โครงงานคอมพิวเตอร์

เกมส์ Perter Adventure

จัดทำโดย

6404062610090 นันทินี แสวงโชคพาหะ

เสนอ

อาจารย์สถิตย์ ประสัมพันธ์

ภาควิชาวิทยการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

คณะวิทยาศาสตร์ประยุกต์ สาขาวิทยาการคอมพิวเตอร์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ปีการศึกษา 2564

บทที่ 1

บทนำ

รายงานเล่มนี้จัดทำขึ้นมาเพื่อเป็นส่วนหนึ่งของวิชา “OOP” (Object-Oriented Programming) เพื่อให้ได้ศึกษาความรู้ใน เรื่องโปรแกรมเชิงวัตถุ กับภาษาจาวาในการวิจัยและพัฒนาโปรเจคเพื่อฝึกทักษะ เสริมความเข้าใจในการเขียนโปรแกรม ผู้จัดทำหวังว่า รายงานเล่มนี้จะเป็นประโยชน์กับผู้อ่าน หรือนักศึกษาและอาจารย์ ที่กำลังหาข้อมูลเรื่องนี้อยู่ หากมีข้อแนะนำหรือผิดพลาดประการใด ผู้จัดทำขออภัยมา ณ ที่นี้ ด้วยขอบคุณค่ะ

ที่มาและความสำคัญของโครงงาน

รายงานนี้เป็นส่วนหนึ่งของวิชา การเขียนโปรแกรมเชิงวัตถุ ซึ่งที่มาของโปรเจคเกมส์ Perter Adventureคือ ชอบเกมส์แนว RPG เลยมีแนวคิดที่จะลองทำเกมส์แนว RPG ขึ้นมาเอง ดังนั้นผู้สร้าง จึงได้ทำการสร้างเกมที่มีชื่อว่า Perter Adventure

ความสำคัญของโปรเจคนี้คือจะทำให้ตัวผู้สร้างได้รู้ว่าเกี่ยวการเขียนโปรแกรมเชิงวัตถุ ในแต่ละส่วนประกอบของเกมแบ่งไปด้วยอะไรบ้าง และกว่าจะทำเกมส์ เกมส์นึงได้ ต้องใช้ความลำบากขนาดไหนในการสร้าง ซึ่งโปรเจคนี้ก็ถือว่าเป็นการเพิ่มความสามารถในการเขียนโปรมแกรมของผู้สร้างเป็นอย่างมาก

ประเภทโครงงาน

เป็นโครงงานตามสาระการเรียนรู้

ประโยชน์ที่คาดว่าจะได้รับ

ได้เรียนรู้เกี่ยวกับการ ทำเกมส์และ ได้ทักษะการเขียนแบบเป็นระเบียบมากขึ้นและได้ทักษะการเขียนโค้ด java แบบเชิงวัตถุที่รู้ว่า อะไรควรจัดอยู่ในส่วนไหนบ้าง อย่างเกมส์ Perter Adventure จะมีแยกส่วนของ player, npc, monster และ อื่นๆ ซึ่งในแต่ละส่วนนั้นจะมีส่วนของ Function ต่างๆ ด้วย ความเชื่อมต่อของวิชาทำให้มองเห็นว่า สามารถนำไปปรับ กับ mobile app ได้ด้วย จากการเขียนโปรแกรมเชิงวัตถุ

บทที่ 2

ส่วนการพัฒนา

เนื้อเรื่อง

เมื่อ Peter ที่เป็นนักผจญภัย ได้รับคำสั่งจาก เจ้าชาย ให้ออกตามหา เจ้าหญิงที่หายตัวไปในป่า ขณะที่เดินทางกลับจากอาณาจักรเพื่อนบ้าน แต่ระหว่างทางก็พบมอนสเตอร์และอุปสรรคมากมาย

