<TA.BEAM'S ADVENTURE>

Created by

Teejuta Sriwaranon 6632102412

Nattarin Chetpattananondh 6632075121

2110215 Programming Methodology

Semester 2 Year 2023

Chulalongkorn University

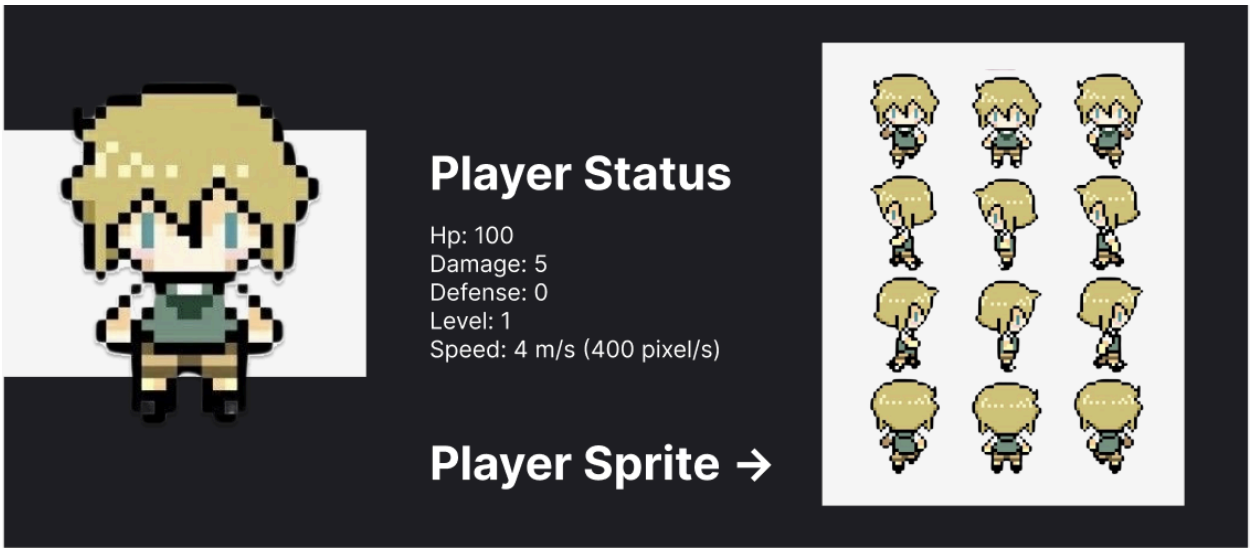
< TA.BEAM’S ADVENTURE>

Introduction

TA.Beam’s adventure is a RPG game. Player takes the role of the little boy character, TA.Beam. This game has 2 stages, farming, and boss fighting. In the farming stage, player can gain extra coins to buy more items and equipment. In the boss fighting stage, player has to conquer the boss by the classic game, “ROCK PAPER SCISSORS”.

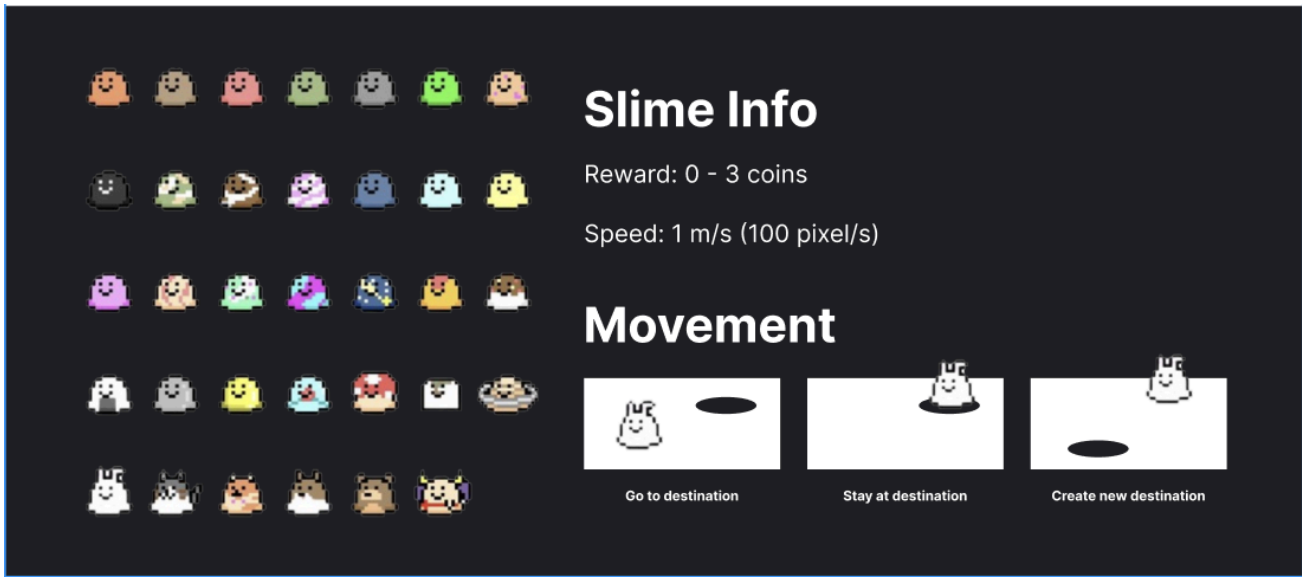
Game components

Player



Player sprite to create walking animation

Slime



Kill slime to gain extra coins.

Bosses

There are 4 bosses in this game, including chicken, pig, sheep, and rabbit, where each boss has a different most likely pick.

