Cinnamon Roll Pony

Created by

Yada Vasinonta 6631311021
Khanapong Kamchoom 6631302421

2110215 Programming Methodology
Semester 2 Year 2023
Chulalongkorn University

Cinnamon Roll Pony

Introduction

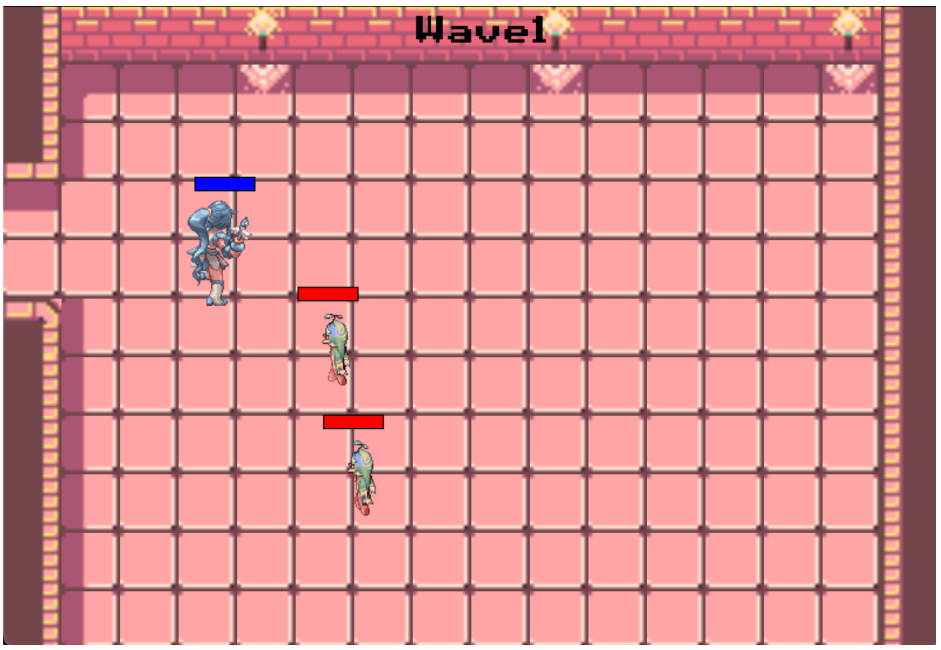
Cinnamon Roll Pony is a simple Fighting Game. The objective is to defeat all the monsters of all waves and obtain victory.

Rules

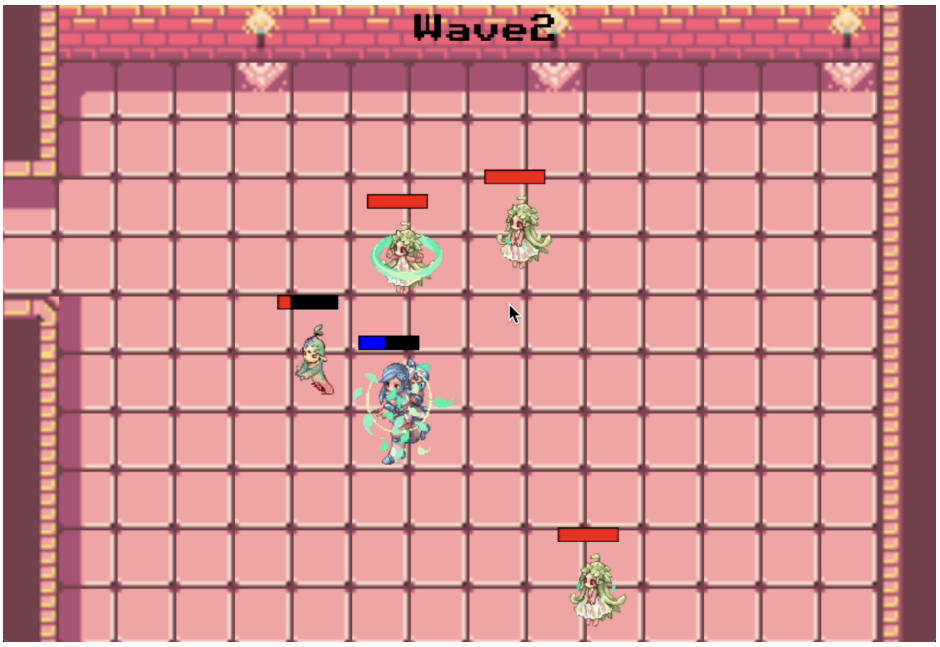
There are 3 waves of monsters, the first wave consisting of the weakest monster, the second wave consisting of weak and medium-strength monsters, and the last wave consisting of the strongest monster. The player needs to attack and eliminate all of them without their HP reaching 0 in order to win.