วิธีการเล่น

กด W A S D ในการบังคับทิศทาง

กด Enter เพื่อยืนยัน หรือ โจมตี

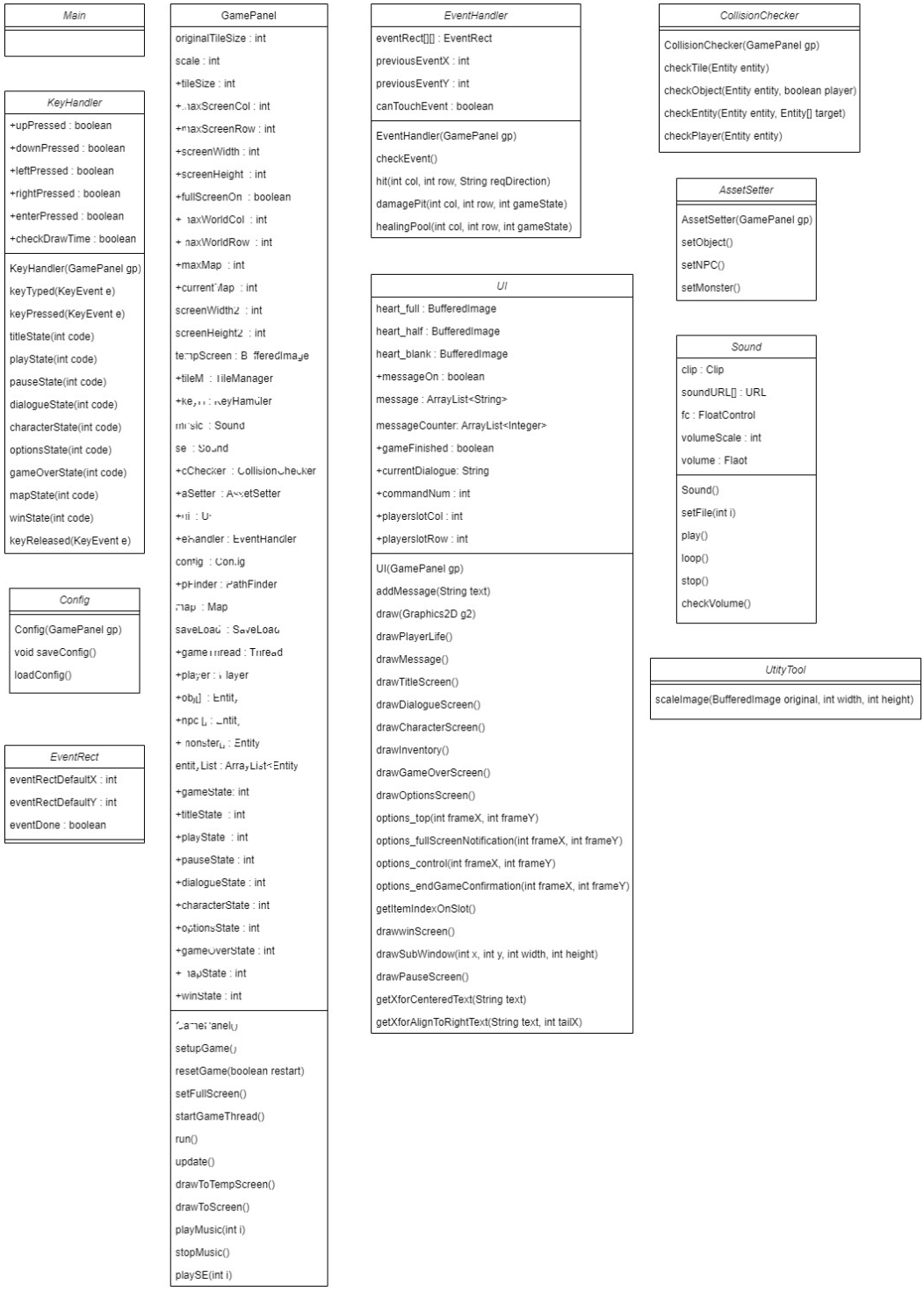
กด E ในการเปิด inventory

กด P เพื่อหยุดเกมส์

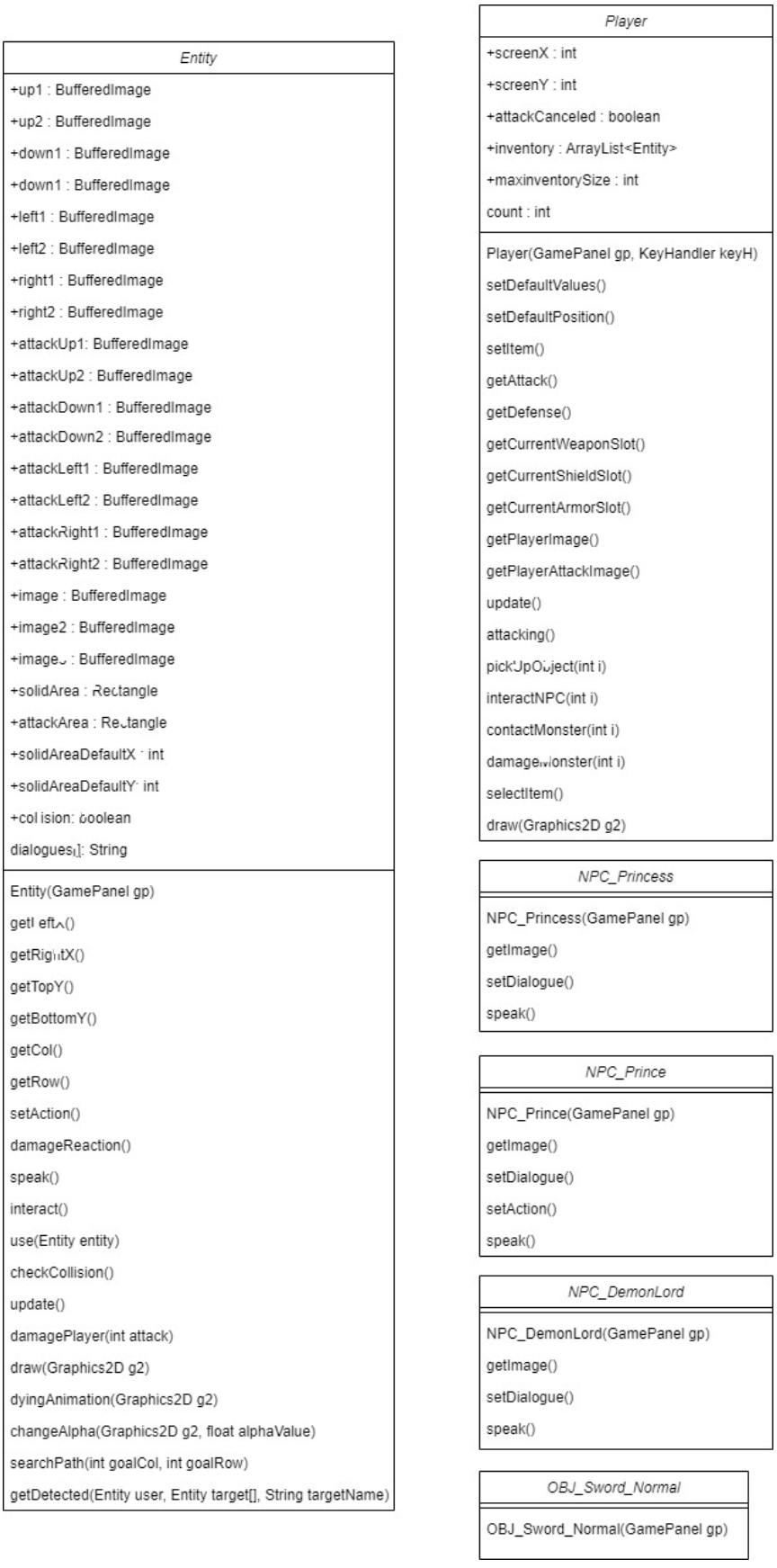
กด Esc เพื่อตั้งค่า

กด M ในการเปิดแผนที่

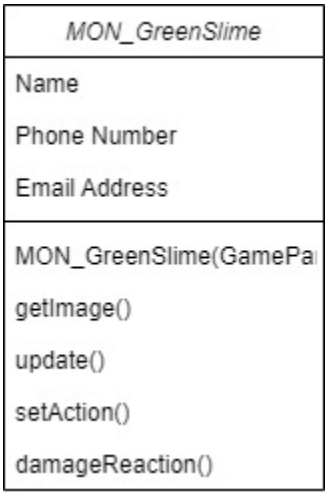
1.คลาสไดอะแกรม



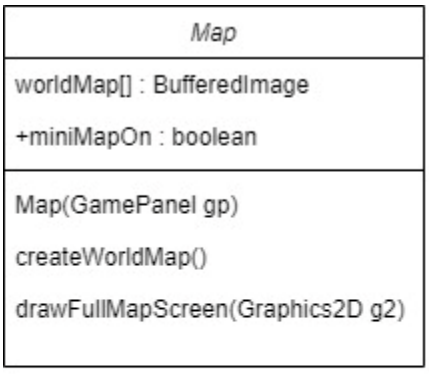
คลาสไดอะแกรมของ package main **มี** class GamePanel **ที่**extends JPanel **และ** imolements Runnable

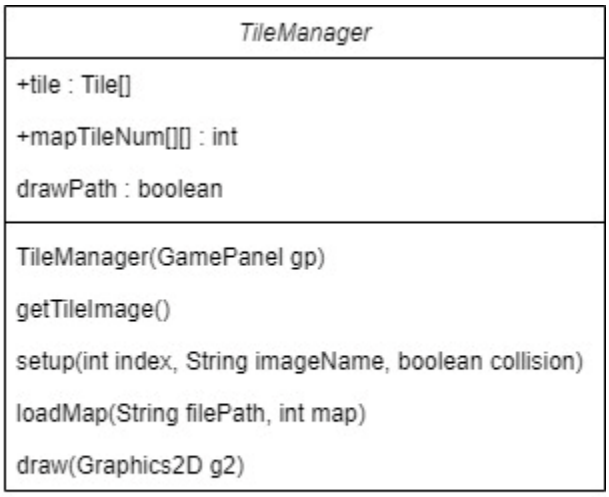

Entity
+up1 : BufferedImage
+up2 : BufferedImage
+down1 : BufferedImage
+down1 : BufferedImage
+left1 : BufferedImage
+left2 : BufferedImage
+right1 : BufferedImage
+right2 : BufferedImage
+attackUp1: BufferedImage
+attackUp2 : BufferedImage
+attackDown1 : BufferedImage
+attackDown2 : BufferedImage
+attackLeft1 : BufferedImage
+attackLeft2 : BufferedImage
+attackRight1 : BufferedImage
+attackRight2 : BufferedImage
+image : BufferedImage
+image2 : BufferedImage
+image3 : BufferedImage
+solidArea : Rectangle
+attackArea : Rectangle
+solidAreaDefaultX : int
+solidAreaDefaultY : int
+collision: boolean
dialogues[]: String
Entity(GamePanel gp)
getLeftX()
getRightX()
getTopY()
getBottomY()
getCol()
getRow()
setAction()
damageReaction()
speak()
interact()
use(Entity entity)
checkCollision()
update()
damagePlayer(int attack)
draw(Graphics2D g2)
dyingAnimation(Graphics2D g2)
changeAlpha(Graphics2D g2, float alphaValue)
searchPath(int goalCol, int goalRow)
getDetected(Entity user, Entity target[], String targetName)
Player
+screenX : int
+screenY : int
+attackCanceled : boolean
+inventory : ArrayList<Entity>
+maxinventorySize : int
count : int
Player(GamePanel gp, KeyHandler keyH)
setDefaultValues()
setDefaultPosition()
setItem()
getAttack()
getDefense()
getCurrentWeaponSlot()
getCurrentShieldSlot()
getCurrentArmorSlot()
getPlayerImage()
getPlayerAttackImage()
update()
attacking()
pickUpObject(int i)
interactNPC(int i)
contactMonster(int i)
damageMonster(int i)
selectItem()
draw(Graphics2D g2)
NPC_Princess
NPC_Princess(GamePanel gp)
getImage()
setDialogue()
speak()
NPC_Prince
NPC_Prince(GamePanel gp)
getImage()
setDialogue()
setAction()
speak()
NPC_DemonLord
NPC_DemonLord(GamePanel gp)
getImage()
setDialogue()
speak()
OBJ_Sword_Normal
OBJ_Sword_Normal(GamePanel gp)

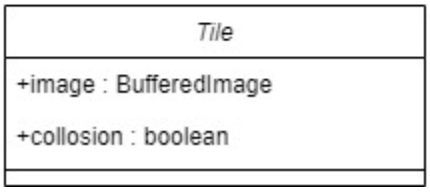
คลาสไดอะแกรมของ package entity มี class Entity จะประกอบไปด้วย class NPC_DemonLord , NPC_Prince , NPC_Princess, Player ที่สืบทอดมาจาก Entity



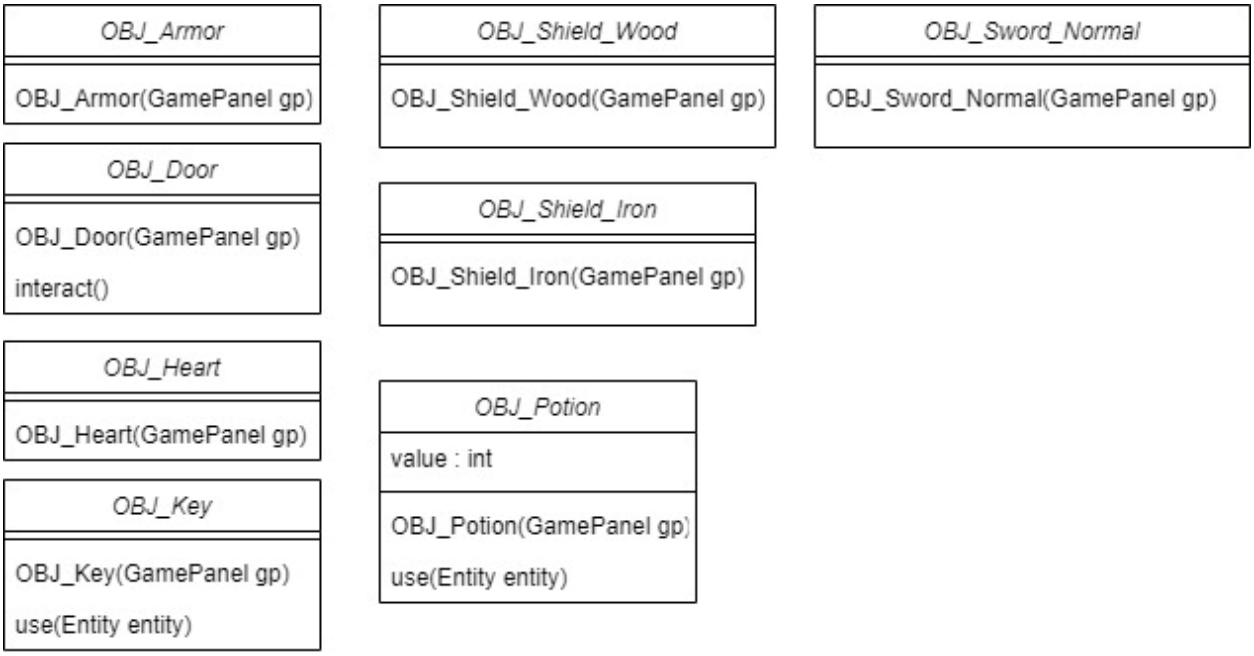
คลาสไดอะแกรมของ package monster มี class MON_GreenSlime ที่สืบทอดมาจาก Entity