Equipment

<Armor>

Armor can increase player defense. If player loses in that turn, Armor durability will be decreased by 1.

<Weapon>

Weapon can increase player attack power, if player wins in that turn, Weapon durability will be decreased by 1.

<Amulet>

Amulet can increase player chance to win. In each turn, Amulet durability will be decreased by 1.

Item

<Hp Potion>

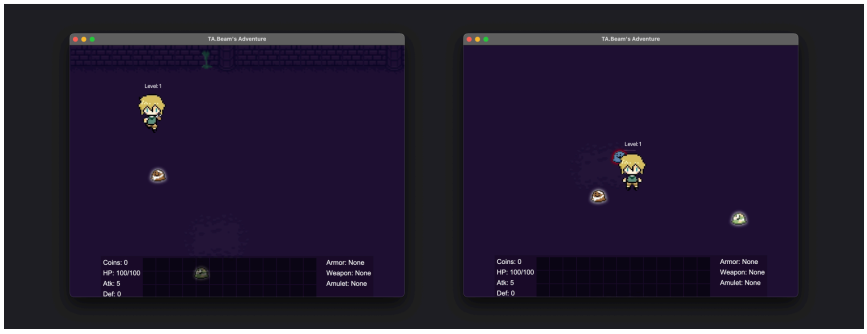
Player can use Hp Potion to gain 10 Hp.

<Refine Iron>

Refine Iron is used to upgrade equipment.

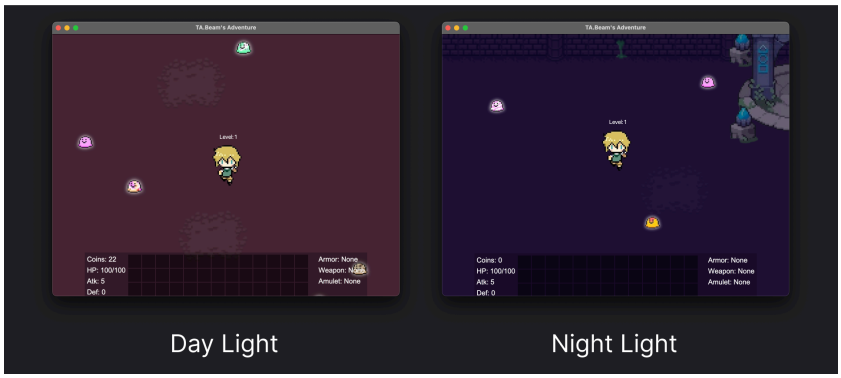
Game system

Camera



Camera follows the player while walking but does not go outside the border.

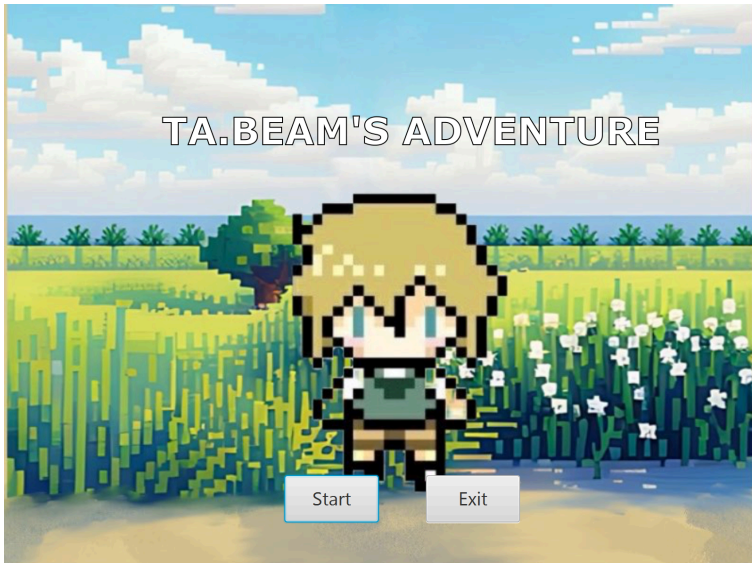
Day & Night system



The game's graphics change depending on day and night.

