Java Project Report

Ming and Plakim's
Mole Whacker

Submitted by

Budsakorn Khosagrid 6031035821
Prawsang Chayakulkeeree 6031038721

Course

Programming Methodology 2110215

Prawsang Chayakulkeeree 6031038721
Budsakorn Khosagrid 6031035821

Introduction

This game is inspired by the mole whacker arcade game everyone is familiar with. The playing is very simple: whack the moles as quickly as possible. The game is over when all of the holes are filled. To make this game more exciting, we have added power ups, such as bombs and score multipliers, and more other things. The git repository of this game is available at https://github.com/prawsang/ProgMethMoleWhacker.git

Gameplay

At the beginning of the game, you will be greeted by the welcome screen below (Figure 1). Click “Start Game” to start playing and “Exit” to quit the game. To reset your high score back to 0, click “Reset High Score”. Once clicking the button, an alert will pop up for you to confirm the reset. If you choose to cancel on that alert, your high score will not be reset.

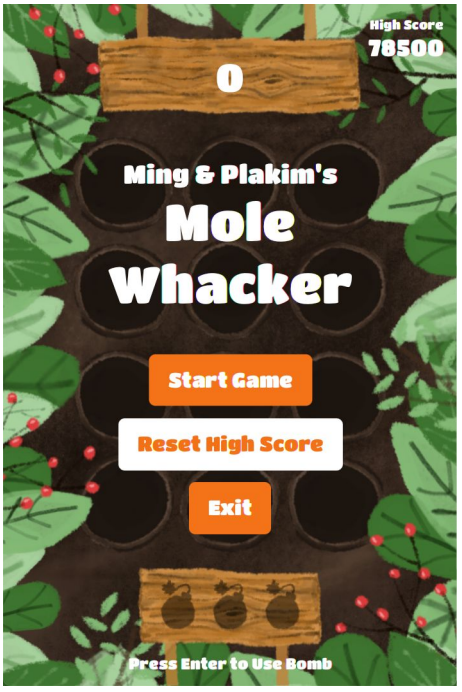

Figure 1: Welcome Screen

After clicking “Start Game”, the game will start. Play the game by clicking at the moles to whack them. As the game progresses, the mole will appear more frequent, and there will be bombs and other items appearing. The items and their meanings are listed on the next page.

Moles

Normal Mole

This type of mole disappears with one hit (one click). This is the first type of item that will appear in the game. Score is added by 100 when you hit this type of mole.

Strong Mole

This type of mole disappears with 3 hits (three clicks). Score is added by 100 when you make this type of mole disappear.

Figure 2.1: Screenshot of gameplay

Power Ups

Bomb

This can be collected in the game for future use. Once you click on it, the bomb will be collected. The pane at the bottom of the screen will show you how many bombs have been collected (Figure 2.2). To use the bomb, press “Enter” on your keyboard. The bomb will clear every hole in the game and will show effects (Figure 2.3). Only three bombs can be stored.

Figure 2.2: The pane that shows how many bombs have been collected

Figure 2.3: Bomb effect

Fever Star

Clicking on this will enable the “Fever” mode. During this mode, scores added to the game will be multiplied by 3, and the strong moles disappear with 1 hit instead of 3. A light vignette effect will also be shown during this mode (Figure 2.4). This mode lasts for 10 seconds.

Figure 2.4: Fever effect

Dynamite

This type of power up will clear out the items in the adjacent holes as shown in Figure 2.5. If there are moles in the adjacent holes, the score will be added. However, if there are power ups, the score will not be added, and the power up will disappear without being used.

Figure 2.5: Using the dynamite

Streak Star

Clicking on this will enable the “Streak” mode. During this mode, players can repeated press spacebar for extra scores while normally playing the game. The count of how many times the spacebar is pressed will be shown on the screen (Figure 2.6). This mode lasts for 5 seconds, and at the end of this mode, score will be added by the number of times the spacebar is pressed times 100.

Figure 2.6: Game during streak mode

Ending the Game

The game ends when all of the holes are filled. Once the game is over and you get a high score, it will be recorded and shown on the top right of the screen. You can either choose to play the game again or exit the game.

