*SCALE) );
91        crab = new Crab_Spawn();
92        }
93    }
94
95    public void render(Graphics g) {
96        if(check == 1)
97           map.player_xPos = player.xPos;
98        map.get_Map_Level(g);
99
100       if(!pause && check == 1) {
101          player.updateBigMap = map.xLvlOffset;
102          player.renderPlayer(g);
```

MAINGAME FOR STARTING THE GAME

```
103
104             crab.Offset =  map.xLvlOffset;
105             crab.renderCrabs(g);
106
107             trap.Offset =  map.xLvlOffset;
108             trap.renderTraps(g);
109
110             potion.Offset = map.xLvlOffset;
111             potion.renderPotion(g);
112
113             npc.Offset = map.xLvlOffset;
114             npc.renderNPC(g);
115
116
117             cannon.Offset =  map.xLvlOffset;
118             cannon.renderGeneral(g,player);
119         }
120
121         Door.checkRange(map.xLvlOffset);
122         Door.checkKeyRange(map.xLvlOffset);
123
124     }
```

RUN METHOD CALLED BY THREAD IN MAIN GAME

```java
@Override
public void run() {

    double nanoFPS = 1000000000.0  / FPS;
    long lastTime = System.nanoTime();
    long now = System.nanoTime();
    long loadframe = System.currentTimeMillis();
    long nowframe = System.currentTimeMillis();

    while(true) {
        checkSound();
        now = System.nanoTime();

        if(now - lastTime >= nanoFPS) {
            panel_game.repaint();
            lastTime = now;
        }

        if(winning && Door.getKey) {
            if(winningPrompt()) {
                Door.reset();
                initializePlayer();
            }else
                System.exit(0);

        }
        winning = false;

        nowframe = System.currentTimeMillis();
        if(nowframe - loadframe >= 150) {
            update();
            loadframe = System.currentTimeMillis();

        }
    }

}
```

GAME_JPANEL

```java
    public Game_JPanel(MainGame game) {
        this.game = game;

        pauseMenu = new PauseMenu();
        pauseMenu.setVisible(false);

        dieMenu = new DieMenu();
        dieMenu.setVisible(false);

        play = new PlayingMenu();
        play.setVisible(true);

        add(pauseMenu);
        add(dieMenu);
        add(play);
        doorNotification();
        setPanelSize();

     }

    private void setPanelSize() {
        Dimension size = new Dimension(MainGame.GAME_WIDTH, MainGame.GAME_HEIGHT);
        setMinimumSize(size);
        setPreferredSize(size);
        setMaximumSize(size);

    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        game.render(g);

    }


    public MainGame getGame() {
     return game;
    }
```

METHOD TO LOAD MAPS

```
79      public static int[][] GetMapLevelData() {
80
81          BufferedImage img = LoadImage(mapControler.get(index));
82          int[][] levelData = new int[img.getHeight()][img.getWidth()];
83
84
85          for (int j = 0; j < img.getHeight(); j++) {
86              for (int i = 0; i < img.getWidth(); i++) {
87                  Color color = new Color(img.getRGB(i, j));
88                  int value  = color.getRed();
89                  initializeObjects(i,j,color.getBlue(),color.getGreen());
90
91                  if (value >= 48)
92                      value = 0;
93
94                  if(value == 23) {
95                      Door.x = i*MainGame.TILES_SIZE;
96                      Door.y = j*MainGame.TILES_SIZE;
97                  }
98
99                  levelData[j][i] = value;
100             }
101
102         }
103
104         return levelData;
105
106     }
```

```java
    private static void initializeObjects(int i, int j, int value1,int value2) {
        switch (value1) {
        case 0:
            getTraps.add(new Traps(i * MainGame.TILES_SIZE, j * MainGame.TILES_SIZE));
            break;

        case 5:
            getCannon.add(new Cannon(i * MainGame.TILES_SIZE, j * MainGame.TILES_SIZE));
            break;

        case 6:
            Door.KeyX = i * MainGame.TILES_SIZE;
            Door.KeyY = j * MainGame.TILES_SIZE;
            break;

        case 7:
            getHeathPotion.add(new HealthPotion(i * MainGame.TILES_SIZE, j * MainGame.TILES_SIZE));
            break;

        case 8:
            getNPC.add(new NPC(i * MainGame.TILES_SIZE, j * MainGame.TILES_SIZE));
            break;
        }

        if (value2 == 0)
            getCrab.add(new Crab(i*MainGame.TILES_SIZE, j*MainGame.TILES_SIZE));