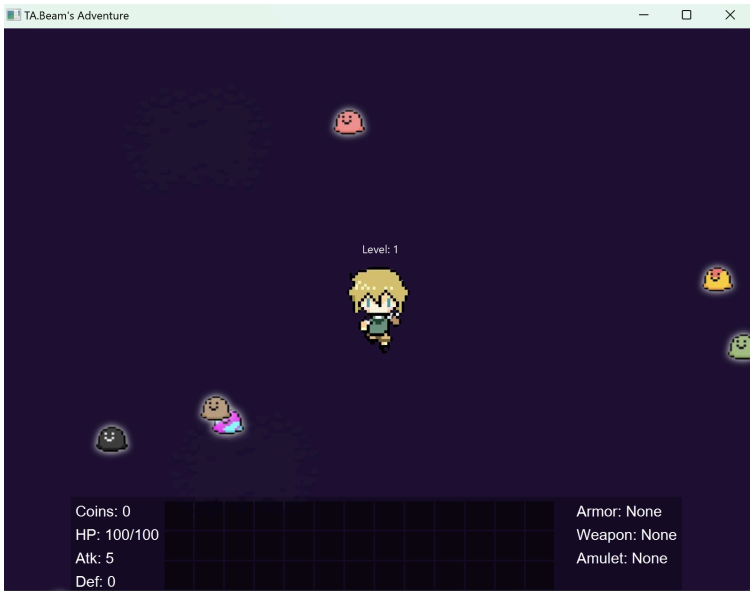
How to play

< First window >



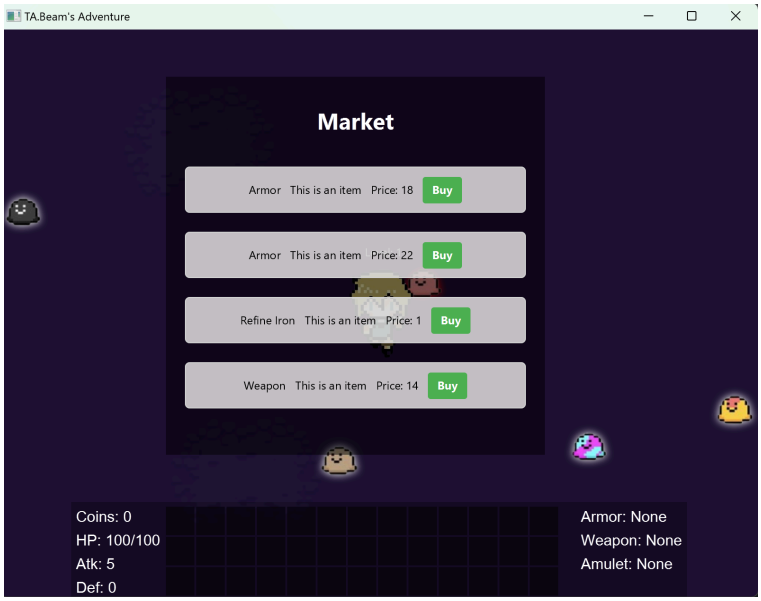
After clicking the start button, player will be brought to the farming stage.

<Farming Stage>



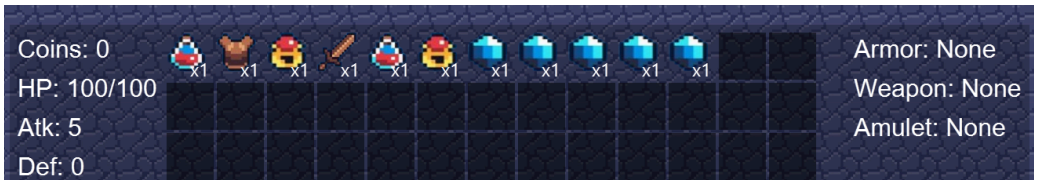
Player can explore the map by pressing “W, A, S, D” keys. Player can defeat the slime by clicking it. After defeating slime, player will gain 0-3 coins randomly. Player can buy items and equipment by pressing the “M” key.

<Market page>



Market page will be refreshed every time player has conquered the boss. Market will random and sell 4 stuffs including different tiers of equipment, Hp Potion ,and Refine Iron.Item prices are also random.

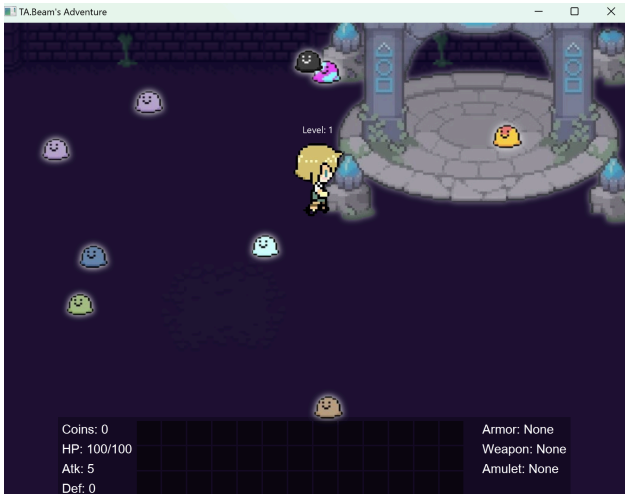
<Inventory>



Player Inventory is shown below. Player can click on icons to use items or wear equipment. After wearing equipment, player can upgrade them by pressing the following keys: “1” for Armor, “2” for Weapon, and “3” for Amulet, each upgrading needs 1 Refine Iron. Player can take off equipment by clicking at the equipment label at the right hand side.

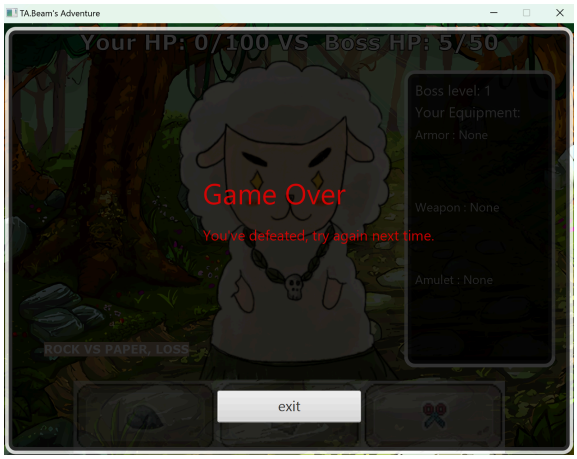
<Teleport door>

If player click on the door, player will be teleported to another dimension and go to the boss fighting stage.



<Boss Fighting Stage>

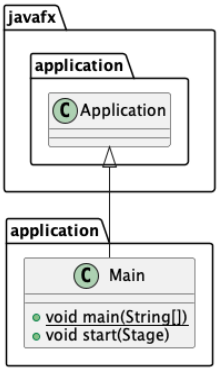
Player and boss fighting by rock paper scissors game. Every time player has conquered the boss, player level will be increased by 1, gain items reward, and gain coins depending on Boss level. Boss level is relative with player level.



