



รายงาน

เรื่อง : The Epic of Linkle

เสนอ

ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์

วิชาการเขียนโปรแกรมเชิงวัตถุ (OOP)

ภาคเรียนที่ 1/2566

โดย

นาย วชิรสรณ์ เต็มรัตนสุวรรณ รหัส 6504062630251

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

คำนำ

รายงานเล่มนี้จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของวิชาการเขียนโปรแกรมเชิงวัตถุ (OOP) คณะวิทยาศาสตร์ประยุกต์ ภาควิชาวิทยาการคอมพิวเตอร์ ปีที่ 2 เพื่อให้ได้ศึกษาหาความรู้ในเรื่อง การเขียนโปรแกรมเชิงวัตถุ และได้ศึกษาอย่างเข้าใจเพื่อเป็นประโยชน์กับการเรียน

ผู้จัดทำหวังว่า รายงานเล่มนี้จะเป็นประโยชน์กับผู้อ่าน หรือนักเรียน นักศึกษาที่กำลังหาข้อมูลเรื่องนี้อยู่ หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้และขออภัยมา ณ ที่นี้ด้วย

วชิรสรณ์ เต็มรัตนสุวรรณ

7 พฤศจิกายน 2566

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโปรเจกต์

การพัฒนาเกมไม่เพียงแต่เป็นกระบวนการทางเทคนิค แต่ยังเป็นศิลปะที่ต้องการความคิดสร้างสรรค์และความใส่ใจในรายละเอียดทุกรายละเอียด โปรเจกต์ "The Epic of Linkle" นี้ไม่เพียงแต่เป็นเกม แต่เป็นสื่อที่เล่าเรื่องราวในโลกแห่งแฟนตาซี ผ่านการเขียนโปรแกรมที่ใส่ใจกับโครงสร้างและการออกแบบที่ดี ซึ่งเป็นผลลัพธ์ของความพยายามและความมุ่งมั่นของนักศึกษาในการเรียนรู้และปฏิบัติทักษะในวิชาการเขียนโปรแกรมเชิงวัตถุ (OOP)

1.2 ประเภทของโครงการ

Java Applications

1.3 ประโยชน์ของโครงการ

1. ช่วยฝึกการสร้างโครงสร้างของโปรแกรมที่มีความเป็นระเบียบ
2. ช่วยฝึกการลดโค้ดที่ต้องเขียนซ้ำซากโดยใช้การ Inheritance และ Encapsulation
3. ช่วยฝึกการจัดโครงสร้างของเกมเป็นระเบียบและเข้าใจง่าย

1.4 ขอบเขตของโครงการ

1.4.1 การวางแผน (Planning)

กำหนดวัตถุประสงค์ของโครงการ โดยกำหนดว่าต้องการสร้างเกมประเภทใด และมีฟีเจอร์หรือระบบใดบ้างที่ต้องการให้มีในเกม ออกแบบคลาสและวัตถุเพื่อกำหนดคลาสต่างๆ และหน้าที่ของแต่ละคลาส เช่น ผู้เล่น (Player), ศัตรู (Enemy), ไอเท็ม (Item) เป็นต้น

1.4.2 การเขียนโปรแกรม (Programming)

สร้างคลาสและวัตถุตามที่วางแผนไว้ โดยการเขียนโค้ดสร้างคลาสของแต่ละวัตถุ และใส่เมธอด (methods) ที่ใช้ในการจัดการข้อมูลและกระทำต่างๆ

1.4.3 การทดสอบและปรับปรุง (Testing and Refining)

ทดสอบโปรแกรม โดยทดสอบการทำงานของโปรแกรมเพื่อตรวจสอบว่าทุกฟีเจอร์ทำงานตามที่ต้องการหรือไม่ และมีการปรับปรุงโค้ดหรือเพิ่มฟีเจอร์ตามความต้องการและข้อเสนอแนะ

1.4.4 เขียนรายงาน (Documentation)

การเขียนรายงานโครงการ โดยภายในเล่มรายงานรวบรวมข้อมูลเกี่ยวกับโครงการ เช่น วัตถุประสงค์, วิธีการทำ, ปัญหาที่พบและวิธีแก้ไข, และผลลัพธ์ที่ได้ รวมทั้งแสดงตัวอย่างโค้ดที่เกี่ยวข้องกับการใช้ OOP ในโปรเจกต์

1.4.5 ตารางแผนการทำงานเดือนกันยายน - พฤศจิกายน

1. ตารางแผนการทำงานเดือนกันยายน

ลำดับ	รายการ	16 – 22 ก.ย.	23 – 30 ก.ย.
1	วางแผนโปรเจ็ค • วางแผนโปรเจ็ควางแผนโปรเจ็ค • ออกแบบโครงสร้างของเกมและคลาสต่างๆ		
2	เขียนโค้ด • เริ่มเขียนโค้ดในภาษา Java • สร้างคลาสต่างๆ และเมธอดพื้นฐานที่จำเป็น		

2. ตารางแผนการทำงานเดือนตุลาคม

ลำดับ	รายการ	01 – 25 ต.ค.	26 – 31 ต.ค.
3	พัฒนาความสามารถของเกม • เพิ่มฟีเจอร์และระบบต่างๆ เช่น การเคลื่อนที่ • ทำการทดสอบความถูกต้องของระบบที่เพิ่มเข้าไป		
4	ทดสอบและปรับปรุง • ทดสอบโปรแกรมโดยให้ผู้ทดสอบใช้งานเกม • รับข้อเสนอแนะและแก้ไขปัญหาที่พบ		

3. ตารางแผนการทำงานเดือนพฤศจิกายน

ลำดับ	รายการ	01 – 09 พ.ย.
5	เขียนรายงานและจัดรูปแบบ • เขียนรายงานโครงการโดยรวบรวมข้อมูลจากการทดลองเล่นเกม • จัดรูปแบบรายงานและตรวจสอบความถูกต้องของข้อมูลและการเขียน	

บทที่ 2

ส่วนการพัฒนา

2.1 เนื้อเรื่องย่อและวิธีการเล่น

2.1.1 เนื้อเรื่อง

ในโลกที่อยู่ในอนาคตที่ห่างไกลจากโลกปกติที่เรารู้จัก มีเกาะร้างหนึ่งซึ่งอยู่ตรงกลางของมหาสมุทรและเต็มไปด้วยสัตว์ประหลาดและธรรมชาติที่ร้ายแรง คุณตื่นขึ้นบนเกาะนี้โดยไม่มีความทรงจำว่าทำไมคุณถึงมาอยู่ที่นี่ แต่คุณสัมผัสได้ว่ามีบางสิ่งที่อยู่ใต้ดินเจี้ยนกำลังเรียกหาคุณอยู่

2.1.2 วิธีการเล่น

ลักษณะการทำงาน	ปุ่ม	ลักษณะการทำงาน	ปุ่ม
เคลื่อนที่ไปข้างหน้า	W	เปิดหน้าต่างเมนู	ESC
เคลื่อนที่ไปข้างหลัง	S	เปิดหน้าต่างตัวละคร	C
เคลื่อนที่ไปทางซ้าย	A	เปิดมินิแมพ	X
เคลื่อนที่ไปทางขวา	D	เปิดแมพขนาดใหญ่	M
ทำการโจมตี	ENTER	ทำการใช้พลังเวท	F
หยุดเกมชั่วคราว	P	ทำการเลือกสิ่งของ	ENTER