    }

}
```

LOAD TILE SET

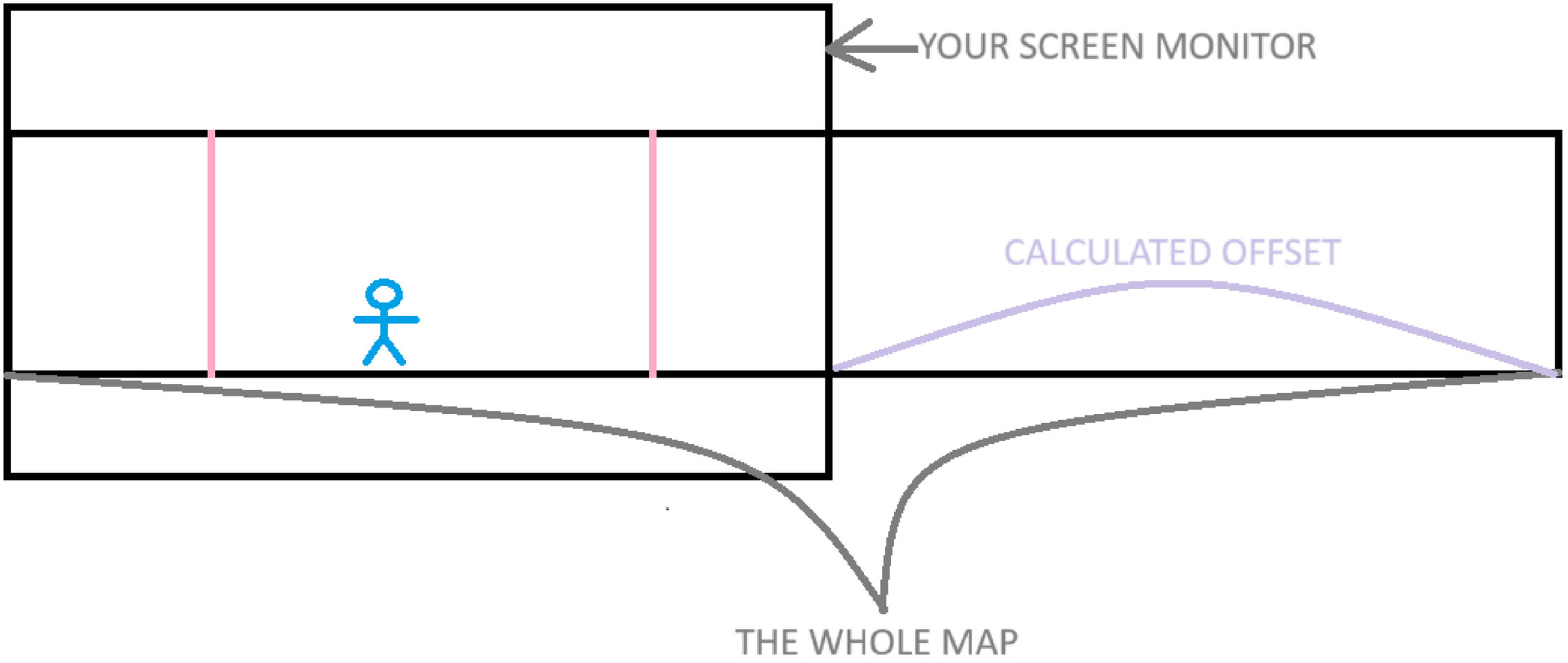
```
public void get_Tile_Block() {
    image = Load.LoadImage(Load.TILE_2D);
    index = new BufferedImage[48];

    int value = 0;

      for(int i = 0; i < 4; i++) {
          for(int j = 0; j < 12; j++ ) {
              index[value] = image.getSubimage(j*32, i*32, 32, 32);
              value++;

          }
      }value = 0;
}
```

MERGE TILE SET

```
public void get_Map_Level(Graphics g) {
    checkMoveLeftAndRight();


    if(firstCheck) {
      g.setColor(Color.DARK_GRAY);
      g.fillRect(0 , 0 , 1248, 672);
    }else
      g.drawImage(scaledImage, 0, 0, 1248, 672, null);

    for (int j = 0; j < MainGame.TOTAL_TILES_IN_HEIGHT; j++)
        for (int i = 0; i < levelData_Player[0].length; i++) {
            int num = level.get_levelData_index(i, j);
            g.drawImage(index[num], MainGame.TILES_SIZE * i - xLvlOffset, MainGame.TILES_SIZE * j , MainGame.TILES_SIZE, MainGame.TILES_SIZE,
        }