If player Hp equals zero, the game is over and player is brought to the first window.

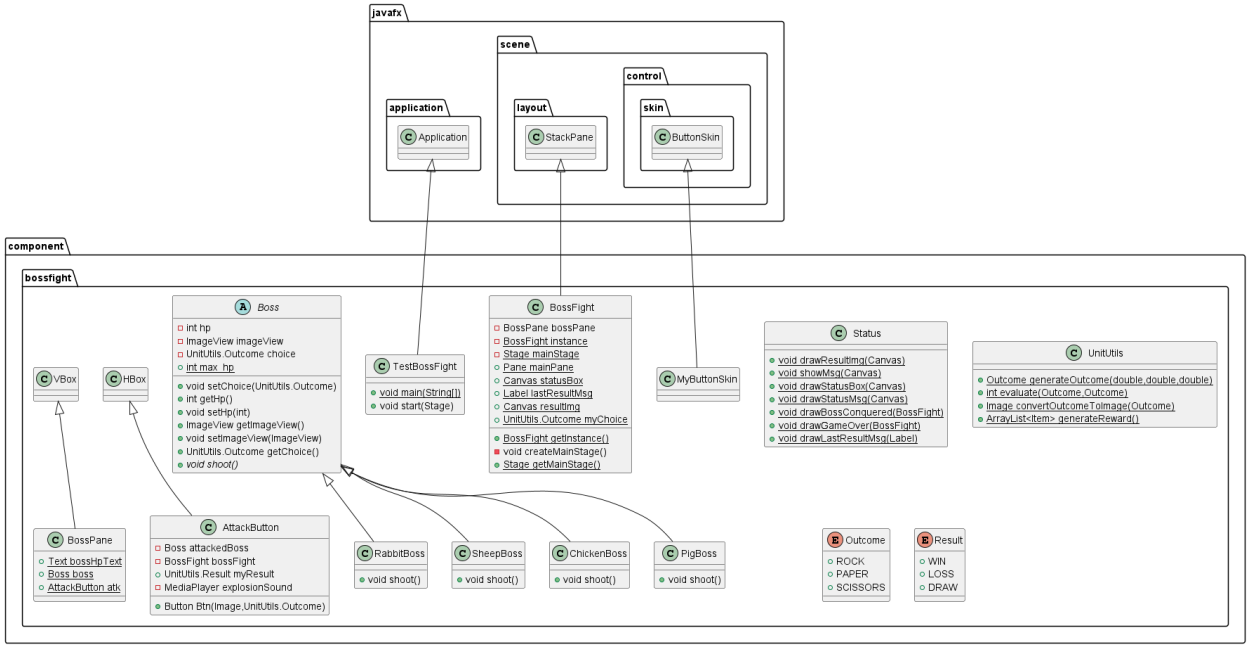
Class diagram

Package application

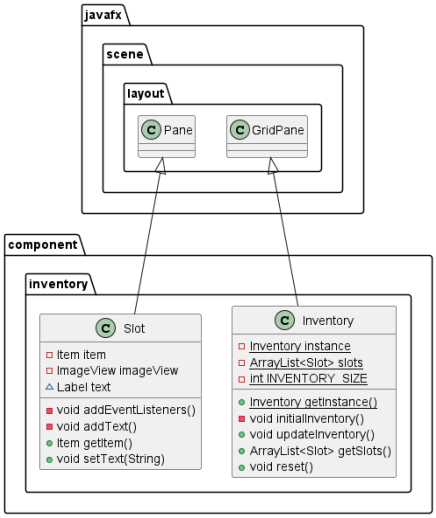


Package component

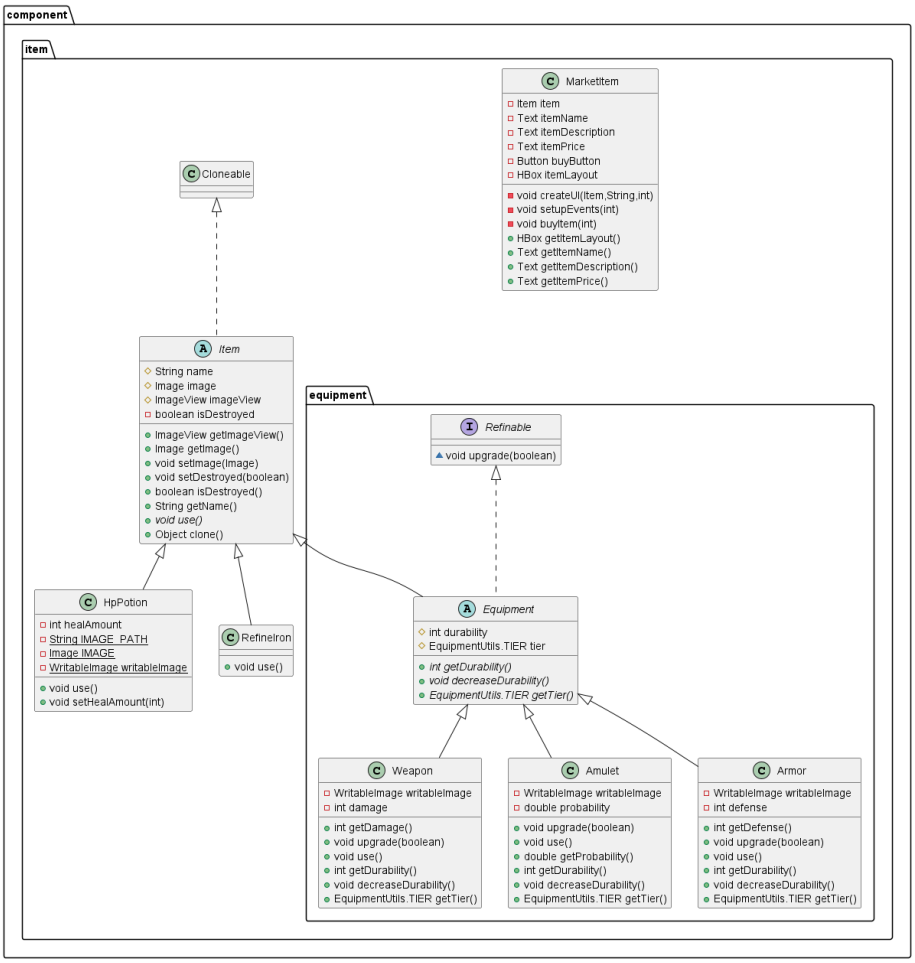
Package bossfight



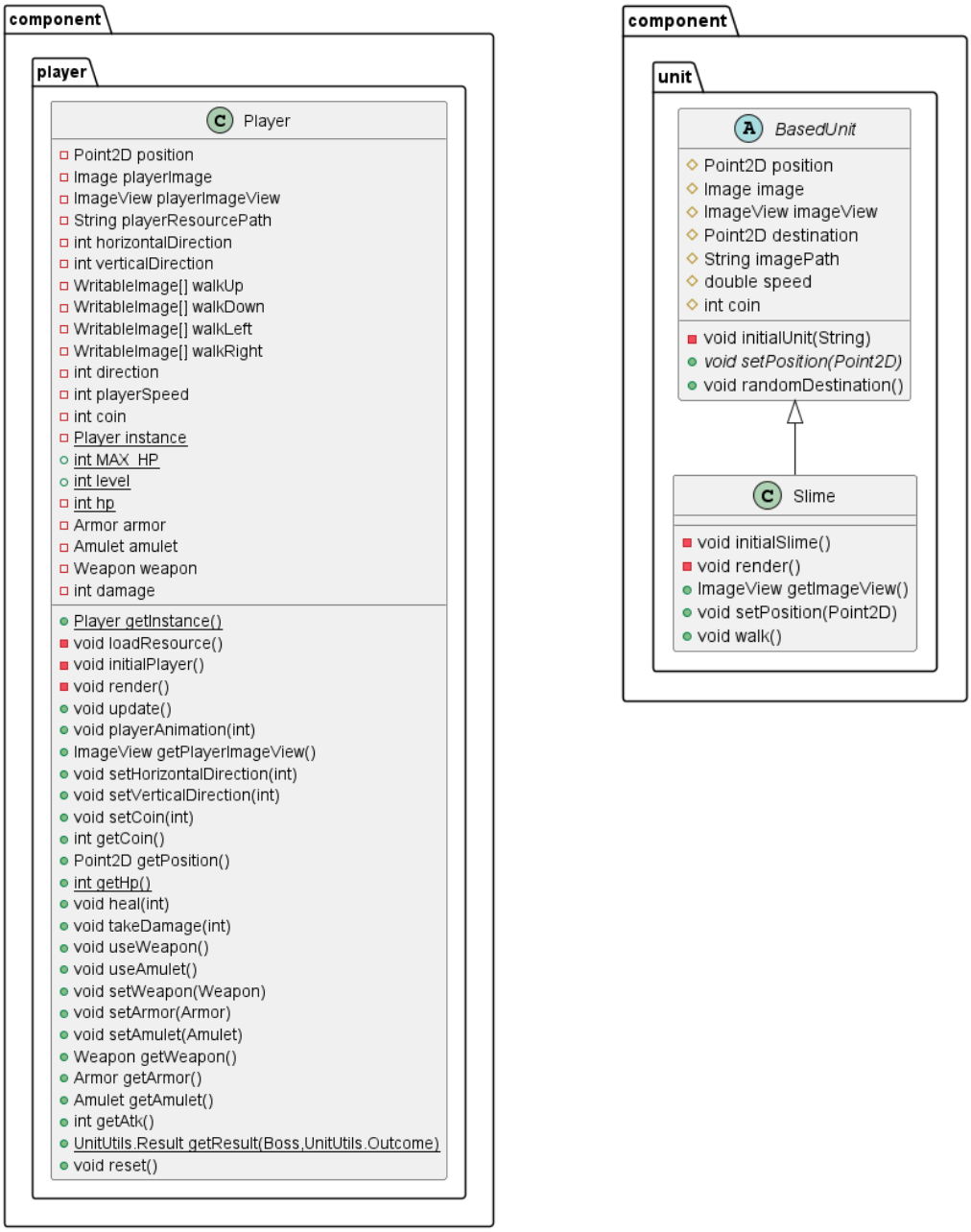
Package inventory



Package item



Package player & unit



1. Package application

1.1 Class Main

1.1.1 Methods

+	void main(String args[ ])	Main method
+	void start(Stage primaryStage)	Start Method

2. Package component

2.1 Package bossfight

2.1.1 Class AttackButton extends HBox

2.1.1.1 Fields

-	Boss attackedBoss	The boss that player is attacking.
-	BossFight bossFight	The boss fighting Pane.
+	UnitUtils.Result myResult	The result in each turn.
-	MediaPlayer explosionSound	Sound when the player attacks.

2.1.1.2 Constructor

+	AttackButton(Boss boss, BossFight bossFight)	Initialize the attack button according to the boss.