2.2 Class Diagram

Entity
+Entity(GamePanel)
gp : GamePanel
+up1 : BufferedImage
+up2 : BufferedImage
+down1 : BufferedImage
+down2 : 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
+guardUp : BufferedImage
+guardDown : BufferedImage
+guardLeft : BufferedImage
+guardRight : BufferedImage
+image : BufferedImage
+image2 : BufferedImage
+image3 : BufferedImage
+solidArea : Rectangle
+attackArea : Rectangle
+solidAreaDefaultX : int
+solidAreaDefaultY : int
+collisionOn : boolean
+dialogues : String[][]
+attacker : Entity
+temp : boolean
+worldX : int
+worldY : int
+direction : String

Entity (continue)
+spriteNum : int
+dialogueSet : int
+dialogueIndex : int
+collision : boolean
+invincible : boolean
+attacking : boolean
+alive : boolean
+dying : boolean
+hpBarOn : boolean
+onPath : boolean
+knockBack : boolean
+knockBackDirection : String
+guarding : boolean
+transparent : boolean
+offBalance : boolean
+loot : Entity
+inRage : boolean
+sleep : boolean
+drawing : boolean
+spriteCounter : int
+actionLockCounter : int
+invincibleCounter : int
+shotAvailableCounter : int
+dyingCounter : int
+hpBarCounter : int
+knockBackCounter : int
+guardCounter : int
+offBalanceCounter : int
+name : String
+defaultSpeed : int
+speed : int
+maxLife : int
+life : int
+maxMana : int
+mana : int
+ammo : int

Entity (continue2)
+level : int
+strength : int
+dexterity : int
+attack : int
+defense : int
+exp : int
+nextLevelExp : int
+coin : int
+motion1_duration : int
+motion2_duration : int
+currentWeapon : Entity
+currentShield : Entity
+currentLight : Entity
+projectile : Projectile
+boss : boolean
+inventory : ArrayList<Entity>
+maxInventorySize : int
+value : int
+attackValue : int
+defenseValue : int
+description : String
+useCost : int
+price : int
+knockBackPower : int
+stackable : boolean
+amount : int
+lightRadius : int
+durability : int
+type : int
+type_player : int
+type_npc : int
+type_monster : int
+type_sword : int
+type_axe : int
+type_shield : int
+type_consumable : int

Entity (continue3)
+type_pickUpOnly : int
+type_obstacle : int
+type_light : int
+type_pickaxe : int
+getScreenX() : int
+getScreenY() : int
+getLeftX() : int
+getRightX() : int
+getTopY() : int
+getBottomY() : int
+getCol() : int
+getRow() : int
+getCenterX() : int
+getXdistance(Entity) : int
+getYdistance(Entity) : int
+getTileDistance(Entity) : int
+getGoalCol(Entity) : int
+getGoalRow(Entity) : int
+resetCounter() : void
+setLoot(Entity) : void
+setAction() : void
+damageReaction() : void
+speak() : void
+facePlayer() : void
+startDialogue(Entity, int) : void
+interact() : void
+use(Entity) : boolean
+checkdrop() : void
+dropItem(Entity) : void
+getParticleColor() : Color
+getParticleSize() : int
+getParticleSpeed() : int
+getParticleMaxLife() : int
+generateParticle(Entity, Entity) : void
+checkCollision() : void
+update() : void

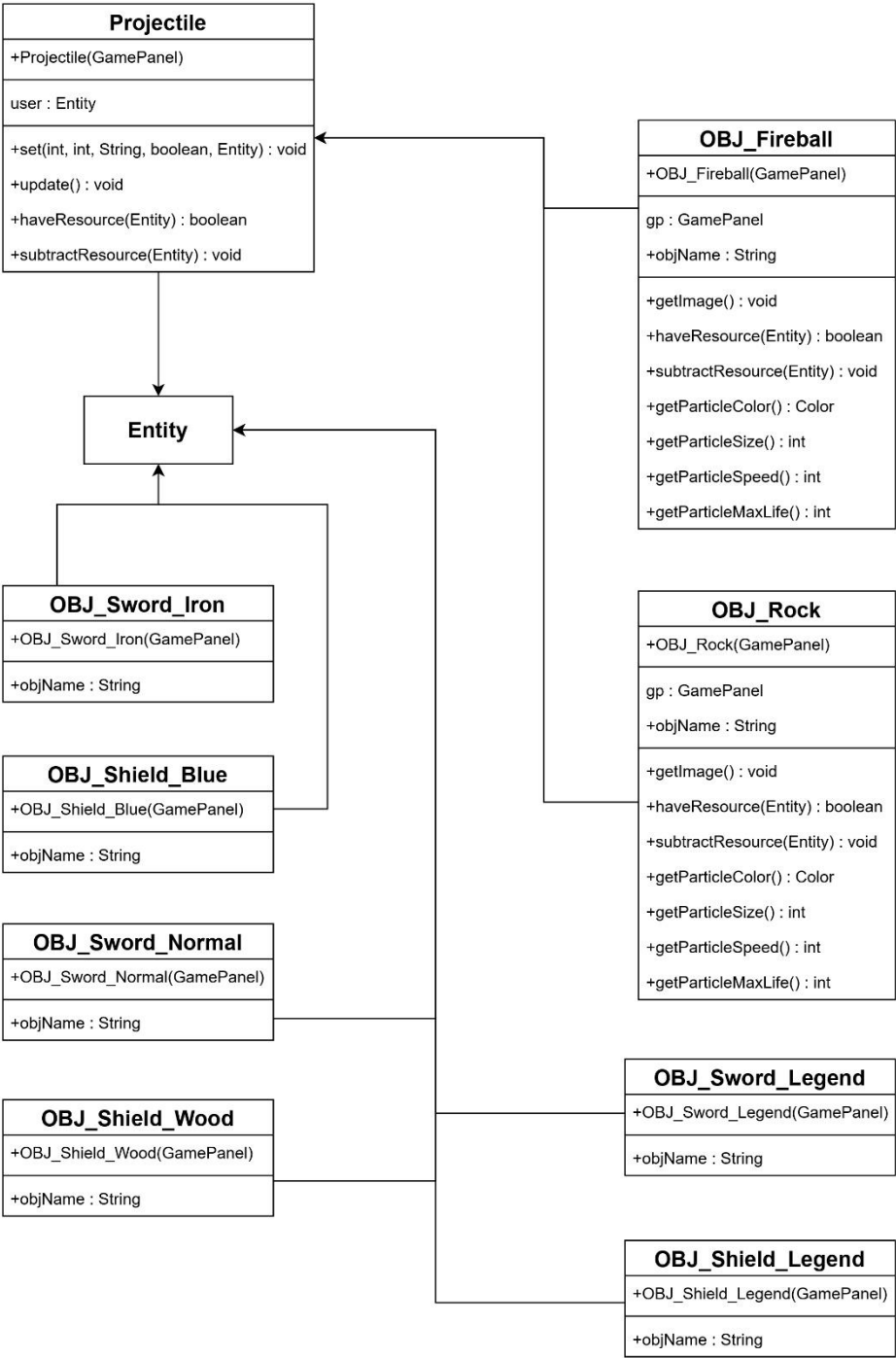
Entity (continue4)
+checkShootOrNot(int, int) : void
+checkAttackOrNot(int, int, int) : void
+checkStartChasingOrNot(Entity, int, int) : void
+checkStopChasingOrNot(Entity, int, int) : void
+getRandomDirection(int) : void
+moveTowardPlayer(int) : void
+getOppositeDirection(String) : String
+attacking() : void
+damagePlayer(int) : void
+setKnockBack(Entity, Entity, int) : void
+inCamera() : boolean
+draw(Graphics2D) : void
+dyingAnimation(Graphics2D) : void
+ChangeAlpha(Graphics2D, float) : void
+setup(String, int, int) : BufferedImage
+searchPath(int, int) : void
+getDetected(Entity, Entity[], String) : int