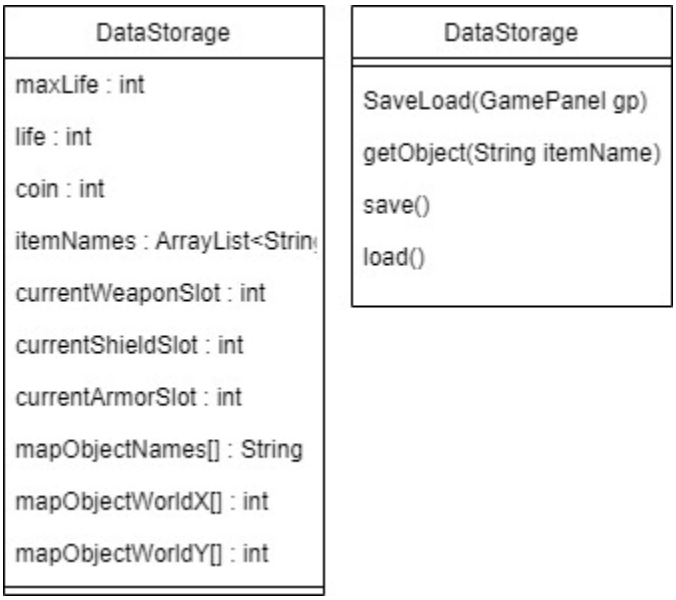




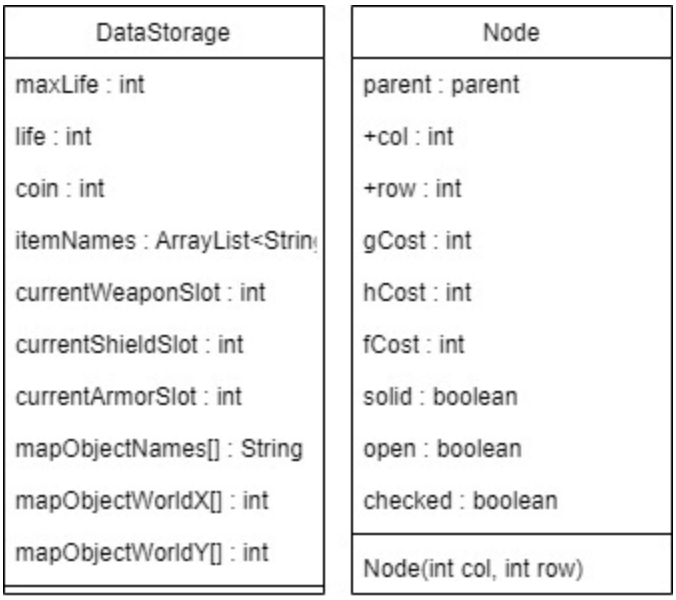
คลาสไดอะแกรมของ package tile มี class Tile , class TileManager, class Map ซึ่ง class Map สืบทอดมาจาก TileManager



คลาสไดอะแกรมของ package object **มี** class OBJ_Armor , class OBJ_Door, class OBJ_Heart, class OBJ_Key, class OBJ_Potion, class OBJ_Shield_Iron, class OBJ_Shield_Wood, class OBJ_Sword_Normal **ซึ่งสืบทอดมาจาก** Entity



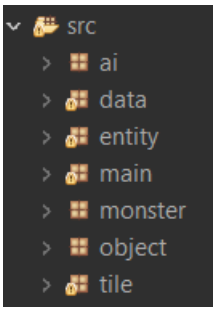
คลาสไดอะแกรมของ package data **มี** class DataStorage, class SaveLoad **เอาไว้** save **และ** load data **ที่เก็บไว้**



คลาสไดอะแกรมของ package ai **มี** class Node, class PathFinder

ส่วนของโปรแกรม

2.อธิบายส่วนของโปรแกรมที่มี



ในการสร้างเกมส์นี้ จะมี package ทั้งหมด 7 package

1.ai คือ ทำให้ monster เดินเข้าหา player

2.data คือ เอาไว้เก็บ data ในเกมส์ และ save, load data เข้ามาใช้(save เพื่อเล่นต่อ)

3.entity คือ เอาไว้เก็บ NPC และ Player

4. main คือ ระบบเกมหลักๆทั้งหมด เช่น การรับค่าจากปุ่ม การแสดงผลต่างๆ

5.monster คือ เก็บ monster ทั้งหมดภายในเกมส์

6.object คือ เก็บ object ทั้งหมดภายในเกมส์

7.tile คือ เอาไว้จัดการและโหลด Map

Class Node

```java
package ai;

public class Node {

    Node parent;
    public int col;
    public int row;
    int gCost;
    int hCost;
    int fCost;
    boolean solid;
    boolean open;
    boolean checked;

    public Node(int col, int row) {
        this.col = col;
        this.row = row;
    }
}
```

ใน class Node จะเก็บตัวแปรที่ใช้ใน package ai ไว้ทั้งหมด

Class PathFinder

```java
package ai;

import java.util.ArrayList;

public class PathFinder {

    GamePanel gp;
    Node[][] node;
    ArrayList<Node> openList = new ArrayList<>();
    public ArrayList<Node> pathList = new ArrayList<>();
    Node startNode, goalNode, currentNode;
    boolean goalReached = false;
    int step = 0;

    public PathFinder(GamePanel gp) {
        this.gp = gp;
        instantiateNodes();
    }
    public void instantiateNodes() {

        node = new Node[gp.maxWorldCol][gp.maxWorldRow];

        int col = 0;
        int row = 0;

        while(col < gp.maxWorldCol && row < gp.maxWorldRow) {

            node[col][row] = new Node(col,row);

            col++;
            if(col == gp.maxWorldCol) {
                col = 0;
                row++;
            }
        }
    }
```

ใน class PathFinder เราจะสร้าง constructor เพื่อสร้างNode, รีเซ็ตNode, เซ็ตค่าของ Node, search Node, getCost, หาPath และ เปิดNode

```
39    public void resetNodes() {
40
41        int col = 0;
42        int row = 0;
43
44        while(col < gp.maxWorldCol && row < gp.maxWorldRow) {
45
46            //Reset open, checked and solid state
47            node[col][row].open = false;
48            node[col][row].checked = false;
49            node[col][row].solid = false;
50
51            col++;
52            if(col == gp.maxWorldCol) {
53                col = 0;
54                row++;
55            }
56        }
57
58        //Reset other settings
59        openList.clear();
60        pathList.clear();
61        goalReached = false;
62        step = 0;
63    }
64    public void setNodes(int startCol, int startRow, int goalCol, int goalRow) {
65
66        resetNodes();
67
68        //Set Start and Goal node
69        startNode = node[startCol][startRow];
70        currentNode = startNode;
71        goalNode = node[goalCol][goalRow];
72        openList.add(currentNode);
73
74        int col = 0;
75        int row = 0;
```

resetNodes คือ รีเซ็ตค่าของ Node ให้เป็นค่าเริ่มต้น

```
77          while(col < gp.maxWorldCol && row < gp.maxWorldRow) {
78
79              //SET SOLID NODE
80              //CHECK TILES
81              int tileNum = gp.tileM.mapTileNum[col][row];
82              if(gp.tileM.tile[tileNum].collosion == true) {
83                  node[col][row].solid = true;
84              }
85              //SET COST
86              getCost(node[col][row]);
87
88              col++;
89              if(col == gp.maxWorldCol) {
90                  col = 0;
91                  row++;
92              }
93          }
94      }
95      public void getCost(Node node) {
96
97          //G cost
98          int xDistance = Math.abs(node.col - startNode.col);
99          int yDistance = Math.abs(node.row - startNode.row);
100         node.gCost = xDistance + yDistance;
101         //H cost
102         xDistance = Math.abs(node.col - goalNode.col);
103         yDistance = Math.abs(node.row - goalNode.row);
104         node.hCost = xDistance + yDistance;
105         //F cost
106         node.fCost = node.gCost + node.hCost;
107     }
108     public boolean search() {
109
110         while(goalReached == false && step < 500) {
111
112             int col = currentNode.col;
113             int row = currentNode.row;
```

setNodes คือ เซ็ตค่าของ Node ทั้ง จุดเริ่ม จุดปัจจุบัน และจุบสิ้นสุด เป็นค่าต่างๆ

getCost คือ หาค่าของ G Cost, H Cost, F Cost และเก็บค่าที่ได้ไว้

```
115             //Check the current node
116             currentNode.checked = true;
117             openList.remove(currentNode);
118 
119             //Open the up node
120             if(row - 1 >= 0) {
121                 openNode(node[col][row-1]);
122             }
123             //Open the left node
124             if(col - 1 >= 0) {
125                 openNode(node[col - 1][row]);
126             }
127             //Open the down node
128             if(row + 1 < gp.maxWorldRow) {
129                 openNode(node[col][row + 1]);
130             }
131             //Open the right node
132             if(col + 1 < gp.maxWorldCol) {
133                 openNode(node[col + 1][row]);
134             }
135 
136             //Find the best node
137             int bestNodeIndex = 0;
138             int bestNodefCost = 0;
139 
140             for(int i=0;i<openList.size();i++) {
141 
142                 //Check if this node's F cost is batter
143                 if(openList.get(i).fCost < bestNodefCost) {
144                     bestNodeIndex = i;
145                     bestNodefCost = openList.get(i).fCost;
146                 }
147                 //If F cost is equal, check the G cost
148                 else if(openList.get(i).fCost == bestNodefCost) {
149                     if(openList.get(i).gCost < openList.get(bestNodefCost).gCost) {
150                         bestNodeIndex = i;
151                     }
152                 }
```

```
153             }
154             //If there is no node in the openList, end the loop
155             if(openList.size() == 0) {
156                 break;
157             }
158             //After the loop, openList[bestNodeIndex] is the next step (=currentN
159             currentNode = openList.get(bestNodeIndex);
160 
161             if(currentNode == goalNode) {
162                 goalReached = true;
163                 trackThePath();
164             }
165             step++;
166         }
167 
168         return goalReached;
169     }
170     public void trackThePath() {
171 
172         Node current = goalNode;
173 
174         while(current != startNode) {
175 
176             pathList.add(0, current);
177             current = current.parent;
178         }
179     }
180     public void openNode(Node node) {
181 
182         if(node.open == false && node.checked == false && node.solid == false) {
183 
184             node.open = true;
185             node.parent = currentNode;
186             openList.add(node);
187         }
188     }
189 }
190 
```

search คือ ค้นหา Node ที่ใกล้ที่สุดที่สามารถเดินได้

trackThePath คือ ติดตาม Path ปัจจุบัน

openNode คือ เช็ค Node ว่าเป็น openNode ไหม ถ้าเป็นจะเก็บลงใน OpenList

```java
package data;

import java.io.Serializable;

public class DataStorage implements Serializable {

    //PLAYER START
    int maxLife;
    int life;
    int coin;

    // PLAYER INVENTORY
    ArrayList<String> itemNames = new ArrayList<>();
    int currentWeaponSlot;
    int currentShieldSlot;
    int currentArmorSlot;

    //OBJECT ON MAP
    String mapObjectNames[];
    int mapObjectWorldX[];
    int mapObjectWorldY[];
}
```