Example

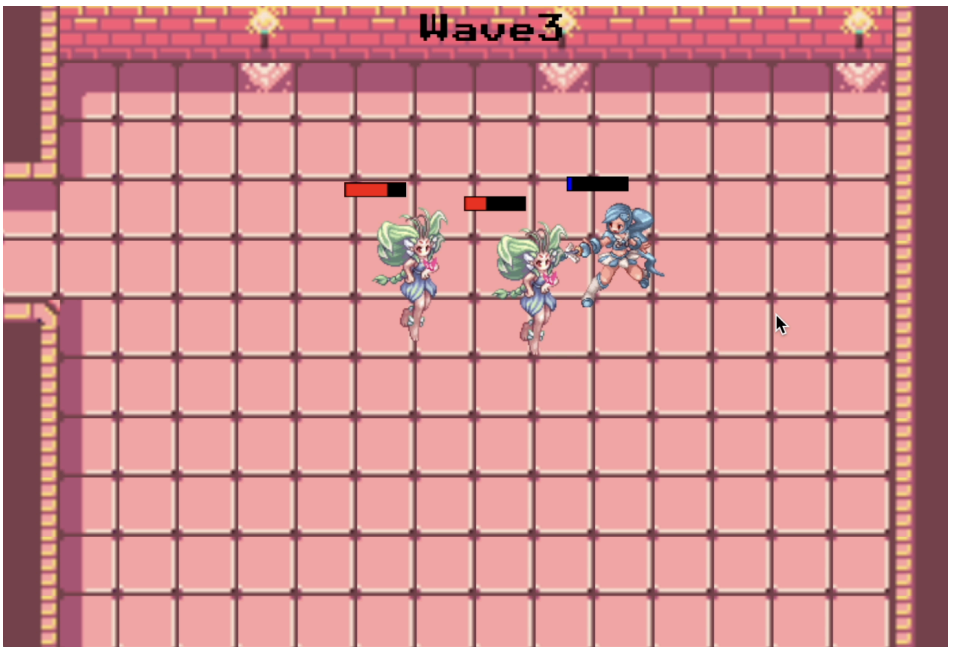
- In wave 1, there are only weak monsters



- If the player defeats all monsters in wave 1, they move on to wave 2, where they fight medium monsters as well.



- If the player defeats all monsters in wave 2, they move on to wave 3, where they fight the strongest monster.