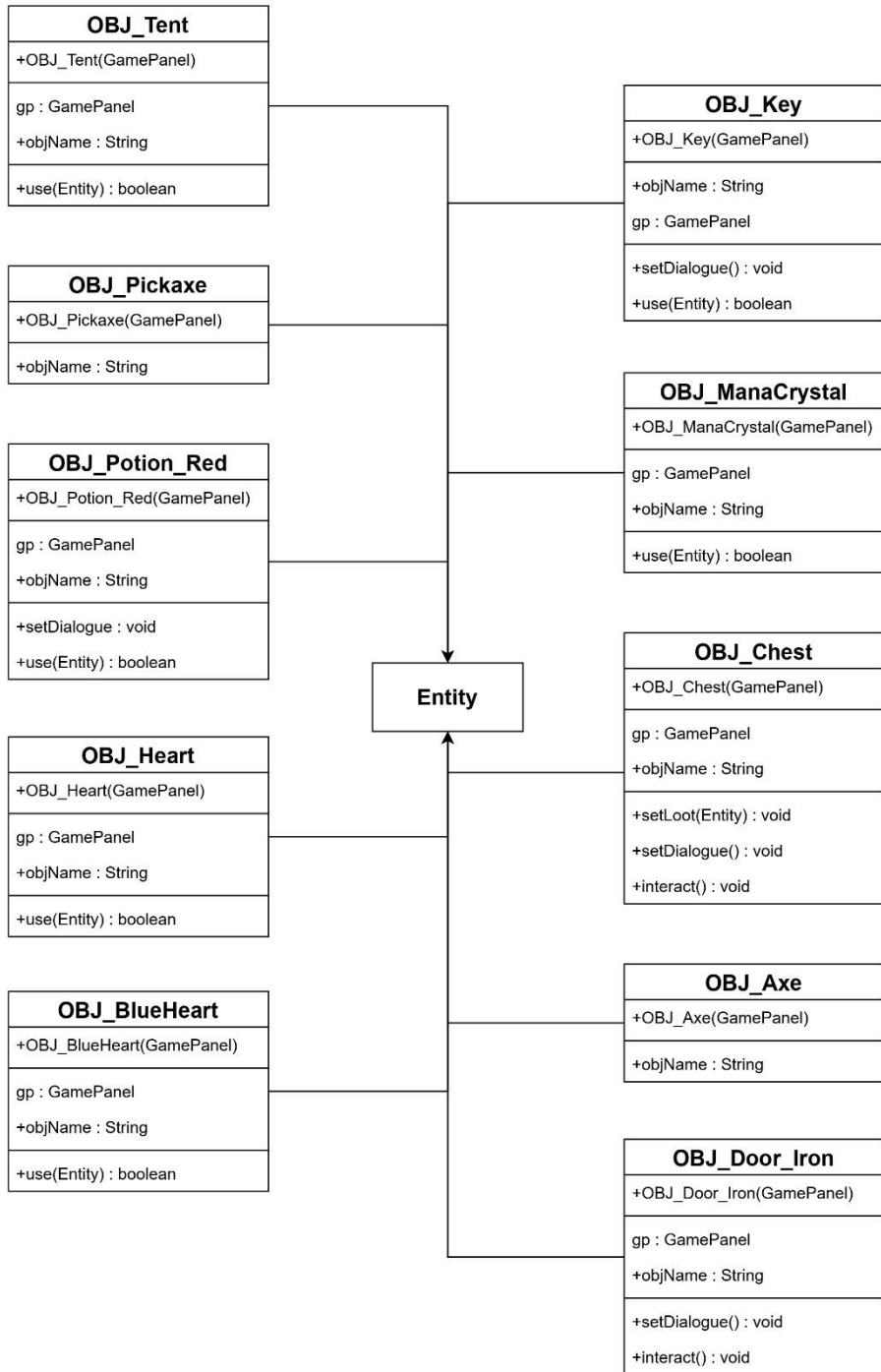
CutsceneManager
+CutsceneManager(GamePanel)
gp : GamePanel
g2 : Graphics2D
+sceneNum : int
+scenePhase : int
+NA : int
+skeletonLord : int
+ending : int
counter : int
alpha : float
y : int
endCredit : String
+draw(Graphics2D) : void
+sence_skeletonLord() : void
+sence_ending() : void
+counterReached(int) : boolean
+drawBlackBackground(float) : void
+drawString(float, float, int, String, int) : void

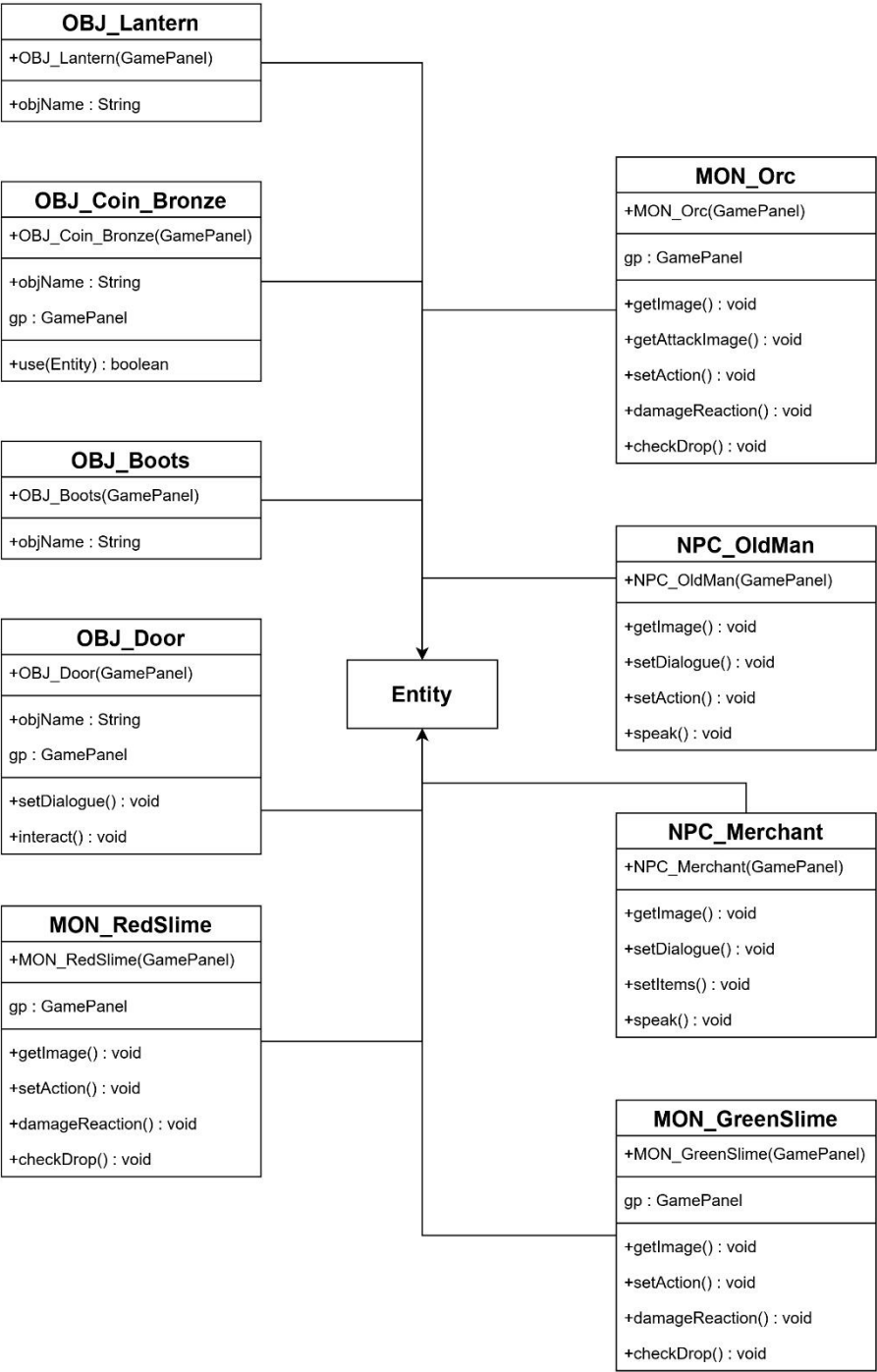
Lighting
+Lighting(GamePanel)
gp : GamePanel
+darknessFilter : BufferedImage
+dayCounter : int
+filterAlpha : float
+day : int
+dusk : int
+night : int
+dawn : int
+dayState : int
+setLightSource() : void
+resetDay() : void
+update() : void
+draw(Graphics2D) : void