ใน class DataStorage เราจะเอาไว้เก็บตัวแปร ที่จะใช้ใน class SaveLoad

```java
package data;

import java.io.File;

public class SaveLoad {

    GamePanel gp;

    public SaveLoad(GamePanel gp) {

        this.gp = gp;
    }
    public Entity getObject(String itemName) {

        Entity obj = null;

        switch(itemName) {
        case "Iron Shield": obj = new OBJ_Shield_Iron(gp); break;
        case "Key": obj = new OBJ_Key(gp); break;
        case "Potion": obj = new OBJ_Potion(gp); break;
        case "Door": obj = new OBJ_Door(gp); break;
        case "Heart": obj = new OBJ_Heart(gp); break;
        case "Wood Shield":  obj = new OBJ_Shield_Wood(gp); break;
        case "Normal Sword":  obj = new OBJ_Sword_Normal(gp); break;
        case "Iron Armor":  obj = new OBJ_Armor(gp); break;
        }
        return obj;
    }
    public void save() {

        try {
            ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(new File("save.dat")));

            DataStorage ds = new DataStorage();

            ds.maxLife = gp.player.maxLife;
            ds.life = gp.player.life;
            ds.coin = gp.player.coin;
```

ใน class SaveLoad เราจะสร้าง constructor เพื่อSaveข้อมูลและLoadข้อมูล โดย getObject จะรับชื่อของ object นั้นๆมาแล้วแปลงเป็น object

```
            //PLAYER INVENTORY
            for(int i=0;i<gp.player.inventory.size();i++) {
                ds.itemNames.add(gp.player.inventory.get(i).name);
            }
            //PLAYER EQUIPMENT
            ds.currentWeaponSlot = gp.player.getCurrentWeaponSlot();
            ds.currentShieldSlot = gp.player.getCurrentShieldSlot();
            ds.currentArmorSlot = gp.player.getCurrentArmorSlot();

            //OBJECT ON MAP
            ds.mapObjectNames = new String[gp.obj.length];
            ds.mapObjectWorldX = new int[gp.obj.length];
            ds.mapObjectWorldY = new int[gp.obj.length];

            for(int i = 0;i < gp.obj.length; i++) {

                if(gp.obj[i] == null) {
                    ds.mapObjectNames[i] = "NA";
                }
                else {
                    ds.mapObjectNames[i] = gp.obj[i].name;
                    ds.mapObjectWorldX[i] = gp.obj[i].worldX;
                    ds.mapObjectWorldY[i] = gp.obj[i].worldY;
                }
            }

            //Write the DataStorage object
            oos.writeObject(ds);
        }
        catch(Exception e){
            System.out.println("Save Exception!");
        }
    }
    public void load() {

        try {
            ObjectInputStream ois = new ObjectInputStream(new FileInputStream(new File("save.dat")));
```

Save จะทำการเขียนข้อมูลลงใน file save.dat

```
            DataStorage ds = (DataStorage)ois.readObject();

            gp.player.maxLife = ds.maxLife;
            gp.player.life = ds.life;
            gp.player.coin = ds.coin;

            //PLAYER INVENTORY
            gp.player.inventory.clear();
            for(int i=0;i<ds.itemNames.size();i++) {
                gp.player.inventory.add(getObject(ds.itemNames.get(i)));
            }
            //PLAYER REQUIPMENT
            gp.player.currentWeapon = gp.player.inventory.get(ds.currentWeaponSlot);
            gp.player.currentShield = gp.player.inventory.get(ds.currentShieldSlot);
            gp.player.currentArmor = gp.player.inventory.get(ds.currentArmorSlot);
            gp.player.getAttack();
            gp.player.getDefense();
            gp.player.getPlayerAttackImage();

            //OBJECT ON MAP
            for(int i = 0;i < gp.obj.length; i++) {

                if(ds.mapObjectNames[i].equals("NA")) {
                    gp.obj[i] = null;
                }
                else {
                    gp.obj[i] = getObject(ds.mapObjectNames[i]);
                    gp.obj[i].worldX = ds.mapObjectWorldX[i];
                    gp.obj[i].worldY = ds.mapObjectWorldY[i];
                }
            }

        } catch (Exception e) {
            // TODO: handle exception
            System.out.println("Load Exception!");
        }
    }
}
```

Load จะทำการอ่านข้อมูลภายใน file save.dat

```java
package entity;

import java.awt.AlphaComposite;

public class Entity {

    GamePanel gp;
    public  BufferedImage up1, up2, down1, down2, left1, left2, right1, right2;
    public BufferedImage attackUp1, attackUp2, attackDown1, attackDown2
    , attackLeft1, attackLeft2, attackRight1, attackRight2;
    public BufferedImage image, image2, image3;
    public Rectangle solidArea = new Rectangle(0, 0, 48, 48);
    public Rectangle attackArea = new Rectangle(0, 0, 0, 0);
    public int solidAreaDefaultX, solidAreaDefaultY;
    public boolean collision = false;
    String dialogues[] = new String[20];

    //STATE
    public int worldX, worldY;
    public String direction = "down";
    public int spriteNum = 1;
    int dialogueIndex = 0;
    public boolean collisionOn = false;
    public boolean invincible = false;  // เวลาโดน monster ตีจะไม่โดนตาเมจสักพัก
    boolean attacking = false;
    public boolean alive = true;
    public boolean dying = false;
    boolean hpBarOn = false;
    public boolean onPath = false;

    //COUNTER
    public int spriteCounter = 0;
    public int actionLockCounter = 0;
    public int invincibleCounter = 0;
    int dyingCounter = 0;
    int hpBarCounter = 0;
```

```
47      //CHARACTER ATTIBUTES
48      public String name;
49      public int speed;
50      public int maxLife;
51      public int life;
52      public int coin;
53      public int attack;
54      public int defense;
55      public Entity currentWeapon;
56      public Entity currentShield;
57      public Entity currentArmor;
58
59      //ITEM ATTRIBUTES
60      public int attackValue;
61      public int defenseValue;
62      public String description = "";
63
64      //TYPE
65      public int type;    // 0 = player, 1 = npc, 2 = monster
66      public final int type_player = 0;
67      public final int type_npc = 1;
68      public final int type_monster = 2;
69      public final int type_sword = 3;
70      public final int type_armor = 4;
71      public final int type_shield = 5;
72      public final int type_consumable = 6;
73      public final int type_pickupOnly = 7;
74      public final int type_obstacle = 8;
```

ประกาศตัวแปร

```
public Entity(GamePanel gp) {
    this.gp = gp;
}
public int getLeftX() {
    return worldX + solidArea.x;
}
public int getRightX() {
    return worldX + solidArea.x + solidArea.width;
}
public int getTopY() {
    return worldY + solidArea.y;
}
public int getBottomY() {
    return worldY + solidArea.y + solidArea.height;
}
public int getCol() {
    return (worldX + solidArea.x)/gp.tileSize;
}
public int getRow() {
    return (worldY + solidArea.y)/gp.tileSize;
}
public void setAction() {}
public void damageReaction() {}
public void speak() {
    if(dialogues[dialogueIndex] == null) {
        dialogueIndex = 0;
    }
    gp.ui.currentDialogue = dialogues[dialogueIndex];
    dialogueIndex++;

    switch(gp.player.direction) {
    case "down":
        direction = "up";
        break;
    case "up":
        direction = "down";
        break;
    case "left":
```

getLeftX, getRightX, getTopY, getBottomY, getCol, getRow คำนวนหาตำแหน่งนั้นๆ

setAction เป็น abstract ที่เอาไว้เรียกใช้เมื่อต้องการให้ npc หรือ monster มีการเคลื่อนที่

damageReaction เป็น abstract ที่เอาไว้เรียกใช้เมื่อต้องการให้ monster มี reaction เมื่อได้รับดาเมจ

```
113         case "left":
114             direction = "right";
115             break;
116         case "right":
117             direction = "left";
118             break;
119         }
120     }
121     public void interact() {}
122     public boolean use(Entity entity) {return false;}
123     public void checkCollision() {
124 
125         collisionOn = false;
126         gp.cChecker.checkTile(this);
127         gp.cChecker.checkObject(this, false);
128         gp.cChecker.checkEntity(this, gp.npc);
129         gp.cChecker.checkEntity(this, gp.monster);
130         boolean contactPlayer = gp.cChecker.checkPlayer(this);
131 
132         if(this.type == type_monster && contactPlayer == true) {
133             damagePlayer(attack);
134         }
135     }
136     public void update() {
137 
138         setAction();
139         checkCollision();
140 
141         //IF COLLISION IS FALSE, PLAYER CAN MOVE
142         if(collisionOn == false) {
143             switch(direction) {
144             case "up": worldY -= speed; break;
145             case "down": worldY += speed; break;
146             case "left": worldX -= speed; break;
147             case "right": worldX += speed; break;
148             }
149         }
150 
```

Speak เมื่อกด enter ที่ npc จะทำการเรียกใช้ speak แสดง ข้อความของ npc

Interact เป็น abstract ที่เอาไว้เรียกใช้ใน class player

Use สร้างไว้เพื่อเรียกใช้ใน class player

checkCollision เช็คว่าชนหรือไม่ จะเช็ค tile, object, npc, monster และ player;

```
        spriteCounter++;
        if(spriteCounter > 10) {
            if(spriteNum == 1) {
                spriteNum = 2;
            }
            else if(spriteNum == 2) {
                spriteNum = 1;
            }
            spriteCounter = 0;
        }
        if(invincible == true) {
            invincibleCounter++;
            if(invincibleCounter > 40) {
                invincible = false;
                invincibleCounter = 0;
            }
        }
    }
    public void damagePlayer(int attack) {

        if(gp.player.invincible == false) {
            // we can give damage
            gp.playSE(6);

            int damage = attack - gp.player.defense;
            if(damage < 0) {
                damage = 0;
            }
            gp.player.life -= damage;
            gp.player.invincible = true;
        }
    }
    public void draw(Graphics2D g2) {

        BufferedImage image = null;
        int screenX = worldX - gp.player.worldX + gp.player.screenX;
        int screenY = worldY - gp.player.worldY + gp.player.screenY;
```