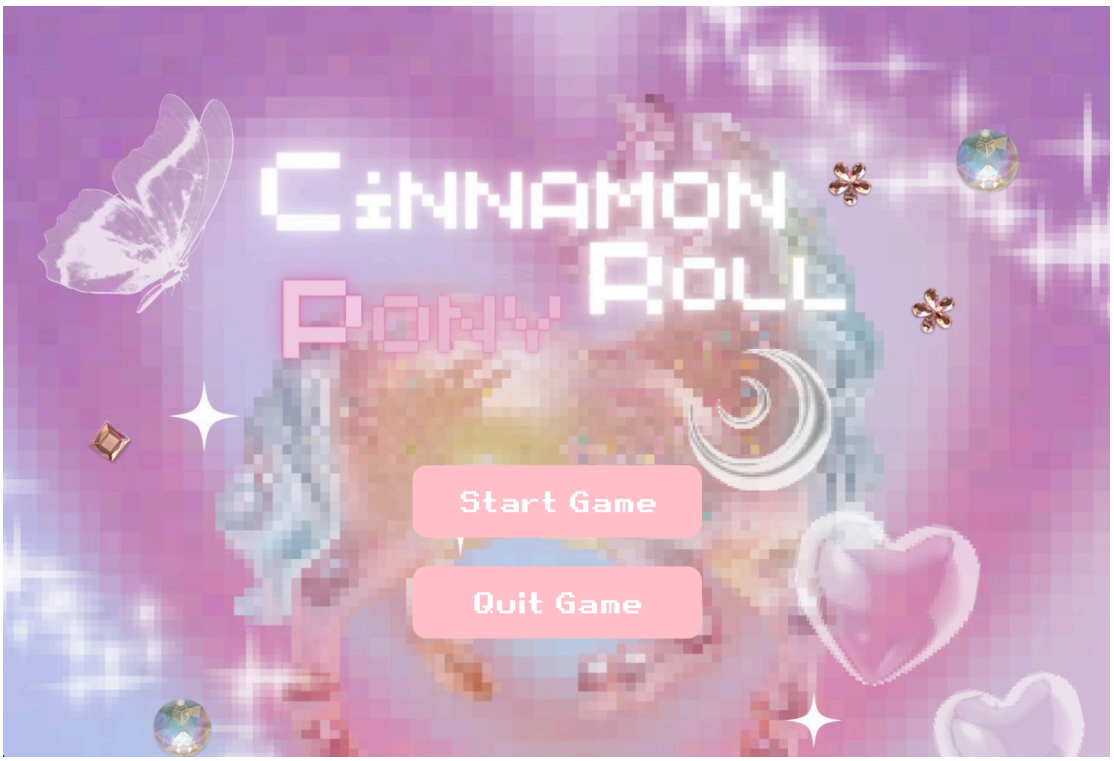


- If the player defeats all monsters in wave 3, they win the game, choose “BACK TO HOME” to go back to the title screen or “QUIT” to quit the game.

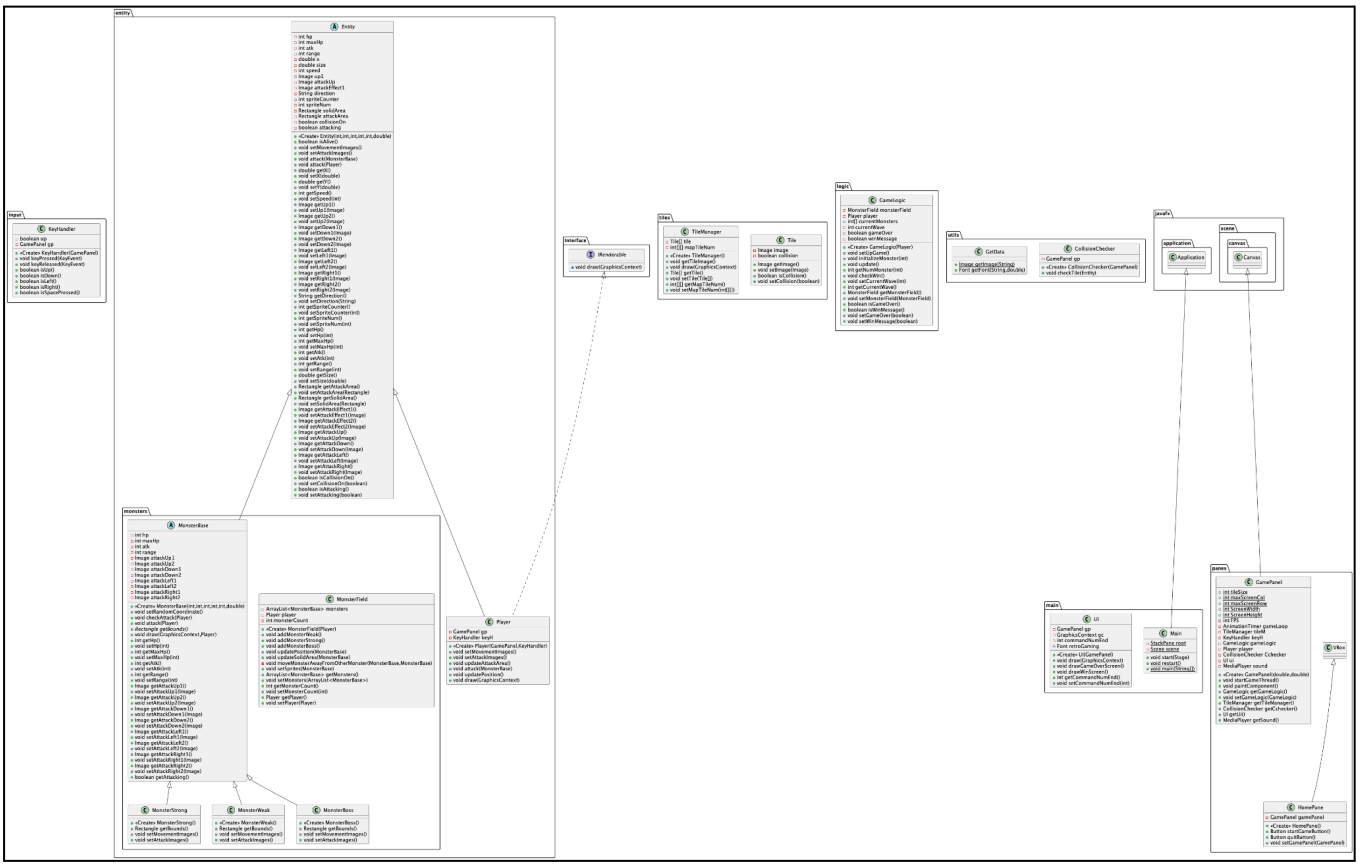
- If the player's HP reaches 0, the game is over, choose "BACK TO HOME" to go back to the title screen or "QUIT" to quit the game.