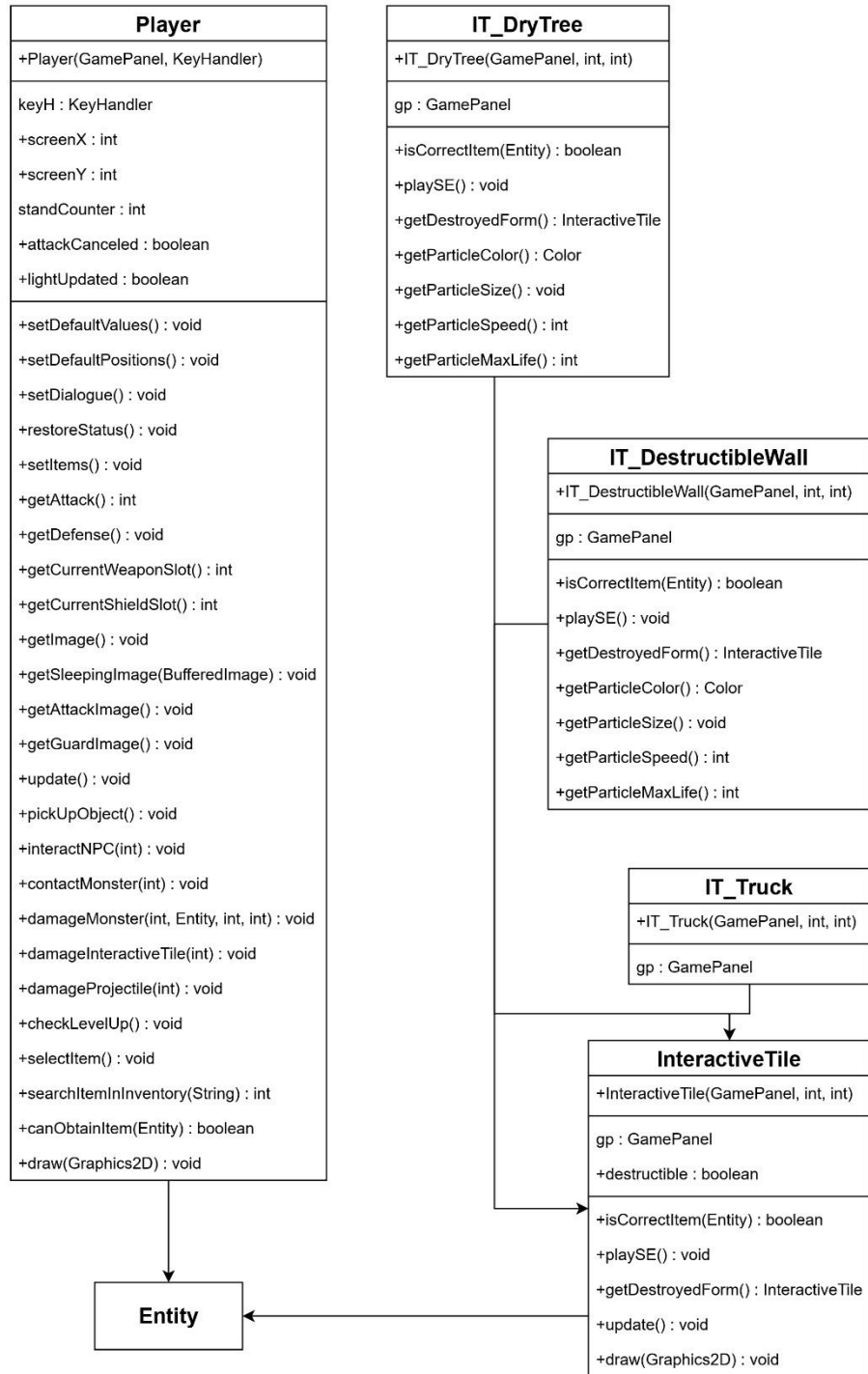
CollisionChecker
+CollisionChecker(GamePanel)
gp : GamePanel
+checkTile(Entity) : void
+checkObject(Entity, boolean) : int
+checkEntity(Entity, Entity[]) : int
+checkPlayer(Entity) : boolean