2.1.1.3 Methods

+	Button Btn(Image image, UnitUtils.Outcome myChoice)	An Image of button

2.1.2 *abstract Class Boss*

2.1.2.1 Fields

-	int hp	Boss Hp

- ImageView imageView	Boss Image
- UnitUtils.Outcome choice	Boss option in each turn
+ int max_hp	Boss max Hp

2.1.2.2 Constructor

+ Boss()	Initialize fields.

2.1.2.3 Methods

+ *abstract void shoot()*	Used to generate outcome according to each boss
+ getter/setter	

2.1.3 Class BossFight extends StackPane

2.1.3.1 Fields

- BossPane bossPane	Pane that represent boss
- BossFight instance	Instance
- Stage mainStage	BossFight Stage
+ Pane mainPane	BossFight Pane
+ Canvas statusBox	Status that is shown at the RHS
+ Label lastResultMsg	Last result message in each turn
+ Canvas resultImg	Result icon when hit attack button
+ UnitUtils.Outcome myChoice	Player choice in each turn

2.1.3.2 Constructor

- BossFight()	Create the main stage

2.1.3.3 Methods

+ BossFight getInstance()	Initialize instance
- void createMainStage()	Create the stage
+ Stage getMainStage()	Getter for mainStage

2.1.4 Class BossPane extends VBox

2.1.4.1 Fields

+ Text bossHpText	Player and boss Hp text
+ Boss boss	Attacking boss
+ AttackButton atk	The Pane attack button

2.1.4.2 Constructor

+ BossPane(Boss boss, BossFight boosFight)	Create UI

2.1.5 Class ChickenBoss extends Boss

2.1.5.1 Constructor

+ ChickenBoss()	Initialize boss image

2.1.5.2 Method

+ void shoot()	Generate boss outcome

2.1.6 Class MyButtonSkin extends ButtonSkin

2.1.6.1 Constructor

+ MyButtonSkin(Button control)	Style for Attack Button

2.1.7 Class PigBoss extends Boss

2.1.7.1 Constructor

+ PigBoss()	Initialize boss image

2.1.7.2 Method

+ void shoot()	Generate boss outcome

2.1.8 Class RabbitBoss extends Boss

2.1.8.1 Constructor

+ RabbitBoss()	Initialize boss image

2.1.8.2 Method

+ void shoot()	Generate boss outcome

2.1.9 Class SheepBoss extends Boss

2.1.9.1 Constructor

+ SheepBoss()	Initialize boss image

2.1.9.2 Method

+ void shoot()	Generate boss outcome

2.1.10 Class Status

2.1.10.1 Methods

+ void drawResultImg(Canvas canvas)	Create result image in each turn
+ void showMsg(Canvas t)	Show and hide result image
+ void drawStatusBox(Canvas canvas)	Draw status box
+ void drawStatusMsg(Canvas canvas)	Draw current wearing equipment

+ void drawBossConquered(BossFight bossFight)	Create UI when player wins
+ void drawGameOver(BossFight bossFight)	Create Game over UI
+ void drawLastResultMsg(Label t)	Draw player last result message

2.1.11 Class TestBossFight extends Application

2.1.11.1 Methods

+ void main(String args[ ])	Main Method
+ void start(Stage primaryStage)	Start Method

2.1.12 Class UnitUtils

2.1.12.1 Enums

+ enum Outcome	ROCK, PAPER, SCISSORS
+ enum Result	WIN, LOSS, DRAW

2.1.12.2 Methods

+ Outcome generateOutcome(double probRock, double probPaper, double probScissors)	Generate an outcome depends on probability
+ int evaluate(Output player, Output computer)	Evaluate the result in the rock scissors paper game
+ Image convertOutcomeToImage(Outcome outcome)	Convert an outcome to image
+ ArrayList<Item> generateReward()	Generate rewards when player wins

2.2 Package inventory

2.2.1 Class Inventory extends GridPane

2.2.1.1 Fields

- Inventory instance	Class instance
- ArrayList<Slot> slots	Slots in player's inventory

- final int INVENTORY_SIZE	The inventory size

2.2.1.2 Constructor

- Inventory()	- Initial inventory - Update inventory

2.2.1.3 Methods

+ Inventory getInstance()	Getter for Inventory instance
- void initialInventory()	Create UI for inventory
+ void updateInventory()	Update player's inventory
+ ArrayList<Slot> getSlots()	Getter for slots
+ void reset()	Reset player's inventory

2.2.2 Class Slot extends Pane

2.2.2.1 Fields

- Item item	Item in the slot
- ImageView imageView	Image of an item

2.2.2.2 Constructor

+ Slot(Item item)	Create item image in slot

2.2.2.3 Methods

- void addEventListeners()	Add listeners to use item
- void addText()	Create item amount text
+ Item getItem()	Getter for an item in slot

2.3 Package item

2.3.1 Package equipment

2.3.1.1 Class Amulet extends Equipment

2.3.1.1.1 Fields

- WritableImage writableImage	An image for amulet
- double probability	Probability to increase player winning chance (depends on tier)

2.3.1.1.2 Constructor

+ Amulet()	Initialize fields

2.3.1.1.3 Methods

+ void upgrade(boolean useIron)	Set fields when amulet is upgraded
+ double getProbability()	Getter for probability
+ void use()	Equip Amulet
+ getter / setter	

2.3.1.2 Class Armor extends Equipment

2.3.1.2.1 Fields

- WritableImage writableImage	An image for armor
- int defense	Armor defense (depends on tier)