Title Screen

Choose "START GAME" to play or "QUIT GAME" to exit the game.



Class Diagram



1. Package main

1.1) Class Main extends Application

1.1.1 Methods

+void start(Stage primaryStage)	what to do when the application is launched
+void restart()	- clear the root - add new HomePane() to root
+void main(String[] args)	method to launch the application

1.3) Class UI

1.3.1 Fields

Gamepanel gp	the gamepanel
GraphicContext gc	graphic context
+int commandNumEnd	the number to determine which choice the player selects when the game is won or over

1.3.2 Constructor

+UI(GamePanel gp)	Constructor for UI initialize game panel

1.3.3 Methods

+void draw(GraphicContext gc)	a method to draw the screen when the game is won or over - initialize graphic context - if the game is over, call drawGameOverScreen() - if the game is won, call drawWinScreen()
+void drawGameOverScreen()	draw the Game Over screen - for text “GAME OVER” set the font to Retro Gaming, set the size to 60, and set the color to BLACK - for text “BACK TO HOME” set the

	font to Retro Gaming, set the size to 40 - for text “QUIT” set the font to Retro Gaming, set the size to 40 - if commandNumEnd equals to 0, add “>” in front of “BACK TO HOME” - if commandNumEnd equals to 1, add “>” in front of “QUIT”
+void drawWinScreen()	draw the You Win screen - for text “YOU WIN” set the font to Retro Gaming, set the size to 60, and set the color to BLACK - for text “BACK TO HOME” set the font to Retro Gaming, set the size to 40 - for text “QUIT” set the font to Retro Gaming, set the size to 40 - if commandNumEnd equals to 0, add “>” in front of “BACK TO HOME” - if commandNumEnd equals to 1, add “>” in front of “QUIT”
generate getter and setter for commandNumEnd	

2. Package entity

2.1) *Abstract Class Entity*

This class is a superclass for all entities

2.1.1 Fields

-int hp	entity’s hp
-int maxHp	entity’s max hp
-int atk	entity’s atk
-int range	entity’s range
-int double x,y	position on the x and y axis of the entity on the game window
-double size	entity’s size
-int speed	entity’s speed
-Image up1, up2, down1, down2, left1, left2, right1, right2	images for the entity when it is moving in different directions

-Image attackUp attackDown, attackLeft, attackRight	images for the entity when it is attacking in different directions
-Image attackEffect1, attackEffect2	images for attack effects
-String direction	determine which direction an entity is faced towards
-int spriteCounter	count the spriteNum
-int spriteNum	determine the sprite that will be shown, having multiple sprites for one action can make the character move more naturally.
-Rectangle solidArea	area for detecting collision of the entity
-Rectangle attackArea	attack area of the entity
-boolean collisionOn	check if the entity should collide with the tile or not set to false
-boolean attacking	check if the entity is attacking set to false