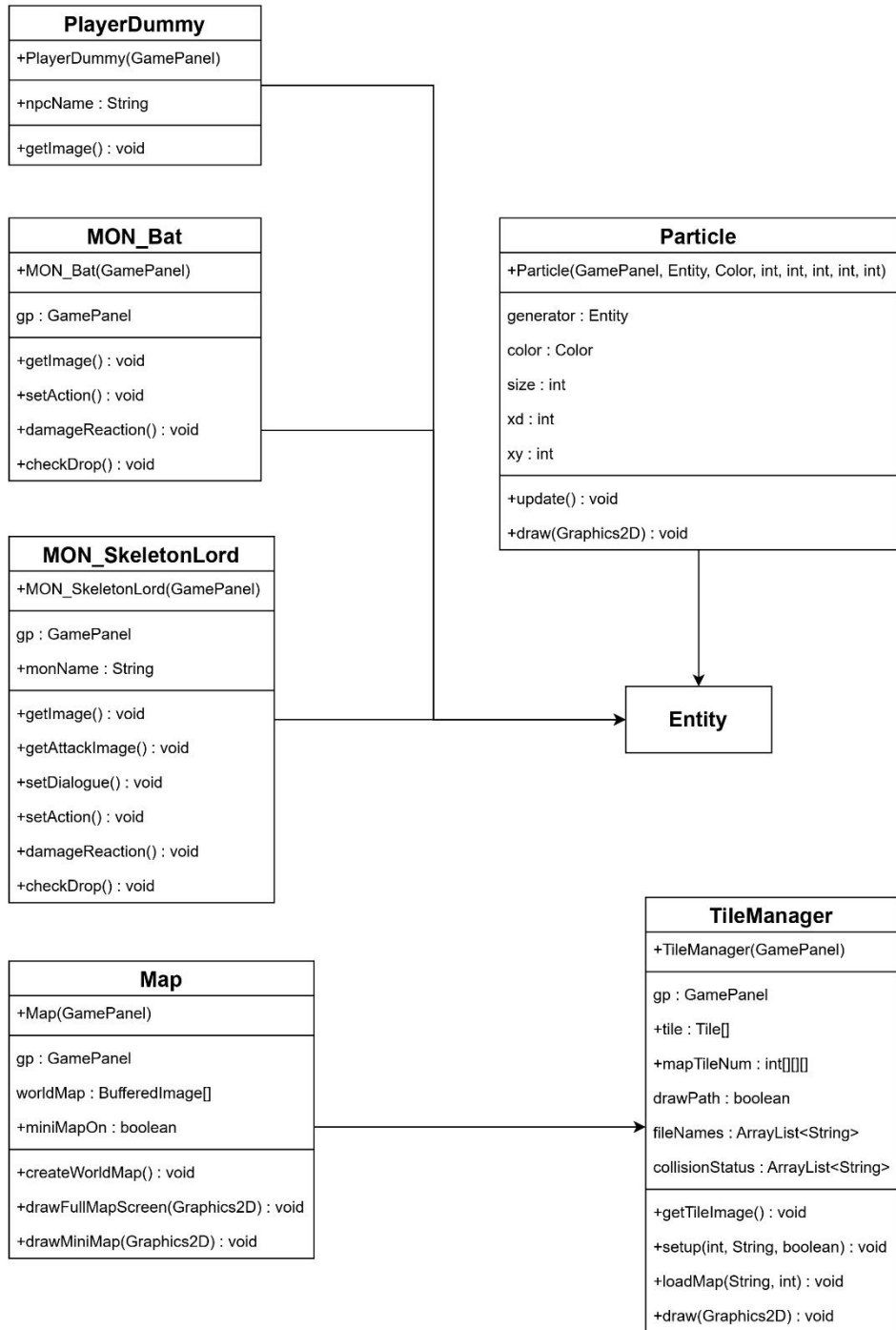
EnvironmentManager
+EnvironmentManager(GamePanel)
gp : GamePanel
+lighting : Lighting
+setup() : void
+update() : void
+draw(Graphics2D) : void











DataStorage
+DataStorage()
level : int maxLife : int life : int maxMana : int mana : int strength : int dexterity : int exp : int nextLevelExp : int coin : int itemNames : ArrayList<String> itemAmounts : ArrayList<Integer> currentWeaponSlot : int currentShieldSlot : int mapObjectName : String[] mapObjectWorldX : int[] mapObjectWorldY : int[] mapObjectLootNames : String[] mapObjectOpened : boolean[]

Config
+Config(GamePanel)
gp : GamePanel
+saveConfig() : void
+loadConfig() : void

SaveLoad
+SaveLoad(GamePanel)
gp : GamePanel
+save() : void
+load() : void

Tile
+Tile()
+collision : boolean
+image : BufferedImage

Progress
+Progress()
+skeletonLordDefeated : boolean

EventRect
+EventRect(GamePanel)
+eventDone : boolean
+eventRectDefaultX : int
+eventRectDefaultY : int

EntityGenerator
+EntityGenerator(GamePanel)
gp : GamePanel
+getObject(String) : Entity

Sound
+Sound()
clip : Clip soundURL : URL[] fc : FloatControl volumeScale : int volume : float +setFile(int) : void +play() : void +loop() : void +stop() : void +checkVolume() : void

AssetSetter
+AssetSetter(GamePanel)
gp : GamePanel
+setObject() : void
+setNPC() : void
+getMonster() : void
+setInteractiveTile() : void

Main
+Main()
window : JFrame
+main(String[]) : void
+setIcon() : void

Node
+Node(int, int)
parent : Node
+col : int
+row : int
gCost : int
hCost : int
fCost : int
solid : boolean
open : boolean
checked : boolean

PathFinder
+PathFinder(GamePanel)
gp : GamePanel node : Node[][] openList : ArrayList<Node> +pathList : ArrayList<Node> startNode : Node goalNode : Node currentNode : Node goalReached : boolean step : int
+instantiateNodes() : void +resetNodes() : void +setNodes(int, int, int, Entity) : void +getCost(Node) : void +search() : boolean +openNode(Node) : void +trackThePath() : void

EventHandler
+EventHandler(GamePanel)
gp : GamePanel eventRect : EventRect[][] eventMaster : Entity previousEventX : int previousEventY : int canTouchEvent : boolean tempMap : int tempCol : int tempRow : int
+setDialogue() : void +checkEvent() : void +hit(int, int, int, String) : boolean +damagePit(int) : void +healingPool(int) : void +teleport(int, int, int, int) : void +speack(Entity) : void +skeletonLord() : void

KeyHandler
+KeyHandler(GamePanel)
gp : GamePanel upPressed : boolean downPressed : boolean leftPressed : boolean rightPressed : boolean enterPressed : boolean shotKeyPressed : boolean spacePressed : boolean checkDrawTime : boolean godModeOn : boolean
+keyTyped(KeyEvent) : void +keyPressed(KeyEvent) : void +titleState(int) : void +playState(int) : void +pauseState(int) : void +dialogueState(int) : void +characterState(int) : void +optionsState(int) : void +gameOverState(int) : void +tradeState(int) : void +mapState(int) : void +playerInventory(int) : void +npcInventory(int) : void +keyReleased(KeyEvent) : void

UtilityTool
+UtilityTool()
+scaleImage(BufferedImage, int, int) : BufferedImage

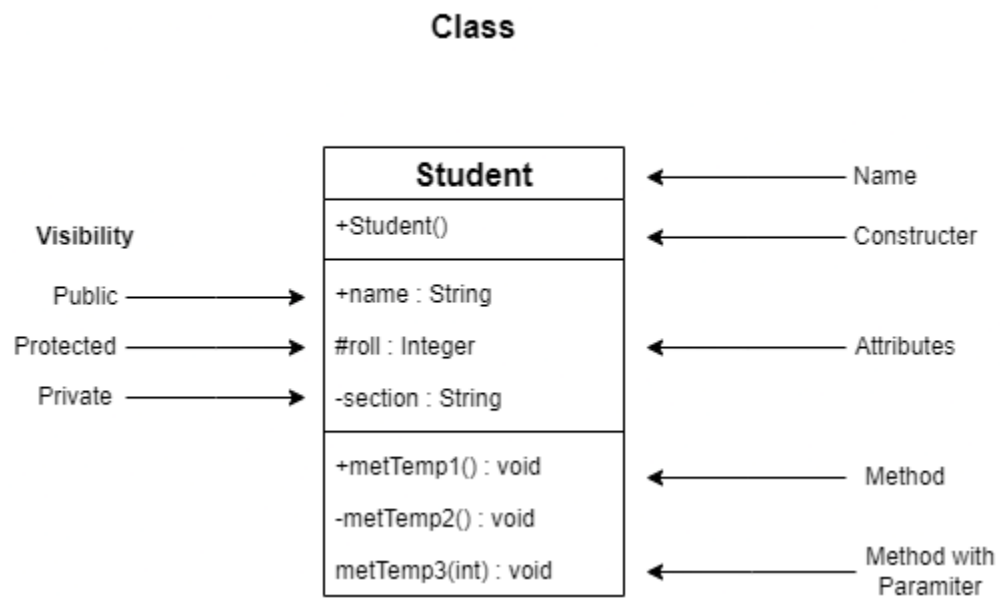
UI
+UI(GamePanel)
gp : GamePanel
g2 : Graphics2D
+maruMonica : Font
+purisaB : Font
heart_full : BufferedImage
heart_half : BufferedImage
heart_blank : BufferedImage
crystal_full : BufferedImage
crystal_blank : BufferedImage
coin : BufferedImage
+messageOn : boolean
message : ArrayList<String>
messageCounter : ArrayList<Integer>
+gameFinished : boolean
+currentDialogue : String
+commandNum : int
+titleScreenState : int
+playerSlotCol : int
+playerSlotRow : int
+npcSlotCol : int
+npcSlotRow : int
subState : int
counter : int
+npc : Entity
charIndex : int
combinedText : String