2.3.1.2.2 Constructor

+ Armor()	Initialize fields

2.3.1.2.3 Methods

+ void upgrade(boolean useIron)	Set fields when an armor is upgraded
+ int getDefense()	Getter for armor defense

2.3.1.3 *abstract class Equipment extends Item implements Refinable*

2.3.1.3.1 Fields

# int durability	Equipment durability
# EquipmentUtils.TIER tier	Equipment tier

2.3.1.3.2 Constructor

+ Equipment(String name, Image image)	Initialize fields

2.3.1.3.3 Methods

+ *abstract int getDurability()*	Getter for equipment durability
+ *abstract void decreaseDurability()*	Decrease equipment durability
+ *abstract EquipmentUtils.TIER getTier()*	Getter equipment tier
+ *abstract void use()*	Equip Amulet
+ getter / setter	

2.3.1.4 Interface Refinable

2.3.1.4.1 Method

+ void upgrade(boolean useIron)	Upgrade method

2.3.1.5 Class Weapon extends Equipment

2.3.1.5.1 Fields

- WritableImage writableImage	An image for Weapon
- int damage	Weapon Damage (depends on tier)

2.3.1.5.2 Constructor

+ Weapon()	Initialize fields

2.3.1.5.3 Methods

+ int getDamage()	Return weapon damage
+ void upgrade(boolean useIron)	Set fields when a weapon is upgraded
+ void use()	Equip Amulet
+ getter / setter	

2.3.2 Class HpPotion extends Item

2.3.2.1 Fields

- int healAmount	Heal amount
- final String IMAGE_PATH	An image path
- final Image IMAGE	An image of Item sprite
- WritableImage writableImage	An image for Hp Potion

2.3.2.2 Constructor

+ HpPotion(int healAmount)	Initialize fields and set healAmount to the input value.

2.3.2.3 Methods

+ void use()	Heal Player HP
+ void setHealAmount(int healAmount)	Set healAmount

2.3.3 *abstract Class Item implements Cloneable*

2.3.3.1 Fields

# String name	An Item name
# Image image	An Item Image
# ImageView imageView	An Item ImageView
- boolean isDestroyed	To check if the item was destroyed

2.3.3.2 Constructor

+ Item(String name, Image image)	Initialize fields and set item information

2.3.3.3 Methods

+ *abstract void use()*	Use Method
+ Object clone()	Clone Method
+ getter / setter	

2.3.4 Class MarketItem

2.3.4.1 Fields

- Item item	An Item information
- Text itemName	Text to show the item name
- Text itemDescription	Text to show the item description
- Text itemPrice	Text to show the item price
- Button buyButton	Button to buy the item
- HBox itemLayout	HBox to display information

2.3.4.2 Constructor

+ MarketItem(Item item, String itemDescription, int itemPrice)	Initialize fields

2.3.4.3 Methods

- void createUI(Item item, String itemDescription, int itemPrice)	Create An UI for MarketItem
- void setupEvents(int itemPrice)	Set itemPrice
- void buyItem(int itemPrice)	Set event when player buy the Item If the player does not have enough money, then show “not enough money” with red text
+ getter / setter	

2.3.5 Class RefineIron

2.3.5.1 Constructor

+ RefineIron()	Initialize class

2.3.5.2 Method

+ void use()	Use method

2.4 Package player

2.4.1 Fields

- Point2D position	A player position (x, y)
- Image playerImage	A player image
- ImageView playerImageView	A player imageView
- String playerResourcePath	A player image Path
- int horizontalDirection	A x direction controller
- int verticalDirection	A y direction controller
- WritableImage[ ] walkUp	An Array containing animation images for walk up
- WritableImage[ ] walkDown	An Array containing animation images for walk down

- WritableImage[ ] walkLeft	An Array containing animation images for walk left
- WritableImage[ ] walkRight	An Array containing animation images for walk right
- int direction	A directory to play an animation controller
- int playerSpeed	Player speed
- int coin	Player coin
- Player instance	Player instance (has only one instance in the game)
+ final int MAX_HP	Player max hp
- Armor armor	Player Armor
- Amulet amulet	Player Amulet
- Weapon weapon	Player Weapon
- int damage	Player damage

2.4.2 Constructor

- Player()	initialize fields and render player sprite

2.4.3 Methods

+ Player getInstance()	Return Player instance
- void loadResource()	Load Animation Sprite from Resource directory
- void initialPlayer()	Initialize fields
- void render()	Render Player Image and Set the animation controller
+ void update()	Update Player' sprite position
+ void playerAnimation(int spriteIndex)	Switch and control (on/off) an animation
+ void heal(int amount)	Increase Player Hp
+ void takeDamage(int amount)	Decrees Player Hp by taking damage and calculating an defense of the armor.

	Reduce armor durability by 1
+ void useWeapon()	Reduce weapon durability by 1
+ void useAmulet()	Reduce amulet durability by 1
+ UnitUtils.Result getResult(Boss boss, UnitUtils.Outcome myChoice)	Determine fighting result
+ void reset()	Reset Player information
+ getters & setters	

2.5 Package unit

2.5.1 *abstract Class BasedUnit*

2.5.1.1 Fields

# Point2D position	An Unit position (x, y)
# Image image	An Unit Image
# ImageView imageView	An Unit imageView
# Point2D destination	An Unit destination position (x, y)
# String imagePath	An image path
# double speed	Unit speed
# int coin	Unit coin

2.5.1.2 Constructor

+ BasedUnit()	Initialize fields
+ BasedUnit(String imagePath)	Initialize fields and set imagePath