Update จะใช้เพื่ออัพเดตสิ่งต่างๆ ใน update จะเรียกใช้ setAction และ checkCollision และเช็คว่าถ้า ไม่ชน player จะเดินได้

damagePlayer จะเช็คว่า player invincible(สถานะที่จะไม่โดนโจมตี) หรือไม่ ถ้าไม่ โดยดาเมจได้

```
        if(worldX + gp.tileSize > gp.player.worldX - gp.player.screenX &&
           worldX - gp.tileSize < gp.player.worldX + gp.player.screenX &&
           worldY + gp.tileSize > gp.player.worldY - gp.player.screenY &&
           worldY - gp.tileSize < gp.player.worldY + gp.player.screenY) {

            switch(direction) {
            case "up":
              if(spriteNum == 1) {image = up1;}
              if(spriteNum == 2) {image = up2;}
                break;
            case "down":
              if(spriteNum == 1) {image = down1;}
              if(spriteNum == 2) {image = down2;}
                break;
            case "left":
              if(spriteNum == 1) {image = left1;}
              if(spriteNum == 2) {image = left2;}
                break;
            case "right":
              if(spriteNum == 1) {image = right1;}
              if(spriteNum == 2) {image = right2;}
                break;
            }

            //Monster HP bar
            if(type == 2 && hpBarOn == true) {

                double oneScale = (double)gp.tileSize/maxLife;
                double hpBarValue = oneScale*life;

                g2.setColor(new Color(35, 35, 35));
                g2.fillRect(screenX - 1, screenY - 16, gp.tileSize + 2, 12);

                g2.setColor(new Color(255, 0, 30));
                g2.fillRect(screenX, screenY - 15, (int)hpBarValue, 10);

                hpBarCounter++;
```

Draw ไว้วาดแสดงบนจอทั้ง animation การเดิน แถบเลือดของ monster

```
public void dyingAnimation(Graphics2D g2) {

    dyingCounter++;

    int i = 5;

    if(dyingCounter <= i) {changeAlpha(g2, 0f);}
    if(dyingCounter > i && dyingCounter <= i*2) {changeAlpha(g2, 1f);}
    if(dyingCounter > i*2 && dyingCounter <= i*3) {changeAlpha(g2, 0f);}
    if(dyingCounter > i*3 && dyingCounter <= i*4) {changeAlpha(g2, 1f);}
    if(dyingCounter > i*4 && dyingCounter <= i*5) {changeAlpha(g2, 0f);}
    if(dyingCounter > i*5 && dyingCounter <= i*6) {changeAlpha(g2, 1f);}
    if(dyingCounter > i*6 && dyingCounter <= i*7) {changeAlpha(g2, 0f);}
    if(dyingCounter > i*7 && dyingCounter <= i*8) {changeAlpha(g2, 1f);}
    if(dyingCounter > i*8) {
        alive = false;
    }
}
public void changeAlpha(Graphics2D g2, float alphaValue) {

    g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, alphaValue));
}
public BufferedImage setup(String imagePath, int width, int height) {

    UtityTool uTool = new UtityTool();
    BufferedImage image = null;

    try {
        image = ImageIO.read(getClass().getResourceAsStream(imagePath + ".png"));
        image = uTool.scaleImage(image, width, height);

    } catch (Exception e) {
        e.printStackTrace();
    }
    return image;
}
```

dyingAnimation คือ แสดง animation การตาย

setup คือ setค่าเริ่มต้นโหลดภาพ

```java
public void searchPath(int goalCol, int goalRow) {

    int startCol = (worldX + solidArea.x)/gp.tileSize;  //Find where player is
    int startRow = (worldY + solidArea.y)/gp.tileSize;  //Find where player is

    gp.pFinder.setNodes(startCol, startRow, goalCol, goalRow);

    if(gp.pFinder.search() == true) {

        //Next worldX & WorldY
        int nextX = gp.pFinder.pathList.get(0).col * gp.tileSize;
        int nextY = gp.pFinder.pathList.get(0).row * gp.tileSize;

        //Entity's solidArea position
        int enLeftX = worldX + solidArea.x;
        int enRightX = worldX + solidArea.x + solidArea.width;
        int enTopY = worldY + solidArea.y;
        int enBottomY = worldY + solidArea.y + solidArea.height;

        if(enTopY > nextY && enLeftX >= nextX && enRightX < nextX + gp.tileSize) {
            direction = "up";
        }
        else if(enTopY < nextY && enLeftX >= nextX && enRightX < nextX + gp.tileSize) {
            direction = "down";
        }
        else if(enTopY >= nextY && enBottomY < nextY + gp.tileSize) {
            //left or right
            if(enLeftX > nextX) {
                direction = "left";
            }
            if(enLeftX < nextX) {
                direction = "right";
            }
        }
        else if(enTopY > nextY && enLeftX > nextX) {
            //up or left
            direction = "up";
            checkCollision();
```

searchPath ค้นหาเส้นทางการเดินที่ใกล้เป้าหมายที่สุด

```java
public int getDetected(Entity user, Entity target[], String targetName) {

    int index = 999;    //detected door for use key

    //check the surrounding object
    int nextWorldX = user.getLeftX();
    int nextWorldY = user.getTopY();

    switch(user.direction) {
    case "up": nextWorldY = user.getTopY()-gp.player.speed; break;
    case "down": nextWorldY = user.getBottomY()+gp.player.speed; break;
    case "left": nextWorldX = user.getLeftX()-gp.player.speed; break;
    case "right": nextWorldX = user.getRightX()+gp.player.speed; break;
    }
    int col = nextWorldX/gp.tileSize;
    int row = nextWorldY/gp.tileSize;

    for(int i=0; i<target.length;i++) {
        if(target[i] != null) {
            if(target[i].getCol() == col &&
                    target[i].getRow() == row &&
                    target[i].name.equals(targetName)) {

                index = i;
                break;
            }
        }
    }
    return index;
}
```

getDetected เช็คว่า player อยู่ใกล้ๆตำแหน่งของobjectไหม

```java
package entity;

import main.GamePanel;

public class NPC_DemonLord  extends Entity {

    public NPC_DemonLord(GamePanel gp) {
        super(gp);

        direction = "left";
        speed = 0;

        getImage();
        setDialogue();
    }

    public void getImage() {

        up1 = setup("/npc/demon_lord",gp.tileSize,gp.tileSize);
        up2 = setup("/npc/demon_lord",gp.tileSize,gp.tileSize);
        down1 = setup("/npc/demon_lord",gp.tileSize,gp.tileSize);
        down2 = setup("/npc/demon_lord",gp.tileSize,gp.tileSize);
        left1 = setup("/npc/demon_lord",gp.tileSize,gp.tileSize);
        left2 = setup("/npc/demon_lord",gp.tileSize,gp.tileSize);
        right1 = setup("/npc/demon_lord",gp.tileSize,gp.tileSize);
        right2 = setup("/npc/demon_lord",gp.tileSize,gp.tileSize);

    }

    public void setDialogue() {

        dialogues[0] = "Hello adventurer...\nWhy you come here?";
        dialogues[1] = "Come to save the princess from me?!";
        dialogues[2] = "I'm sure there must be a misunderstanding.";
        dialogues[3] = "Come to save the princess?";
        dialogues[4] = "I didn't kidnap the princess.";
        dialogues[5] = "If you don't believe it, ask the princess.";
    }
```

Class NPC_DEmonLord มีการ extends Entity และกำหนดค่าต่างๆทั้ง ภาพของnpc และบทพูด

```java
package entity;

import java.util.Random;

public class NPC_Prince extends Entity {

    public NPC_Prince(GamePanel gp) {
        super(gp);

        direction = "left";
        speed = 1;

        getImage();
        setDialogue();
    }
    public void getImage() {

        up1 = setup("/npc/prince_down",gp.tileSize,gp.tileSize);
        up2 = setup("/npc/prince_down",gp.tileSize,gp.tileSize);
        down1 = setup("/npc/prince_down",gp.tileSize,gp.tileSize);
        down2 = setup("/npc/prince_down",gp.tileSize,gp.tileSize);
        left1 = setup("/npc/prince_left1",gp.tileSize,gp.tileSize);
        left2 = setup("/npc/prince_left2",gp.tileSize,gp.tileSize);
        right1 = setup("/npc/prince_right1",gp.tileSize,gp.tileSize);
        right2 = setup("/npc/prince_right2",gp.tileSize,gp.tileSize);
    }
    public void setDialogue() {

        dialogues[0] = "Hello, Are you Peter right?";
        dialogues[1] = "I called you because I need your help.";
        dialogues[2] = "The princess was kidnapped to the \nDemon Lord's castle.";
        dialogues[3] = "And Peter, you are descendant of the Hero.";
        dialogues[4] = "So may I ask you to help the princess \ncome back?";
    }
    public void setAction() {

        actionLockCounter ++;
```

Class NPC_Prince มีการ extends Entity และกำหนดค่าต่าง ๆทั้ง ภาพของnpc และบทพูด

```
public void setAction() {

    actionLockCounter ++;

    if(actionLockCounter == 120) {

        Random random = new Random();
        int i = random.nextInt(50)+1; //pick up a number from 1 to 100

        if(i <= 25) {
            direction = "right";
        }
        if(i > 25 && i<=50) {
            direction = "left";
        }
        actionLockCounter = 0;
    }
}
public void speak() {

    super.speak();
}
```