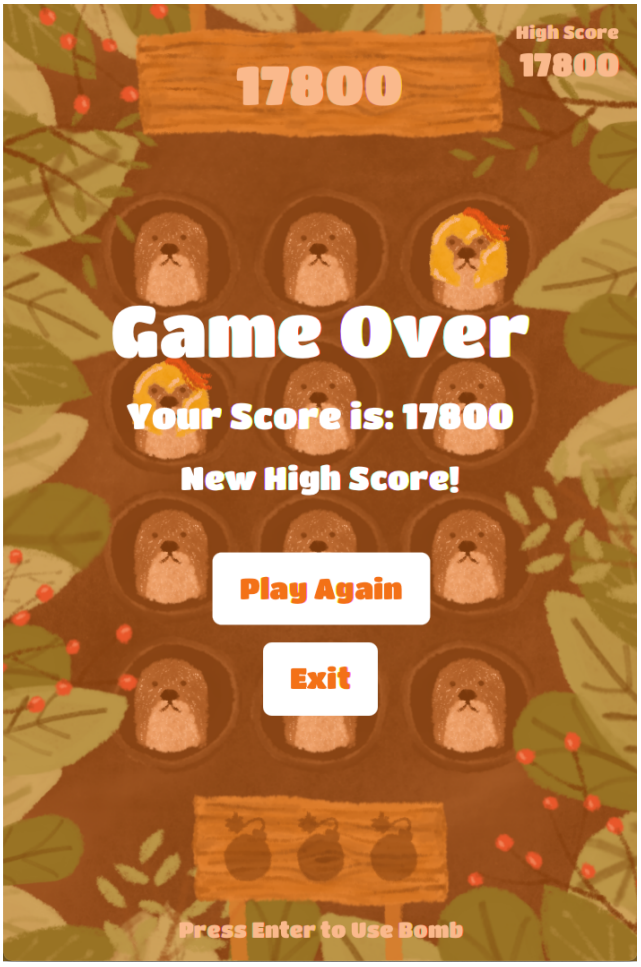

Figure 2.6: Game over screen

Prawsang Chayakulkeeree 6031038721
Budsakorn Khosagrid 6031035821

Implementation Details

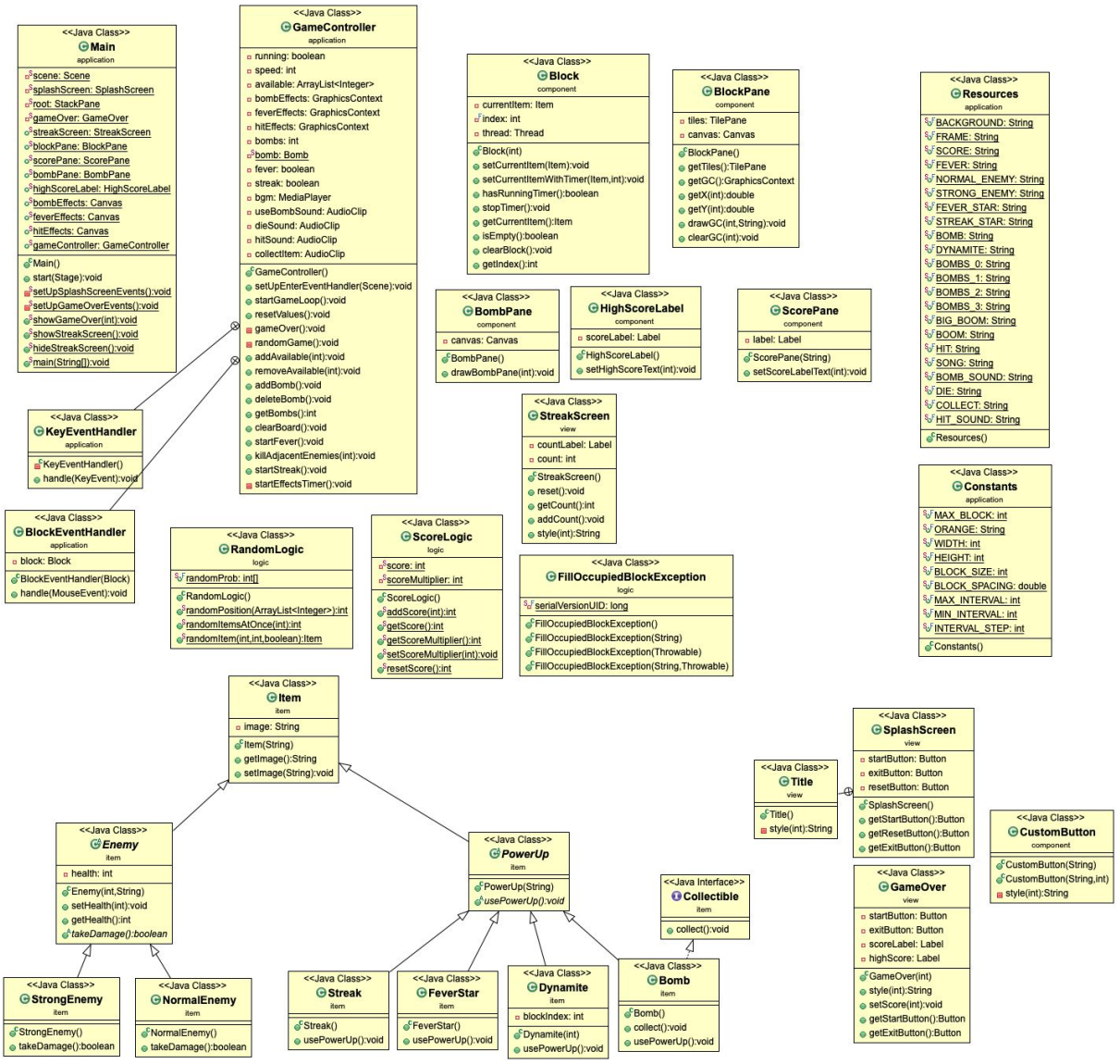

Figure 3: UML Diagram of Project

Class Details

*Noted that Access Modifier Notations can be listed below

+ (public)
# (protected)
- (private)
static will be underlined.
abstract will be italic.

Prawsang Chayakulkeeree 6031038721
Budsakorn Khosagrid 6031035821

1. Package application

1.1 Class `Constants`

1.1.1 Fields

`+ final int MAX_BLOCK`	Number of blocks (holes) in the game
`+ final String ORANGE`	RGB color code of orange color used in the game
`+ final int WIDTH`	Width of the window
`+ final int HEIGHT`	Height of the window
`+ final int BLOCK_SIZE`	Size of the block
`+ final double BLOCK_SPACING`	Spacing between blocks
`+ final int MAX_INTERVAL`	Maximum interval between each random (milliseconds)
`+ final int MIN_INTERVAL`	Minimum interval between each random (milliseconds)
`+ final int INTERVAL_STEP`	Amount of decrement of interval between each random as the game progresses (milliseconds)

1.2 Class `GameController`

1.2.1 Fields

`- boolean running`	Tells whether the game is running
`- int speed`	The interval between each random
`- ArrayList<Integer> available`	List of indices of available blocks
`- Thread randomThread`	Thread to random the items for the blocks
`- GraphicsContext bombEffects`	Graphics context for bombs effects
`- GraphicsContext feverEffects`	Graphics context for fever effects

- GraphicsContext hitEffects	Graphic contexts for hit effects
- int bombs	Amount of bombs collected
- Bomb bomb	A bomb in the game