2.5.1.3 Methods

- void initialUnit(String imagePath)	Initialize fields and set imagePath

+ *abstract void setPosition(Point2D position)*	Set unit position
+ void randomDestination()	Create destination randomly

2.5.2 Class Slime extends BasedUnit

2.5.2.1 Constructor

+ Slime()	Initialize Fields

2.5.2.2 Methods

- void initialSlime()	Initialize Fields
- void render()	Render Slime Sprite randomly
+ ImageView getImageView()	Return Slime imageView
+ void setPosition(Point2D position)	Set slime position
+ void walk()	Walk to the destination and when reach it then call randomDestination() method

3. Package subscene

3.1 Class InventorySubScene extends SubScene

3.1.1 Fields

- Label coinLabel	Label for player coins
- Label hpLabel	Label for player Hp
- Label atkLabel	Label for player Atk power
- Label defLabel	Label for player defense
- Label armorLabel	Label for wearing armor
- Label weaponLabel	Label for wearing weapon
- Label amuletLabel	Label for wearing amulet

- final int WIDTH	Inventory width
- final int HEIGHT	Inventory height
- InventorySubScene instance	Inventory instance

3.1.2 Constructor

+ InventorySubScene()	Create inventory SubScene (call setupStats(), setupEquipment(), and setupItemSlots())

3.1.3 Methods

+ InventorySubScene getInstances()	Getter for instance
- void setupStats(AnchorPane root)	Set up all stats label
+ void updateStats()	Update all stats label
- void setupItemSlots(AnchorPane root)	Set up item slots in player Inventory
- void setupEquipment(AnchorPane root)	Set up all equipment labels and add listeners
- Label createStyledLabel(String text)	Set style for label

3.2 Class MarketSubScene extends SubScene

3.2.1 Fields

- ArrayList<MarketItem> marketItems	ArrayList of Items in market
- AnchorPane root	Pane that display market page
+ Label notEnoughCoin	Label that is shown when the player coins is not enough to buy the item.

3.2.2 Constructor

+ MarketSucScene()	Set style and initialize market items (call createMarketItems())

3.2.3 Methods

- void createItem()	Random item
+ void createMarketItems()	Add random items to market
+ void toggle()	Set visibility for items

4. Package utils

4.1 Class DayNightLight extends Pane

4.1.1 Fields

- FadeTransiton toNightTransiton	The transition to night
- FadeTransiton toDayTransiton	The transition to day
- Boolean isDay	Boolean to check if it's day or night

4.1.2 Constructor

+ DayNightLight()	Initialize fields

4.1.3 Methods

+ void toNight()	Change transition to night
+ void toDay()	Change transition to day

4.2 EquipmentUtils

4.2.1 Fields

- final String IMAGE_PATH	The first Image path for equipment
- final Image IMAGE	The first Image for equipment
- final String IMAGE_PATH2	The second Image path for equipment

- final Image IMAGE2	The second Image for equipment
+ enum TIER	Tiers for equipment

4.2.2 Methods

+ Image getImage(Equipment equipment)	Getter for equipment image
+ Image getRefineIronImage()	Getter for refine iron image

4.3 TileRenderer

4.3.1 Fields

- final String filePath	Path for map image
- final Image tileSet	Image of tiles in map
- int TILE_SIZE	Map tiles size

4.3.2 Methods

+ Image getTile(char i)	Getter for each tiles image

5. Package view

5.1 Class GameViewManager

5.1.1 Fields

- AnchorPane gamePane	AnchorPane to display game UI
- Scene gameScene	Farming scene
- Stage gameStage	Stage in farming stage
- AnimationTimer gameTimer	Timer to update animation

- MarketSubScene marketSubScene	Market subscene to buy items
- InventorySubScene inventorySubScene	Player's inventory that is shown below
+ final int TILE_SIZE	Tiles size to render map and slime
- Player player	Player character
- ArrayList<Slime> slimes	Slime characters
- Static GameViewManager instance	Game instance
- Text playerLevel	Player level text which is shown above player character

5.1.2 Constructor

+ GameViewManager()	Call initialGameStage()

5.1.3 Methods

+ GameViewManager getInstance()	Getter for instance
- void initialGameStage()	- Create game - Update game
- createDayNightLight()	Toggle between day and night
- void renderSilme(int n)	Render slimes in map
- void renderTile()	Render map tiles
- void createListeners()	Create listeners (walking, buying, using items)
- void createGameStage()	Create game stage
- void createSubScenes()	Create market and inventory sub scene
+ void createNewGame()	Create new game when game starts
- void createTeleport()	Create the teleport door in map
- void createGameLoop()	- Create game animation loop - Set layer for each scene
- void updateSlime()	Makes slime to walk

- void updateCamera()	Make the display to follow player while walking
+ boolean isOutsideGame(Point2D position)	Set player boundary
+ Stage getGameStage()	Getter for game stage
+ void reset()	Reset game

5.2 Class ViewManager

5.2.1 Fields

+ final int WINDOW_WIDTH	Game window width
+ final int WINDOW_HEIGHT	Game window height
- AnchorPane mainPane	AnchorPane to display the first game page
- Stage mainStage	First Stage
- Scene mainScene	First scene
- Button startButton	Start button
- Button exitButton	Exit button
- ViewManager instance	First page instance

5.2.2 Constructor

- ViewManager()	Call createMainStage(), and createButtons

5.2.3 Methods

+ ViewManager getInstance()	Getter for instance
- void createMainStage()	Create stage
+ Stage getMainStage()	Getter for mainStage
- void createButton()	Create star & exit button
- void createStartButton()	Create and set style for start button

- void createExitButton()	Create and set style for exit button