UI (continue)
+addMessage(String) : void
+draw(Graphics2D) : void
+drawPlayerLife() : void
+drawMonsterLife() : void
+drawMessage() : void
+drawTitleScreen() : void
+drawPauseScreen() : void
+drawDialogueScreen() : void
+drawCharacterScreen(Entity, boolean) : void
+drawGameOverScreen() : void
+drawOptionsScreen() : void
+options_top(int, int) : void
+options_fullScreenNotification(int, int) : void
+options_control(int, int) : void
+options_endGameConfirmation(int, int) : void
+drawTransition() : void
+drawTradeScreen() : void
+trade_select() : void
+trade_buy() : void
+trade_sell() : void
+drawSleepScreen() : void
+getItemIndexOnSlot(int, int) : int
+drawSubWindow(int, int, int, int) : void
+getXforCenteredText(String) : int
+getXforAlignToRightText(String, int) : int

GamePanel
+GamePanel()
+originalTileSize : int
+scale : int
+tileSize : int
+maxScreenCol : int
+maxScreenRow : int
+screenWidth : int
+screenHeight : int
+maxWorldCol : int
+maxWorldRow : int
+maxMap : int
+currentMap : int
screenWidth2 : int
screenHeight2 : int
tempScreen : BufferedImage
g2 : Graphics2D
+fullScreenOn : boolean
FPS : int
+tileM : TileManager
+keyH : KeyHandler
music : Sound
se : Sound
+cChecker : CollisionChecker
+aSetter : AssetSetter
+ui : UI
+eHandler : EventHandler
config : Config
+pFinder : PathFinder
eManager : EnvironmentManager
map : Map
saveLoad : SaveLoad
+eGenerator : EntityGenerator
+csManager : CutsceneManager
gameThread : Thread
+player : Player
+obj : Entity[]
+npc : Entity[]

GamePanel (continue)
+monster : Entity[]
+iTile : InteractiveTile[]
+projectile : Entity[]
+particleList : ArrayList<Entity>
entityList : ArrayList<Entity>
+gameState : int
+titleState : int
+playState : int
+pauseState : int
+dialogueState : int
+characterState : int
+optionsState : int
+gameOverState : int
+transitionState : int
+tradeState : int
+sleepState : int
+mapState : int
+cutsceneState : int
+bossBattleOn : boolean
+currentArea : int
+nextArea : int
+outside : int
+indoor : int
+dungeon : int
+setupGame() : void
+resetGame(boolean) : void
+setFullScreen() : void
+startGameThread() : void
+run() : void
+update() : void
+drawToTempScreen() : void
+drawToScreen() : void
+playMusic(int) : void
+stopMusic() : void
+playSE(i) : void
+changeArea() : void
+removeTempEntity() : void

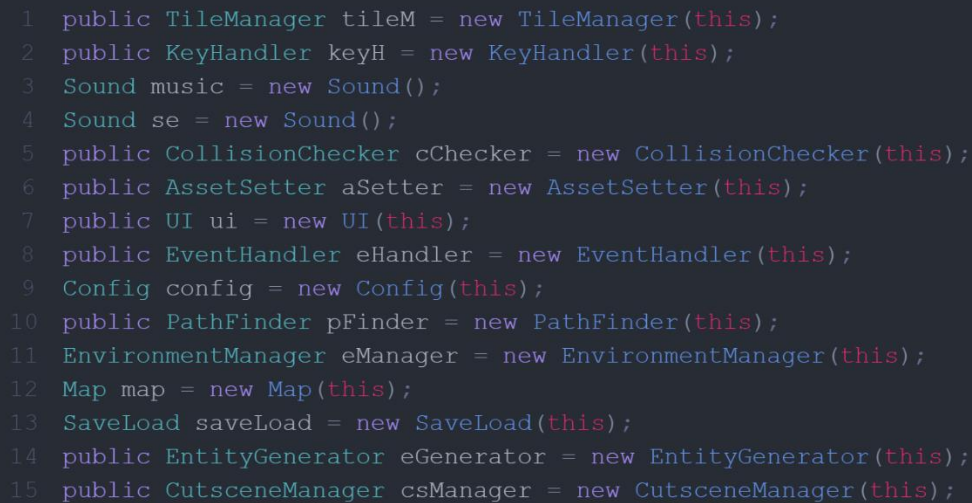
คำอธิบายการอ่าน Class diagram



1. ส่วนด้านบนใช้เพื่ตั้งชื่อคลาส
2. อันที่สองใช้เพื่อแสดง Constructor ของคลาส
3. อันที่สามใช้เพื่อแสดงคุณสมบัติของคลาส
4. อันที่สี่ใช้เพื่ออธิบายการดำเนินการที่ดำเนินการโดยคลาส

2.3 อธิบายส่วนของโปรแกรมที่มีหลักการเขียนโปรแกรมเชิงวัตถุ

2.3.1 ตัวอย่างของ Constructor ที่มีภายในโปรแกรม

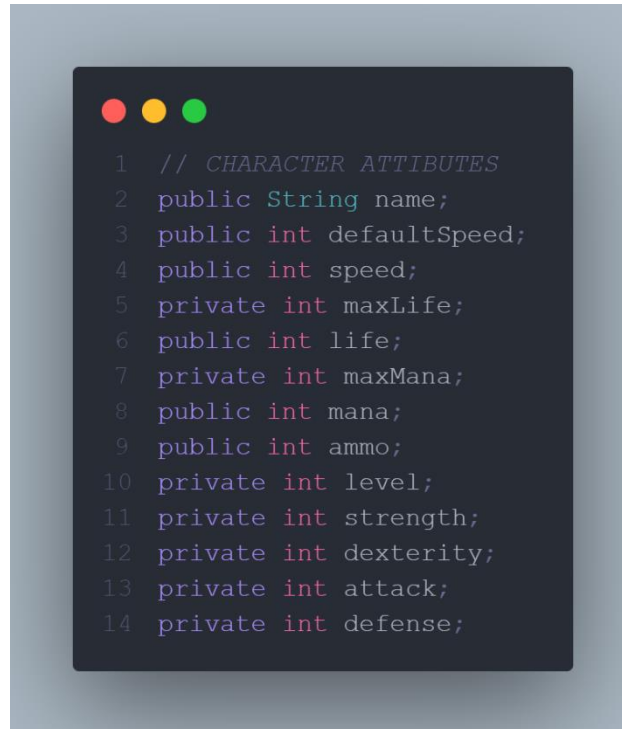


```
1 public TileManager tileM = new TileManager(this);
2 public KeyHandler keyH = new KeyHandler(this);
3 Sound music = new Sound();
4 Sound se = new Sound();
5 public CollisionChecker cChecker = new CollisionChecker(this);
6 public AssetSetter aSetter = new AssetSetter(this);
7 public UI ui = new UI(this);
8 public EventHandler eHandler = new EventHandler(this);
9 Config config = new Config(this);
10 public Pathfinder pFinder = new Pathfinder(this);
11 EnvironmentManager eManager = new EnvironmentManager(this);
12 Map map = new Map(this);
13 SaveLoad saveLoad = new SaveLoad(this);
14 public EntityGenerator eGenerator = new EntityGenerator(this);
15 public CutsceneManager csManager = new CutsceneManager(this);
```