}
```

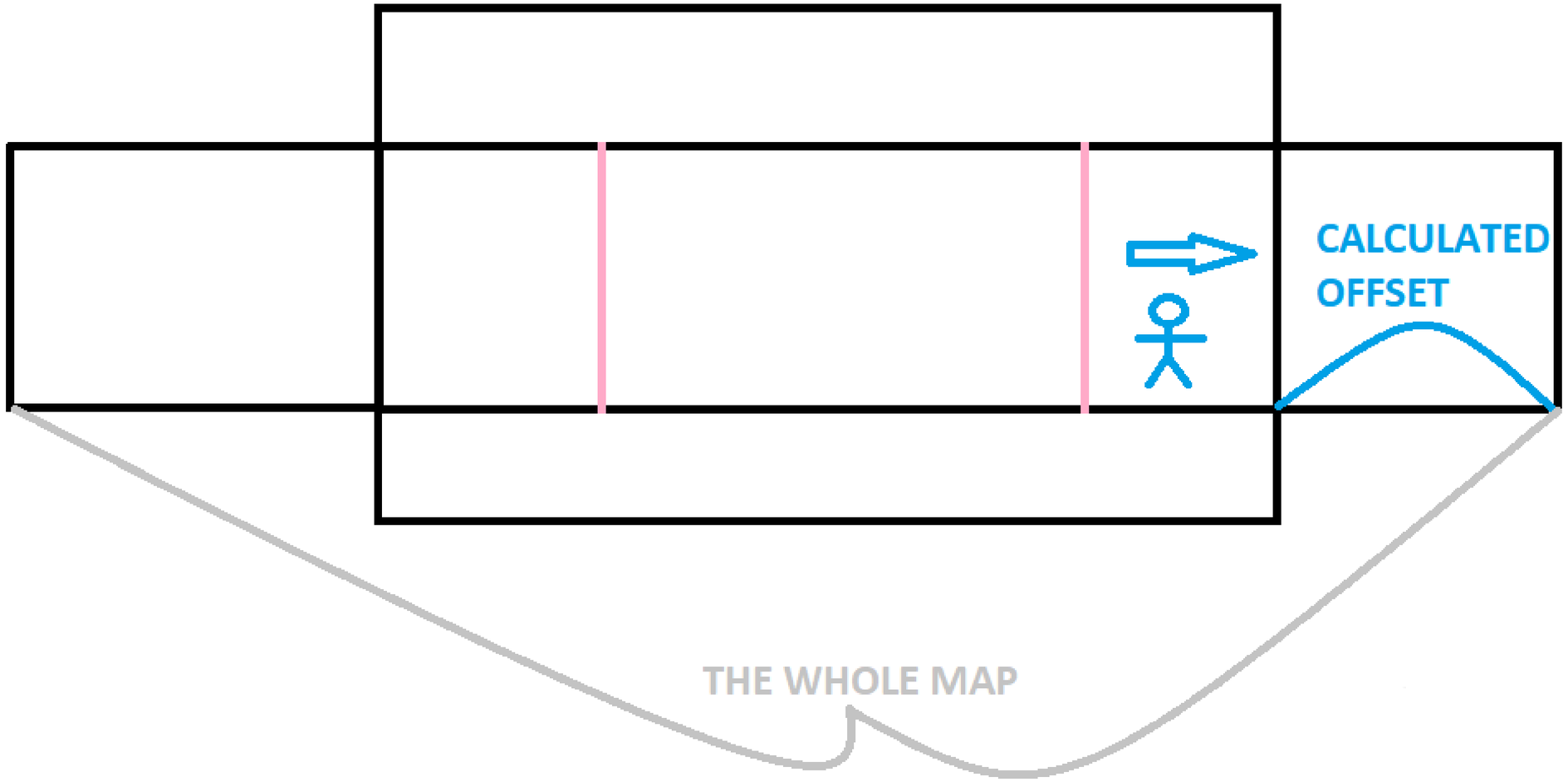
CALCULATE OFFSET

```
public void checkMoveLeftAndRight() {
    int playerX = (int) player_xPos;
    diff = playerX - xLvlOffset;

    if (diff > rightBorder)
        xLvlOffset += diff - rightBorder;
    else if (diff < leftBorder)
        xLvlOffset += diff - leftBorder;

    if (xLvlOffset > maxLvlOffsetX)
        xLvlOffset = maxLvlOffsetX;
    else if (xLvlOffset < 0)
        xLvlOffset = 0;