2.1.2 Constructor

+Entity(int hp, int maxHp, int atk, int range, int speed, double size)	Constructor for Entity initialize all the parameters

2.1.3 Methods

+boolean isAlive()	if HP is greater than 0, return true. If not, return false
+void setMovementImages()	set images for the movements of entities
+void setAttackImages()	set images for the attack effects of entities
+void attack(MonsterBase monster)	attack method for player
+void attack(Player player)	attack method for monsters
generate getters and setters of all fields	if hp is less than 0, set hp to 0

2.2) Class Player extends Entity implements IRenderable

2.2.1 Fields

-GamePanel gp	the game panel
-KeyHandler keyH	handles each input from the player

2.2.2 Constructor

+Player(GamePanel gp, KeuHandler keyH)	Constructor for Player - set hp to 100. - set max hp to 100. - set attack to 10. - set attack range to 30. - set speed to 4. - set size to 2 * GamePanel.tileSize. - initialize gamepanel and keyhandler - set X to 20 - set Y to 150 - set direction to “right” - set solidArea to Rectangle(0, 0, 48, 48)

2.2.3 Methods

+void setMovementImages()	set the player images for up1, up2, down1, down2, left1, left2, right1, right2
+void setAttackImages()	set the player images for attackUp, attackDown, attackLeft, attackRight
+void updateAttackArea()	- update the area that the player is able to attack from their position - set the attackArea - setFill to Color.BLACK
+void attack(MonsterBase monster)	if the player is attacking, check if the player’s attackArea intersects with the monster’s solidArea. If they intersects, deduct the monster’s hp by the player’s atk
+void updatePosition()	update the player’s direction based on the input - if keyH.isUp(), set direction to “up” - if keyH.isDown(), set direction to “down” - if keyH.isLeft(), set direction to “left” - if keyH.isRight(), set direction to “right” then, check the if the enter key is press for

	attack or not - if keyH.isEnterPressed(), set attacking to true, if not, set attacking to false update the player's position based on the direction and collision - set collisionOn to false first - call checkTile from CollisionChecker to check if collision should be on or not - if collisionOn is false, update the player's position based on the player's direction and speed - update spriteNum and spriteCounter to make the player animation move naturally check if player is alive - if the player is dead, set gameOver (from GameLogic) to true
+void draw(GraphicContext gc)	a method to draw the player and player's hp bar - create a variable image to store the player's image, set it to null - call getPlayerImage() and getPlayerAttackImage() - determine the image according to player's direction, spriteNum, and attacking - draw the hp bar - draw the player with the image we determined

2.3) Package monsters

This package contains all types of monsters, including base class (abstract class).

2.3.1 *Abstract Class MonsterBase*

This class is a base class for all types of monsters.

2.3.1.1 Fields

-int hp	monster's hp
-int maxHp	monster' max hp
-int atk	monster's attack
-int range	monster's attack range

-Image attackUp1, attackUp2, attackDown1, attackDown2, attackLeft1, attackLeft2, attackRight1, attackRight2	images for the monsters when it is attacking in different directions

2.3.1.2 Constructor

+MonsterBase(int hp, int maxHp, int atk, int range, int speed, double size)	Constructor for MonsterBase - initialize all the parameters - call setRandomCoordinate() - set solid area to a rectangle with monster's x, y coordinate and the size of monster's size by monster's size - set direction to right

2.3.1.3 Methods

+void setRandomCoordinate()	set random x, y position for monsters to spawn in the bound of game panel using Math.random()
+void checkAttack(Player player)	if distance between monster and player is less than or equal to monster's attack range minus 10, set attacking to true.
+void attack(Player player)	determine the motion of monsters in each frame when attacking. if the monster completes its attacking moves, deal damage to the player by its attack.
+abstract Rectangle getBounds()	return monster's rectangular bound
+void draw(GraphicContext gc, Player player)	draw monsters on the game screen using GraphicContext class when they are facing different directions. if the monster performs an attack, draw its attack images. if not, draw its movement images. this method also draws its hp bar that shows its current hp.
+getters and setters of all fields	

2.3.2 Class MonsterWeak extends MonsterBase

This class is a subclass of MonsterBase. It's the weakest type of monster.