มีการ setAction ทำให้ npc เดินไปมา

```java
public class NPC_Princess extends Entity {

    public NPC_Princess(GamePanel gp) {
        super(gp);

        direction = "left";
        speed = 0;

        getImage();
        setDialogue();
    }

    public void getImage() {

        up1 = setup("/npc/princess",gp.tileSize,gp.tileSize);
        up2 = setup("/npc/princess",gp.tileSize,gp.tileSize);
        down1 = setup("/npc/princess",gp.tileSize,gp.tileSize);
        down2 = setup("/npc/princess",gp.tileSize,gp.tileSize);
        left1 = setup("/npc/princess",gp.tileSize,gp.tileSize);
        left2 = setup("/npc/princess",gp.tileSize,gp.tileSize);
        right1 = setup("/npc/princess",gp.tileSize,gp.tileSize);
        right2 = setup("/npc/princess",gp.tileSize,gp.tileSize);
    }

    public void setDialogue() {

        dialogues[0] = "...\nWho are you?!\nAre you an adventurer?";
        dialogues[1] = "Come to save me, Help from whom?";
        dialogues[2] = "Demon Lord didn't kidnap me.";
        dialogues[3] = "He saved me from demonic beasts.";
        dialogues[4] = "and stay here until the carriage is repaired.";
        dialogues[5] = "Today the carriage was completely repaired.\nThen I can go back with you.";
    }

    public void speak() {

        super.speak();
```

Class NPC_Princess มีการ extends Entity และกำหนดค่าต่างๆทั้ง ภาพของnpc และบทพูด

```java
package entity;

import java.awt.AlphaComposite;

public final class Player extends Entity{

    KeyHandler keyH;

    public final int screenX;
    public final int screenY;
    public boolean attackCanceled = false;
    public ArrayList<Entity> inventory = new ArrayList<>();
    public final int maxinventorySize = 20;
    int count = 0;

    public Player(GamePanel gp, KeyHandler keyH) {

        super(gp);
        this.keyH = keyH;

        screenX = gp.screenWidth/2 - (gp.tileSize/2);
        screenY = gp.screenHeight/2 - (gp.tileSize/2);

        solidArea = new Rectangle();
        solidArea.x = 8;
        solidArea.y = 16;
        solidArea.width = 32;
        solidArea.height = 32;
        solidAreaDefaultX = solidArea.x;
        solidAreaDefaultY = solidArea.y;

        //ATTACK AREA
//        attackArea.width = 36;
//        attackArea.height = 25;

        setDefaultValues();
    }
```

Class player มีการ extends Entity และ ประกาศตัวแปรที่ต้องใช้

```
public void setDefaultValues() {

    worldX = gp.tileSize * 23;
    worldY = gp.tileSize * 21;
    speed = 4;
    direction = "down";

    //PLAYER STATUS
    maxLife = 6;
    life = maxLife;
    coin = 0;
    currentWeapon = new OBJ_Sword_Normal(gp);
    currentShield = new OBJ_Shield_Wood(gp);
    currentArmor = new OBJ_Armor(gp);
    attack = getAttack();
    defense = getDefense();

    getPlayerImage();
    getPlayerAttackImage();
    setItem();
}
public void setDefaultPosition() {

    worldX = gp.tileSize * 23;
    worldY = gp.tileSize * 21;
    direction = "down";
}
public void restoreStatus() {
    life = maxLife;
    invincible = false;
    attacking = false;
}
```

setDefaultValues จะ set ค่าสถานะเริ่มต้นของ player

setDefaultPosition จะ set ตำแหน่งเริ่มต้นของ player

restoreStatus คืนค่ากลับ

```
public void setItem() {

    inventory.clear();
    inventory.add(currentWeapon);
    inventory.add(currentShield);
    inventory.add(currentArmor);
    inventory.add(new OBJ_Key(gp));
}
public int getAttack() {
    attackArea = currentWeapon.attackArea;
    return attack = currentWeapon.attackValue;
}
public int getDefense() {
    return defense = currentShield.defenseValue + currentArmor.defenseValue;
}
public int getCurrentWeaponSlot() {
    int currentWeaponSlot = 0;
    for(int i=0;i<inventory.size();i++) {
        if(inventory.get(i) == currentWeapon) {
            currentWeaponSlot = i;
        }
    }
    return currentWeaponSlot;
}
public int getCurrentShieldSlot() {
    int currentShieldSlot = 0;
    for(int i=0;i<inventory.size();i++) {
        if(inventory.get(i) == currentShield) {
            currentShieldSlot = i;
        }
    }
    return currentShieldSlot;
}
```

setItem ไว้ setitem ใน player

getAttack คำนวนค่า attack

getDefense คำนวนรับค่า Defense

getCurrentWeaponSlot, getCurrentShieldSlot, getCurrentArmor หาว่าใช้ อาวุธ, โล่ และ เกราะ อะไรอยู่

```
public int getCurrentArmorSlot() {
    int currentArmorSlot = 0;
    for(int i=0;i<inventory.size();i++) {
        if(inventory.get(i) == currentArmor) {
            currentArmorSlot = i;
        }
    }
    return currentArmorSlot;
}
public void getPlayerImage() {

    up1 = setup("/player/up1",gp.tileSize,gp.tileSize);
    up2 = setup("/player/up2",gp.tileSize,gp.tileSize);
    down1 = setup("/player/down1",gp.tileSize,gp.tileSize);
    down2 = setup("/player/down2",gp.tileSize,gp.tileSize);
    left1 = setup("/player/left1",gp.tileSize,gp.tileSize);
    left2 = setup("/player/left2",gp.tileSize,gp.tileSize);
    right1 = setup("/player/right1",gp.tileSize,gp.tileSize);
    right2 = setup("/player/right2",gp.tileSize,gp.tileSize);
}
public void getPlayerAttackImage() {

    attackUp1 = setup("/player/attack_up1",gp.tileSize,gp.tileSize*2);
    attackUp2 = setup("/player/attack_up2",gp.tileSize,gp.tileSize*2);
    attackDown1 = setup("/player/attack_down1",gp.tileSize,gp.tileSize*2);
    attackDown2 = setup("/player/attack_down2",gp.tileSize,gp.tileSize*2);
    attackLeft1 = setup("/player/attack_left1",gp.tileSize*2,gp.tileSize);
    attackLeft2 = setup("/player/attack_left2",gp.tileSize*2,gp.tileSize);
    attackRight1 = setup("/player/attack_right1",gp.tileSize*2,gp.tileSize);
    attackRight2 = setup("/player/attack_right2",gp.tileSize*2,gp.tileSize);
}
```

getPlayerImage โหลด imageการเดิน มา

getPlayerAttaclImage โหลด imageการโจมตี มา

```
public void update() {

    if(attacking == true) {
        attacking();
    }
    else if(keyH.upPressed == true || keyH.downPressed == true ||
            keyH.leftPressed == true || keyH.rightPressed == true || keyH.enterPressed == true) {
        if(keyH.upPressed == true) {
            direction = "up";
        }
        else if(keyH.downPressed == true) {
            direction = "down";
        }
        else if(keyH.leftPressed == true) {
            direction = "left";
        }
        else if(keyH.rightPressed == true) {
            direction = "right";
        }

        //CHECK TILE COLLISION
        collisionOn = false;
        gp.cChecker.checkTile(this);

        //CHECK OBJECT COLLISION
        int objIndex = gp.cChecker.checkObject(this, true);
        pickUpObject(objIndex);

        //CHECK NPC COLLISION
        int npcIndex = gp.cChecker.checkEntity(this, gp.npc);
        interactNPC(npcIndex);

        //CHECK MONSTER COLLISION
        int monsterIndex = gp.cChecker.checkEntity(this, gp.monster);
        contactMonster(monsterIndex);
```

```
        //CHECK EVENT
        gp.eHandler.checkEvent();

        //IF COLLISION IS FALSE, PLAYER CAN MOVE
        if(collisionOn == false && keyH.enterPressed == false) {
            switch(direction) {
            case "up": worldY -= speed; break;
            case "down": worldY += speed; break;
            case "left": worldX -= speed; break;
            case "right": worldX += speed; break;
            }
        }

        if(keyH.enterPressed == true && attackCanceled == false) {
            gp.playSE(7);
            attacking = true;
            spriteCounter = 0;
        }

        attackCanceled = false;
        gp.keyH.enterPressed = false;

        spriteCounter++;
        if(spriteCounter > 10) {
            if(spriteNum == 1) {
                spriteNum = 2;
            }
            else if(spriteNum == 2) {
                spriteNum = 1;
            }
            spriteCounter = 0;
        }
    }
```

```
// This needs to be outside of key if statement!
if(invincible == true) {
    invincibleCounter++;
    if(invincibleCounter > 60) {
        invincible = false;
        invincibleCounter = 0;
    }
}
if(life <= 0) {
    gp.gameState = gp.gameOverState;
    gp.ui.commandNum = -1;
    gp.stopMusic();
    gp.playSE(10);
}
```

Update จะ update player ทั้งการเดิน เช็คการชน tile, object, npc และ monster

```
public void pickUpObject(int i) {

    if(i != 999) {
        //PICKUP ONLY ITEM
        if(gp.obj[i].type == type_pickupOnly) {
            gp.obj[i].use(this);
            gp.obj[i] = null;
        }
        //OBSTACLE
        else if(gp.obj[i].type == type_obstacle) {
            if(keyH.enterPressed == true) {
                attackCanceled = true;
                gp.obj[i].interact();
            }
        }
        //INVENTORY ITEM
        else {
            String text;

            if(inventory.size() != maxinventorySize) {
                inventory.add(gp.obj[i]);
                gp.playSE(2);
                text = "Got a " + gp.obj[i].name + "!";
            }
            else
            {
                text = "Your cannot carry any more!";
            }
            gp.ui.addMessage(text);
            gp.obj[i] = null;
        }
    }
}
```

q

pickUpObject ไว้เก็บ Object และแสดงลงใน กระเป๋า

```
public void interactNPC(int i) {

    if(gp.keyH.enterPressed == true) {
        if(i != 999) {
            attackCanceled = true;
            gp.gameState = gp.dialogueState;
            if(gp.npc[i] != gp.npc[1]) {
                gp.npc[i].speak();
            }
            if(gp.npc[i] == gp.npc[1]) {
                if(count < 6) {
                    gp.npc[i].speak();
                    count++;
                }
                else {
                    gp.gameState = gp.winState;
                    gp.stopMusic();
                    gp.playSE(4);
                }
            }
        }
    }


}
```