ตัวอย่างนี้อยู่ภายใน Class ที่มีชื่อว่า GamePanel โดย Constructors หรือตัวสร้างคือเมธอดที่ถูกเรียกใช้เมื่อสร้าง Object ซึ่ง Constructors มีหน้าที่กำหนดค่าเริ่มต้นให้กับตัวแปรและประกาศและกำหนดค่าให้กับ Object ที่ถูกสร้างขึ้น เช่น

1. Public TileManager tileM = new TileManager(this); คือการสร้างวัตถุ TileManager และส่งตัวออบเจกต์ปัจจุบันเป็นพารามิเตอร์ โดยที่มีระดับการเข้าถึงคือ public
2. Sound music = new Sound(); - สร้างวัตถุ Sound โดยไม่มีการส่งพารามิเตอร์

2.3.2 ตัวอย่างของ Encapsulation ที่มีภายในโปรแกรม

A code editor window with a dark background and light-colored text. It contains 14 lines of Java code defining character attributes. The code uses public for public attributes and private for private attributes. The attributes are: name (String), defaultSpeed (int), speed (int), maxLife (int), life (int), maxMana (int), mana (int), ammo (int), level (int), strength (int), dexterity (int), attack (int), and defense (int).

```
1 // CHARACTER ATTRIBUTES
2 public String name;
3 public int defaultSpeed;
4 public int speed;
5 private int maxLife;
6 public int life;
7 private int maxMana;
8 public int mana;
9 public int ammo;
10 private int level;
11 private int strength;
12 private int dexterity;
13 private int attack;
14 private int defense;
```

ตัวอย่างนี้คือ Character Attributes บางส่วนของ Entity โดยมีการ Encapsulation ผ่านระดับการเข้าถึงของตัวแปรบางตัวแปร ซึ่งถูกประกาศเป็น private หมายความว่าสามารถเข้าถึงได้เฉพาะภายในคลาสที่ถูกประกาศ โดยต้องมีสร้างเมทอด getter และ setter เพื่อให้สามารถเข้าถึงและตั้งค่าข้อมูลที่ถูกป้องกันไว้ได้

2.3.3 ตัวอย่างของ Inheritance ที่มีภายในโปรแกรม

```
1 public class MON_GreenSlime extends Entity {
2
3     GamePanel gp;
4
5     public MON_GreenSlime(GamePanel gp) {}
6
7     public void getImage() {
8
9         up1 = setup("/monster/greenslime_down_1", gp.tileSize, gp.tileSize);
10        up2 = setup("/monster/greenslime_down_2", gp.tileSize, gp.tileSize);
11        down1 = setup("/monster/greenslime_down_1", gp.tileSize, gp.tileSize);
12        down2 = setup("/monster/greenslime_down_2", gp.tileSize, gp.tileSize);
13        left1 = setup("/monster/greenslime_down_1", gp.tileSize, gp.tileSize);
14        left2 = setup("/monster/greenslime_down_2", gp.tileSize, gp.tileSize);
15        right1 = setup("/monster/greenslime_down_1", gp.tileSize, gp.tileSize);
16        right2 = setup("/monster/greenslime_down_2", gp.tileSize, gp.tileSize);
17    }
18    public void setAction () {
19
20        if(onPath == true) {
21
22            // CHECK IF IT STOP CHASING
23            checkStopChasingOrNot(gp.player, 15, 100);
24
25            // SEARCH DIRECTION TO GO
26            searchPath(getGoalCol(gp.player), getGoalRow(gp.player));
27
28        }
29        else {
30            // CHECK FOR START CHASING
31            checkStartChasingOrNot(gp.player, 5, 100);
32
33            // RANDOM DIRECTION
34            getRandomDirection(120);
35        }
36    }
37 }
```

ตัวอย่างนี้อยู่ภายใน Class ที่มีชื่อว่า MON_GreenSlime โดยมีการ Inheritance มาจาก Class ที่มีชื่อว่า Entity ทำให้ MON_GreenSlime สามารถใช้งานฟังก์ชันและข้อมูลที่ถูกกำหนดใน Entity ได้โดยตรง โดยไม่ต้องนำเข้ามาใหม่

2.4 ส่วนประกอบของ Graphics และ GUI ที่มีภายในโปรแกรม

2.4.1 กราฟิก (Graphics)

java.awt.AlphaComposite: ใช้ในการกำหนดการผสมภาพแบบต่าง ๆ เช่น การผสมภาพด้วยค่าโปร่งใส

java.awt.Color: ใช้ในการจัดการสีและแสดงสีในรูปแบบ RGB.

java.awt.Graphics: เป็นอ็อบเจกต์ที่ใช้ในการวาดรูปภาพและตัวหนังสือบน component

java.awt.Graphics2D: คุณสมบัติเพิ่มเติมสำหรับการวาดภาพจาก java.awt.Graphics

java.awt.RenderingHints: ใช้กำหนดคุณลักษณะการวาดที่ต้องการ เช่น คุณภาพของการวาด

java.awt.BasicStroke: ใช้กำหนดลักษณะของเส้นทางการวาด เช่น ความหนาของเส้น

java.awt.Font: ใช้ในการจัดการตัวหนังสือ เช่น การตั้งความหนา, ขนาด, และลักษณะอื่น ๆ

java.awt.FontFormatException: จัดการข้อผิดพลาดที่เกี่ยวข้องกับรูปแบบ Font ที่ไม่ถูกต้อง

java.awt.image.BufferedImage: เป็นคลาสที่ใช้สร้างและจัดการกับรูปภาพในรูปแบบของ buffer

java.awt.RadialGradientPaint: ใช้สร้างลวดลายแสงแบบรัศมี

java.awt.Rectangle: ใช้ในการจัดการข้อมูลของสี่เหลี่ยมผืนผ้า

java.awt.Dimension: ใช้ในการระบุขนาดของ component หรือขอบเขตของพื้นที่

java.awt.GraphicsDevice: ใช้ในการจัดการอุปกรณ์กราฟิก (เช่น หน้าจอ) ในเครื่องคอมพิวเตอร์

java.awt.GraphicsEnvironment: ใช้เพื่อให้เข้าถึงข้อมูลเกี่ยวกับกรณีพิเศษของกราฟิกที่ระบบใช้งาน