2.3.2.1 Constructor

+MonsterWeak()	Constructor for MonsterWeak - set hp to 100. - set max hp to 100. - set attack to 3. - set attack range to 50. - set speed to 3. - set size to 1.5 * GamePanel.tileSize. - set movement images, attack images.

2.3.2.1 Methods

+ Rectangle getBounds()	return monster's rectangular bound with monster's x, y coordinate and the size of 1.5 * GamePanel.tileSize by 1.5 * GamePanel.tileSize.

2.3.3 Class MonsterStrong extends MonsterBase

This class is a subclass of MonsterBase. It's the second strongest type of monster.

2.3.3.1 Constructor

+MonsterStrong()	- set hp to 200. - set max hp to 200. - set attack to 8. - set attack range to 80. - set speed to 3. - set size to 1.5 * GamePanel.tileSize. - set movement images, attack images.

2.3.3.2 Methods

+ Rectangle getBounds()	return monster's rectangular bound with monster's x, y coordinate and the size of 2 * GamePanel.tileSize by 2 * GamePanel.tileSize.

2.3.4 Class MonsterBoss extends MonsterBase.

This class is a subclass of MonsterBase. It's the strongest type of monster.

2.3.4.1 Constructor

+MonsterBoss()	- set hp to 700. - set max hp to 700. - set attack to 10. - set attack range to 80. - set speed to 2. - set size to 2.5 * GamePanel.tileSize. - set movement images, attack images.

2.3.4.1 Methods

+ Rectangle getBounds()	return monster's rectangular bound with monster's x, y coordinate and the size of 2 * GamePanel.tileSize by 2 * GamePanel.tileSize.

2.3.5 Class MonsterFields

This class contains an arraylist of monsters that are in the game. It's the class that manages the number of monsters.

2.3.5.1 Fields

-ArrayList<MonsterBase> monsters	contains the monsters that are in the game.
-Player player	MonsterField's player
-int monsterCount	counts the number of monsters in the arraylist.

2.3.5.2 Constructor

+MonsterField(Player player)	- initialize all parameters. - initialize monsters to new ArrayList. - set monsterCount to 0.

2.3.5.3 Methods

+void addMonsterWeak()	- add one MonsterWeak to monsters. - increase monsterCount by 1.
+void addMonsterStrong()	- add one MonsterStrong to monsters. - increase monsterCount by 1.
+void addMonsterBoss()	- add one MonsterBoss to monsters. - increase monsterCount by 1.
+void updatePosition(MonsterBase monster)	if the monster is attacking, let it performs an attack first. if not, update its position. by updating its position, calculate dx and dy which are the difference of x, y coordinate between the monster and the player. if dx is more than dy, emphasize moving horizontally. if not, emphasize moving vertically. this method also manages the collision between each monster by checking if their bounds intersect.
+void updateSolidArea(MonsterBase monster)	set a new solid area of the monster every time it moves.
+void moveMonsterAwayFromOtherMonster (MonsterBase monster, MonsterBase otherMonster)	move monster away from other monster by its speed.
+void setSprites(MonsterBase monster)	set sprite number and sprite counter.
+getters and setters of all fields	

3. Package tiles

3.1) Class Tile

3.1.1 Fields

-Image image	Image of the tile
-boolean collision	collision status of the tile set to false

3.1.2 Methods

generate getters and setters for all fields	

3.2) Class TileManager

3.2.1 Fields

-Tile[] tile	an array of all tiles used in the game
-int mapTileNum[][]	a map of the tiles

3.2.2 Constructor

+TileManager()	Constructor for TileManager - initialize tile array - get the tile images

3.2.3 Methods

+void getTileImage()	- load each tile in to the tile array - set collision of certain tiles to true
+void draw(GraphicContext gc)	draw the tiles on each position on the game screen according mapTileNum
generate getters of all fields	

4. Package utils

4.1) Class CollisionChecker

4.1.1 Fields

-Gamepanel gp	the game panel

4.1.2 Constructor

+CollisionChecker(GamePanel gp)	Constructor for CollisionChecker initialize game panel

4.1.3 Methods

+void checkTile(Entity entity)	check if the entity can walk past the tile or not

4.2) Class GetData

4.2.1 Methods

+Image getImage(String source)	return the image with the path source
+Font getFont(String source, double size)	return font with the path source and the size