interactNPCเช็คเงื่อนไขการพูดกับ npc

```
public void contactMonster(int i) {

    if(i != 999) {
        if(invincible == false && gp.monster[i].dying == false) {

            gp.playSE(6);

            int damage = gp.monster[i].attack - defense;
            if(damage < 0) {
                damage = 0;
            }
            life -= damage;
            invincible = true;
        }
    }
}
public void damageMonster(int i) {

    if(i != 999) {
        if(gp.monster[i].invincible == false) {

            gp.playSE(5);

            int damage = attack - gp.monster[i].defense;
            if(damage < 0) {
                damage = 0;
            }
            gp.monster[i].life -= damage;
            gp.ui.addMessage(damage + " damage!");

            gp.monster[i].invincible = true;
            gp.monster[i].damageReaction();

            if(gp.monster[i].life <= 0) {
                gp.monster[i].dying = true;
                gp.ui.addMessage("killed the " + gp.monster[i].name + "!");
                gp.ui.addMessage("Coin + " + gp.monster[i].coin + "!");
                coin += gp.monster[i].coin;
```

contactMonster ถ้า monsterตายและในขณะแสดงการตายจะไม่โดนดาเมจ

damageMonster ทำดาเมจใส่monster

```
public void selectItem() {
    int itemIndex = gp.ui.getItemIndexOnSlot();

    if(itemIndex < inventory.size()) {

        Entity selectedItem = inventory.get(itemIndex);

        if(selectedItem.type == type_sword) {

            currentWeapon = selectedItem;
            attack = getAttack();
        }
        if(selectedItem.type == type_shield) {

            currentShield = selectedItem;
            defense = getDefense();
        }
        if(selectedItem.type == type_armor) {

            defense = getDefense();
        }
        if(selectedItem.type == type_consumable) {

            if(selectedItem.use(this) == true) {
                inventory.remove(itemIndex);
            }
        }
    }
}
```

selectItem เลือก item จากกระเป๋า

```
public void draw(Graphics2D g2) {

      BufferedImage image = null;
      int tempScreenX = screenX;
      int tempScreenY = screenY;

      switch(direction) {
          case "up":
              if(attacking == false) {
                  if(spriteNum == 1) {image = up1;}
                  if(spriteNum == 2) {image = up2;}
              }
              if(attacking == true) {
                  tempScreenY = screenY - gp.tileSize;
                  if(spriteNum == 1) {image = attackUp1;}
                  if(spriteNum == 2) {image = attackUp2;}
              }
              break;
          case "down":
              if(attacking == false) {
                  if(spriteNum == 1) {image = down1;}
                  if(spriteNum == 2) {image = down2;}
              }
              if(attacking == true) {
                  if(spriteNum == 1) {image = attackDown1;}
                  if(spriteNum == 2) {image = attackDown2;}
              }
              break;
          case "left":
              if(attacking == false) {
                  if(spriteNum == 1) {image = left1;}
                  if(spriteNum == 2) {image = left2;}
              }
              if(attacking == true) {
                  tempScreenX = screenX - gp.tileSize;
                  if(spriteNum == 1) {image = attackLeft1;}
```

```
            case "left":
                if(attacking == false) {
                    if(spriteNum == 1) {image = left1;}
                    if(spriteNum == 2) {image = left2;}
                }
                if(attacking == true) {
                    tempScreenX = screenX - gp.tileSize;
                    if(spriteNum == 1) {image = attackLeft1;}
                    if(spriteNum == 2) {image = attackLeft2;}
                }
                break;
            case "right":
                if(attacking == false) {
                    if(spriteNum == 1) {image = right1;}
                    if(spriteNum == 2) {image = right2;}
                }
                if(attacking == true) {
                    if(spriteNum == 1) {image = attackRight1;}
                    if(spriteNum == 2) {image = attackRight2;}
                }
                break;
        }

        if(invincible == true) {
            g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.4f));    //จางๆ
        }
        g2.drawImage(image, tempScreenX, tempScreenY, null);

        //Reset alpha
        g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 1f));

        //DEBUG
//          g2.setFont(new Font("Arial", Font.PLAIN, 26));
//          g2.setColor(Color.white);
//          g2.drawString("Invincible:"+invincibleCounter, 10, 400);
    }
}
```

Draw ว่าการเดิน และวาดช่วงเวลาที่เป็นอมตะ

```java
package main;

import entity.NPC_DemonLord;

public class AssetSetter {

    GamePanel gp;
    public AssetSetter(GamePanel gp) {
        this.gp = gp;
    }

    public void setObject() {

        int i = 0;

        gp.obj[i] = new OBJ_Key(gp);
        gp.obj[i].worldX = gp.tileSize*37;
        gp.obj[i].worldY = gp.tileSize*43;
        i++;
        gp.obj[i] = new OBJ_Key(gp);
        gp.obj[i].worldX = gp.tileSize*33;
        gp.obj[i].worldY = gp.tileSize*7;
        i++;
        gp.obj[i] = new OBJ_Key(gp);
        gp.obj[i].worldX = gp.tileSize*20;
        gp.obj[i].worldY = gp.tileSize*42;
        i++;
        gp.obj[i] = new OBJ_Shield_Iron(gp);
        gp.obj[i].worldX = gp.tileSize*33;
        gp.obj[i].worldY = gp.tileSize*21;
        i++;
        gp.obj[i] = new OBJ_Potion(gp);
        gp.obj[i].worldX = gp.tileSize*23;
        gp.obj[i].worldY = gp.tileSize*7;
        i++;
        gp.obj[i] = new OBJ_Potion(gp);
        gp.obj[i].worldX = gp.tileSize*14;
        gp.obj[i].worldY = gp.tileSize*25;
```

Class AssetSetter ไว้ set obj, npc, monster ใน map

```java
package main;

import entity.Entity;

public class CollisionChecker {

    GamePanel gp;

    public CollisionChecker(GamePanel gp) {
        this.gp = gp;
    }

    public void checkTile(Entity entity) {

        int entityLeftWorldX = entity.worldX + entity.solidArea.x;
        int entityRightWorldX = entity.worldX + entity.solidArea.x + entity.solidArea.width;
        int entityTopWorldY = entity.worldY + entity.solidArea.y;
        int entityBottomWorldY = entity.worldY + entity.solidArea.y + entity.solidArea.height;

        int entityLeftCol = entityLeftWorldX / gp.tileSize;
        int entityRightCol = entityRightWorldX / gp.tileSize;
        int entityTopRow = entityTopWorldY / gp.tileSize;
        int entityBottomRow = entityBottomWorldY / gp.tileSize;

        int tileNum1, tileNum2;

        switch(entity.direction) {
        case "up":
            entityTopRow = (entityTopWorldY - entity.speed) / gp.tileSize;
            tileNum1 = gp.tileM.mapTileNum[entityLeftCol][entityTopRow];
            tileNum2 = gp.tileM.mapTileNum[entityRightCol][entityTopRow];
            if(gp.tileM.tile[tileNum1].collosion == true || gp.tileM.tile[tileNum2].collosion == true) {
                entity.collisionOn = true;
            }
            break;
        case "down":
            entityBottomRow = (entityBottomWorldY + entity.speed) / gp.tileSize;
            tileNum1 = gp.tileM.mapTileNum[entityLeftCol][entityBottomRow];
```

```java
    case "down":
        entityBottomRow = (entityBottomWorldY + entity.speed) / gp.tileSize;
        tileNum1 = gp.tileM.mapTileNum[entityLeftCol][entityBottomRow];
        tileNum2 = gp.tileM.mapTileNum[entityRightCol][entityBottomRow];
        if(gp.tileM.tile[tileNum1].collosion == true || gp.tileM.tile[tileNum2].collosion == true) {
            entity.collisionOn = true;
        }
        break;
    case "left":
        entityLeftCol = (entityLeftWorldX - entity.speed) / gp.tileSize;
        tileNum1 = gp.tileM.mapTileNum[entityLeftCol][entityTopRow];
        tileNum2 = gp.tileM.mapTileNum[entityLeftCol][entityBottomRow];
        if(gp.tileM.tile[tileNum1].collosion == true || gp.tileM.tile[tileNum2].collosion == true) {
            entity.collisionOn = true;
        }
        break;
    case "right":
        entityRightCol = (entityRightWorldX + entity.speed) / gp.tileSize;
        tileNum1 = gp.tileM.mapTileNum[entityRightCol][entityTopRow];
        tileNum2 = gp.tileM.mapTileNum[entityRightCol][entityBottomRow];
        if(gp.tileM.tile[tileNum1].collosion == true || gp.tileM.tile[tileNum2].collosion == true) {
            entity.collisionOn = true;
        }
        break;
    }
```

checkTile เช็คการชน ของ player

```java
public int checkObject(Entity entity, boolean player) {

    int index = 999;

    for(int i = 0 ; i< gp.obj.length ; i++) {

        if(gp.obj[i] != null) {

            //Get entity's solid area position
            entity.solidArea.x = entity.worldX + entity.solidArea.x;
            entity.solidArea.y = entity.worldY + entity.solidArea.y;
            //Get the object's solid area position
            gp.obj[i].solidArea.x = gp.obj[i].worldX + gp.obj[i].solidArea.x;
            gp.obj[i].solidArea.y = gp.obj[i].worldY + gp.obj[i].solidArea.y;

            switch(entity.direction) {
            case "up":
                entity.solidArea.y -= entity.speed; break;
            case "down":
                entity.solidArea.y += entity.speed; break;
            case "left":
                entity.solidArea.x -= entity.speed; break;
            case "right":
                entity.solidArea.x += entity.speed; break;
            }

            if(entity.solidArea.intersects(gp.obj[i].solidArea)) {
                if(gp.obj[i].collision == true) {
                    entity.collisionOn = true;
                }
                if(player == true) {
                    index = i;
                }
            }

            entity.solidArea.x = entity.solidAreaDefaultX;
            entity.solidArea.y = entity.solidAreaDefaultY;
```

```java
                entity.solidArea.y = entity.solidAreaDefaultY;
                gp.obj[i].solidArea.x = gp.obj[i].solidAreaDefaultX;
                gp.obj[i].solidArea.y = gp.obj[i].solidAreaDefaultY;
            }
        }

        return index;
    }
```