2.4.2 GUI (Graphical User Interface)

javax.swing.ImageIcon: ใช้ในการโหลดรูปภาพจากไฟล์

javax.swing.JFrame: เป็นคลาสที่ใช้สร้างหน้าต่าง GUI หลักของแอปพลิเคชัน

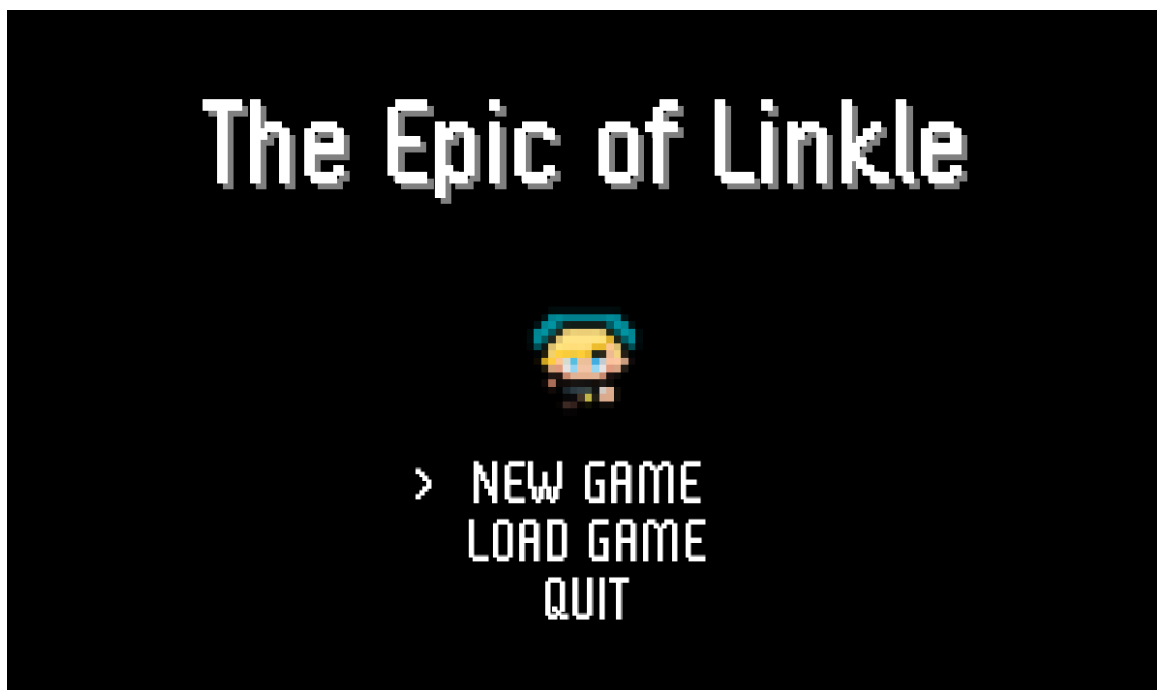
2.5 Event Handling ที่มีภายในโปรแกรม

Event Handling ของโปรแกรมนี้อยู่ใน Class ที่มีชื่อว่า KeyHandler โดยมีการ implements KeyListener โดยมีเมทอดที่ต้อง implement คือ keyTyped(), keyPressed(), และ keyReleased() โดยเมทอดที่สำคัญคือ keyPressed และ keyReleased

keyPressed : เมทอดนี้ถูกเรียกเมื่อมีการกดปุ่มบนคีย์บอร์ด ใช้ในการตรวจสอบปุ่มที่ถูกกด และทำการปรับค่าตัวแปรต่าง ๆ ตามปุ่มที่ถูกกด

keyReleased : เมทอดนี้ถูกเรียกเมื่อปุ่มบนคีย์บอร์ดถูกปล่อยหลังจากที่ถูกกดไว้ ใช้ในการตรวจสอบปุ่มที่ถูกปล่อย และทำการปรับค่าตัวแปรต่าง ๆ ตามปุ่มที่ถูกปล่อย

ตัวอย่างโค้ดในการใช้งาน Event Handling ในสถานการณ์ที่แตกต่างกัน เช่น ในโหมด titleState() เมื่อกดปุ่ม W หรือ S จะทำให้ตัวแปร gp.ui.commandNum ถูกเพิ่มหรือลดลง และเมื่อกดปุ่ม Enter แล้วตัวแปร gp.ui.commandNum ถูกตรวจสอบเพื่อทำการเปลี่ยนสถานะเกม (เช่น เริ่มเกม, โหลดเกม, หรือออกจากเกม)



2.6 อัลกอริทึมที่สำคัญในโปรแกรม

หนึ่งในอัลกอริทึมที่สำคัญในโปรแกรมมีชื่อว่า A* (A-Star) เป็นการใช้วิธีการค้นหาเส้นทางเพื่อหาเส้นทางที่สั้นที่สุดจากตำแหน่งเริ่มต้นไปยังตำแหน่งปลายทางในตารางขนาดใหญ่ (grid) ซึ่งแต่ละช่องในตารางมีค่าความยาก (cost) ที่ต่างกัน ซึ่งอยู่ในส่วน Package ที่มีชื่อว่า ai

Node : คลาสนี้เก็บข้อมูลของแต่ละช่องในตารางที่ใช้ในการคำนวณ A* Pathfinding โดยมีข้อมูลเช่น col และ row แทนตำแหน่งของช่อง, gCost, hCost, และ fCost ที่ใช้ในการคำนวณค่าความยาก, และตัวแปร solid, open, และ checked ที่บอกถึงสถานะของช่องนั้น ๆ

PathFinder : คลาสนี้คือเมธอดของการค้นหาเส้นทาง A* โดยมีเมธอดที่สำคัญได้แก่

- instantiateNodes(): สร้างตารางขนาดใหญ่ที่ใช้ในการคำนวณ A* Pathfinding
- resetNodes(): รีเซ็ตสถานะของช่องทั้งหมดให้เป็นสถานะเริ่มต้น
- setNodes(): ตั้งค่าตำแหน่งเริ่มต้นและปลายทางของการค้นหาเส้นทาง A* พร้อมกับกำหนดของ entity ที่มีผลต่อการคำนวณค่าความยากของแต่ละช่อง
- getCost(Node node): คำนวณค่า gCost, hCost, และ fCost ของช่องที่กำหนด
- search(): ทำการค้นหาเส้นทางโดยใช้อัลกอริทึม A*
- openNode(Node node): เปิดช่องที่ยังไม่ได้เปิดให้ตามค่าความยากและเพิ่มลงใน openList
- trackThePath(): ติดตามเส้นทางที่ได้จากผลลัพธ์ของการค้นหาเส้นทาง A*

บทที่ 3

สรุป

3.1 ปัญหาที่พบระหว่างการพัฒนา

- การควบคุมการประสานงานระหว่างเหตุการณ์ เช่น การเดินพร้อมกับการโจมตี
- ปัญหาทางด้านการเช็ก collision ทำให้ตัวละครอัดมัมไม่สามารถเดินออกมาได้
- ปัญหาด้านการเรียงลำดับการวาดภาพ เช่น วาด NPC ทับตัวละครผู้เล่น
- ปัญหาทางด้านแหล่งเสียงที่จัดการกับการทำงานของเอฟเฟกต์แต่ละอย่างในเวลาเดียวกัน

3.2 จุดเด่นของโปรแกรม

โปรแกรมนั้นมีการแบ่งส่วนการทำงานของแต่ละ Class อย่างชัดเจนทำให้การเพิ่ม Map, Monster, NPC, OBJ และ Item เข้ามาใหม่นั้นสามารถทำได้ง่ายดาย

3.3 คำแนะนำสำหรับน้องรุ่นต่อไป

- ช่วงต้นเทอมอยากทำอะไรให้รีบทำ และให้ศึกษาไว้ก่อนล่วงหน้าและทำโปรเจกก่อนได้เลย เพราะต้องคุณจจะรอนสอนจบยันบทสุดท้ายแล้วคุณค่อยทำโปรเจก ช่วงนั้นจะติดช่วงสอบพอดี แล้วคุณจะต้องเลือกระหว่าง เวลาทำโปรเจก, เวลาอ่านหนังสือ และเวลานอน ว่าสิ่งไหนสำคัญที่

GitHub: <https://github.com/Eshwaldz/The-Epic-of-Linkle>