5. Package logic

5.1) Class GameLogic

5.1.1 Fields

-MonsterField monsterField	handles all monsters' actions
-Player player	handles the player
+final int[] currentMonsters	an array (the size is 1) number of monsters spawned
-int currentWave	the current wave of monsters
-boolean gameOver	check if the game is over
-boolean winMessage	check if the game is won

5.1.2 Constructor

+GameLogic(Player player)	Constructor for GameLogic - initialize player - set currentWave to 1

5.1.3 Methods

+void setUpGame()	a method to set up the game

	- call initializeMonster(currentWave) - set gameOver and winMessage to false
+void initializeMonster(int wave)	spawn monsters of each wave one by one
+void update()	update the player - call updatePosition() and updateAttackArea() from Player check if the game is won - call checkWin()
+int getNumMonster(int wave)	a method to get the total number of monsters spawned at the end of each wave - if wave is 1, the number is 10 - if wave is 2, another 15 monsters are spawned, making the number 25 - if wave is 3, another 2 monsters are spawned, making the number 27 return the number
+void checkWin()	check if the player has defeated the wave and check if the game is won - if a monster is dead, remove it from the monsters arraylist - if all monsters of the current wave are defeated, move on to the next wave - if the current wave is wave 3, set winMessage to true
generate getters and setters of all fields	

6. Package input

6.1) Class KeyHandler

6.1.1 Fields

-boolean up, down, left, right, spacePressed	check if certain keys are pressed
GamePanel gp	the game panel

6.1.2 Constructor

+KeyHandler(GamePanel gp)	Constructor for KeyHandler

	initialize game panel

6.1.3 Methods

+void keyPressed(KeyEvent e)	handle what to do when certain keys are pressed, the actions can be different depending on gameOver and winMessage - if the game is over or won, then W and S is for moving up and down the options, ENTER is for selecting the option - if the game is ongoing, then W, S, A, D is for moving up, down, left, and right respectively. SPACE is for attacking
+void keyReleased(KeyEvent e)	handle what to do when certain keys are released - stop the action when the key is released
generate getters and setters of all fields	

7. Package panes

7.1) Class GamePanel extends Canvas

7.1.1 Fields

+final int tileSize	Tile size on the game equals 48
+final int maxScreenCol	Column of tiles for the game window equals 16
+final int maxScreenRow	Row of tiles for the game window equals 12
+final int ScreenWidth	Width of the game window equals 768
+final int ScreenHeight	Height of the game window equals 526
-final int FPS	Frame per second equals 60
-AnimationTimer gameLoop	an AnimationTimer to loop and draw the

	game 60 times per second
-TileManager tileM	Tile Manager for managing each tiles
-KeyHandler keyH	handles each input from the player
-GameLogic gameLogic	handles game logic
-Player player	handles the player
-CollisionChecker Cchecker	a checker to check tile collision
-UI ui	handles the drawing of the game at different status
-MediaPlayer sound	background music

7.1.2 Constructor

+Gamepanel(double width, double height)	Constructor for GamePanel Set up the Game panel - set width and height - setFocusTraversable to True - set on KeyPressed and KeyReleased - play the background music - set the game logic - initialize ui

7.1.3 Methods

+void startGameThread()	a method to keep the game running at 60 FPS using AnimationTimer - set up the game - each time, call paintComponent() and gameLogic.update()
+void paintComponent()	a method to draw everything in the game. Including tiles, player, monsters, and messages
generate getters for gameLogic, tileM, Cchecker, ui, and sound. generate setter for gameLogic	

7.2) Class HomePane extends VBox

7.2.1 Fields

-GamePanel gamePanel	the game panel

7.2.2 Constructor

+HomePane()	Constructor for HomePane - set the game panel - set the background size - set the background image - add startGameButton to the Pane - add quitButton to the Pane

7.2.3 Methods

+Button startGameButton()	a Button to start the game add gamePanel to this pane's children and call startGameThread from GamePanel when clicked
+Button quitButton()	a Button to quit the game
generate Getter and Setter for gamePanel	

8. Package interface

8.1) Interface IRenderable

8.1.1 Methods

void draw(GraphicContext gc)	draw renderable