}
```



SCREEN DISPLAYED = WHOLE MAP - CALCULATED OFFSET

CALCULATED OFFSET BASED ON PLAYER POSITION



SCREEN DISPLAYED = WHOLE MAP - CALCULATED OFFSET

CALCULATED OFFSET BASED ON PLAYER POSITION

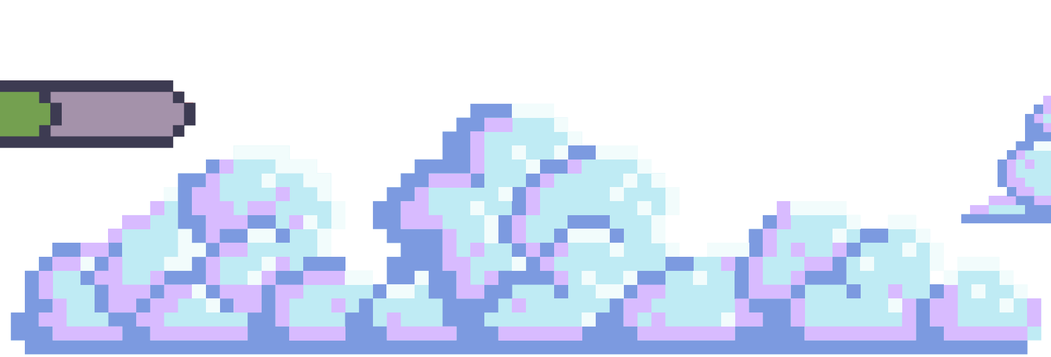
FIND KEY
KILL ENEMIES
TO WIN

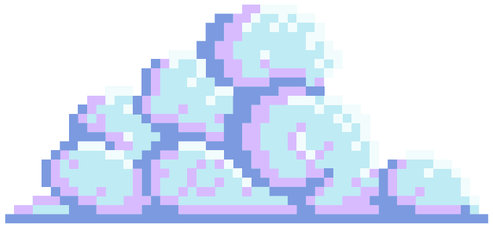
```
public class Door {

    public static int x;
    public static int y;
    public static int KeyX;
    public static int KeyY;


    public static boolean getKey = false;

    public static void checkRange(int offSet) {

        double range = x - Player.PlayerX_RightPos - offSet;
        if(range <= 27 && !getKey )
            Game_JPanel.displayNotification(1);
        else
            Game_JPanel.displayNotification(0);

        if(range <= 27)
            if(getKey && !Crab_Spawn.clearCrab)
                System.out.println("Kill all enemies");
            else
                MainGame.winning = true;

    }

    public static void checkKeyRange(int offSet) {
        double rangeX = KeyX - Player.PlayerX_RightPos - offSet;
        double rangeY = KeyY - Player.PlayerY_UpPos;

        if(Math.abs(rangeX) < 25 && rangeY > 45)
            getKey = true;

    }
```

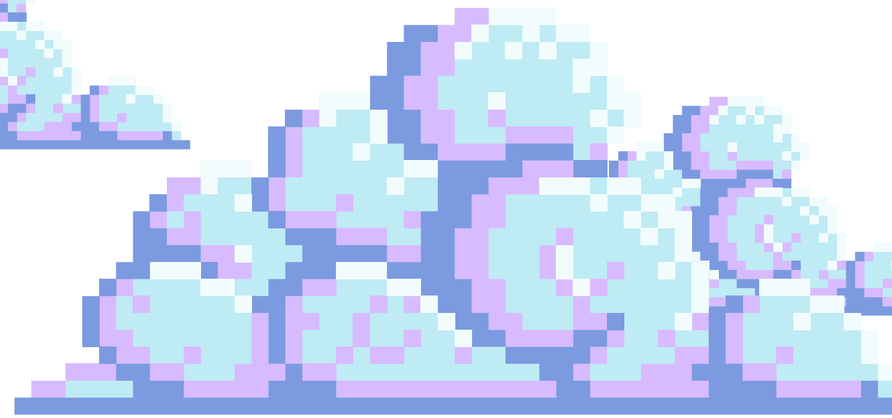
06.

CONCLUSION

"I never dreamt of success. I worked for it."
-Estée Lauder-







- ***Secure our Knowledge***

- ***Teamwork to Success***

- ***Grasp the Fundamentals of Game Development***

△□✕○

06.

FUTURE WORK

"It does not matter how slowly you go so long as you do not stop."
—Confucius-

FUTURE WORK

Improve

Story

Dynamic

Creative

Gameplay

Level Design

REFERENCE

[1] Platform Game (2022)[Java]. Retrieved May 10, 2024, from https://github.com/KaarinGaming/PlatformerTutorial

[2] Java Swing Tutorial. Javatpoint. (n.d.). Retrieved December 11, 2023, from https://www.javatpoint.com/java-swing

[3] UML Diagrams Tutorial. Javatpoint. (n.d.). Retrieved January 4, 2023, from https://www.javatpoint.com/uml-diagrams

REFERENCE

[4] Drew, O. (2022). Othneildrew/Best-README-Template., from https://github.com/othneildrew/Best-README-Template

[5] SOLID Principle in Programming: Understand With Real Life Examples. GeeksforGeeks. Retrieved December 28, 2023, from https://www.geeksforgeeks.org/solid-principle

[*]quintino-pixels.itch.io/tilemap-mini-dungeon, from https://quintino-pixels.itch.io/starry-night

THANK YOU

Do you have any question?