checkObject ไว้เช็ค object

```java
public boolean checkPlayer(Entity entity) {

    boolean contactPlayer = false;
    //Get entity's solid area position
    entity.solidArea.x = entity.worldX + entity.solidArea.x;
    entity.solidArea.y = entity.worldY + entity.solidArea.y;
    //Get the object's solid area position
    gp.player.solidArea.x = gp.player.worldX + gp.player.solidArea.x;
    gp.player.solidArea.y = gp.player.worldY + gp.player.solidArea.y;

    switch(entity.direction) {
    case "up":
        entity.solidArea.y -= entity.speed; break;
    case "down":
        entity.solidArea.y += entity.speed; break;
    case "left":
        entity.solidArea.x -= entity.speed; break;
    case "right":
        entity.solidArea.x += entity.speed; break;
    }

    if(entity.solidArea.intersects(gp.player.solidArea)) {
        entity.collisionOn = true;
        contactPlayer = true;
    }

    entity.solidArea.x = entity.solidAreaDefaultX;
    entity.solidArea.y = entity.solidAreaDefaultY;
    gp.player.solidArea.x = gp.player.solidAreaDefaultX;
    gp.player.solidArea.y = gp.player.solidAreaDefaultY;

    return contactPlayer;
}
```

checkPlayer ไว้เช็ค player

```java
package main;

import java.io.BufferedReader;

public class Config {

    GamePanel gp;

    public Config(GamePanel gp) {

        this.gp = gp;
    }

    public void saveConfig() {
        try {

            BufferedWriter bw = new BufferedWriter(new FileWriter("config.txt"));

            //Full screen
            if(gp.fullScreenOn == true) {
                bw.write("On");
            }
            if(gp.fullScreenOn == false) {
                bw.write("Off");
            }
            bw.newLine();

            //Music volume
            bw.write(String.valueOf(gp.music.volumeScale));
            bw.newLine();

            //SE
            bw.write(String.valueOf(gp.se.volumeScale));
            bw.newLine();

            bw.close();

        } catch (IOException e) {
```

saveConfig save การตั้งค่า

loadConfig load การตั้งค่า

```java
package main;

import entity.Entity;

public class GamePanel extends JPanel implements Runnable{
    //SCREEN SETTING
    final int originalTileSize = 16; // 16*16 tile
    final int scale = 3;

    public final int tileSize = originalTileSize*scale; // 48*48 tile
    public final int maxScreenCol = 20;
    public final int maxScreenRow = 12;
    public final int screenWidth = tileSize*maxScreenCol;  // 960 pixels
    public final int screenHeight = tileSize*maxScreenRow; // 576 pixels
    public boolean fullScreenOn = false;


    //WORLD SETTING
    public final int maxWorldCol = 50;
    public final int maxWorldRow = 50;
    public final int maxMap = 10;
    public int currentMap = 0;

    //FOR FULL SCREEN
    int screenWidth2 = screenWidth;
    int screenHeight2 = screenHeight;
    BufferedImage tempScreen;
    Graphics2D g2;

    //FPS
    int FPS = 60;

    //SYSTEM
    public TileManager tileM = new TileManager(this);
    public KeyHandler keyH = new KeyHandler(this);
    Sound music = new Sound();
    Sound se = new Sound();
    public CollisionChecker cChecker = new CollisionChecker(this);
```

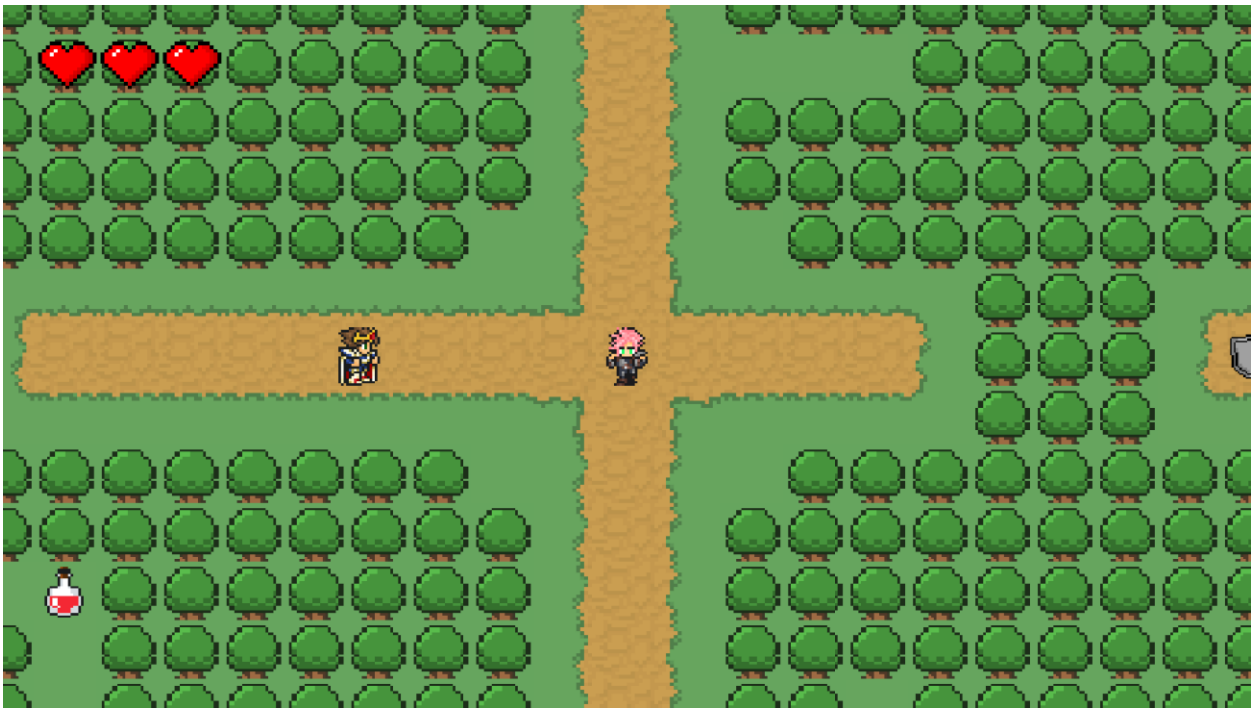
Class GamePanel มีการ extends JPanel และการ implements Runnable และเรียกใช้ 0kd class อื่นๆมากมาย

```java
package main;

import java.awt.event.KeyEvent;

public class KeyHandler implements KeyListener {

    GamePanel gp;
    public boolean upPressed, downPressed, leftPressed, rightPressed, enterPresse
    //DEBUG
    boolean checkDrawTime = false;

    public KeyHandler(GamePanel gp) {
        this.gp = gp;
    }

    @Override
    public void keyTyped(KeyEvent e) {
    }

    @Override
    public void keyPressed(KeyEvent e) {
        int code = e.getKeyCode();
        //TITLE STATE
        if(gp.gameState == gp.titleState) {
            titleState(code);
        }
        //PLAY STATE
        else if(gp.gameState == gp.playState) {
            playState(code);
        }

        //PAUSE STATE
        else if(gp.gameState == gp.pauseState) {
            pauseState(code);
        }
```

Class KeyHandler รับค่าจากการกดปุ่มต่างๆ มีการ implements KeyListenner

หน้า GUI ตอนเริ่มเกมส์

หน้าเปิดเกม



เมื่อแพ้

เมื่อชนะ

Components ที่มี

ในโปรแกรมที่ทำจะประกอบไปด้วย JFrame Jpanel และ Keylistener และ JPanel เป็น compostion กับ JFrame เนื่องจากถ้าไม่มี JFrame จะไม่สามารถใช้งาน JPanel ได้

บทที่ 3

สรุป

ปัญหาระหว่างการพัฒนา

ในการพัฒนา แต่ละส่วนนั้นยากมาก เพราะว่าเนื่องมี Animation มาเกี่ยวข้องทำให้การปรับภาพให้ความสมดุลและเสถียรเป็นไปได้ด้วยจึงทำต้องทำการหาข้อมูลเพิ่มเติมและมีปัญหาอยู่หลาย ครั้งเช่น เก็บไอเทมแต่ไอเทมไม่หายไป การปรับค่าสถานะของผู้เล่นและมอนสเตอร์ให้มีความสมดุลไม่ให้ยากและง่ายเกินไป การกำหนด FPS ให้มีความไหลลื่นนั้น ต้องใช้เทรดและหาข้อมูลเพิ่มเติม ซึ่งถ้าเอาหลักๆแล้ว สามารถแบ่งเป็นหัวข้อได้ดังนี้

1. Animation ไม่มีภาพ หรือ ขาดๆเกินทำให้ไม่สวยงาม และขาดความเสถียร
2. การกำหนดทิศทางการเดินของnpcไม่ให้มีความซ้ำที่กัน ต้องใช้ random เข้าช่วยเพื่อแก้ปัญหา
3. การเดินของมอนสเตอร์ เข้าหาผู้เล่น
4. Area Damage ต้องสร้างกรอบเวลาชนแล้วคิดความเสียหายทำได้ยากมาก
5. เนื่องจากอนิเมชั่นกำหนดค่อนข้างยากทำให้มีข้อระยะเวลาจำกัดในการทำ จึงทำได้แค่บางส่วนเท่านั้น
6. ในระหว่างผิดแผนที่วางไว้มาก จึงต้องทำการตัดรายละเอียดบางส่วนของเกมส์ออก

จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

โปรแกรมที่ไม่เหมือนใคร คือ ระบบของเกมส์ที่มีการคุยกับ npc แสดงหลอดเลือดของมอนสเตอร์

และการแสดง map

คำแนะนำสำหรับผู้สอน

เนื้อหาโอเคแล้วค่ะ แต่ถ้ารู้สึกว่าสอนเร็วไปหน่อย อาจจะเพราะเวลามีน้อย เลยไม่สามารถลงแบบละเอียดได้ ระยะเวลาการทำโปรเจคน้อยไปค่ะ ถ้ารู้รายละเอียดของเกมส์ตั้งแต่ก่อน midterm จะสามารถเริ่มทำบางส่วนไว้ได้เลย เช่น วาดตัวละครและ animation คิดคอนเซ็ปต์เกมได้ล่วงหน้าเป็นต้นค่ะ