- boolean fever	Tells whether the game is in fever mode
- boolean streak	Tells whether the game is in streak mode
- MediaPlayer bgm	Media player for background music
- AudioClip useBombSound	Audio clip for bomb sound
- AudioClip dieSound	Audio clip that plays when a mole dies
- AudioClip hitSound	Audio clip that plays when a mole is hit
- AudioClip collectItem	Audio clip that plays when a power up is collected

1.2.2 Constructor

+ GameController()	Initializes fields, adds blocks to BlockPane.

1.2.3 Methods

+ void setUpKeyHandler(Scene scene)	Sets up event handler for key events.
+ void startGameLoop()	Starts randomThread, thread for increasing speed (decreasing interval between each random), and play background music.
+ void resetValues()	Clears all blocks, graphic contexts, resets speed, score, amount of bombs, and other fields to their initial values.
- void gameOver()	Called when the game is over. Calls resetValues(), stops the background music, and shows the GameOver view.

`- void randomGame()`	Calls functions from RandomLogic to randomly choose a block to put an item in, random item type, and random how many blocks will be filled in one call.
`+ void addAvailable(int index)`	Adds index of a block to a list of indices of available blocks
`+ void removeAvailable(int index)`	Removes index of a block from a list of indices of available blocks
`+ void addBomb()`	Adds 1 to the current amount of bombs
`+ void deleteBomb()`	Minus 1 from the current amount of bombs
`+ int getBombs()`	Returns the current amount of bombs
`+ void clearBoard()`	Clears the board, and adds score by the number of enemies(moles) killed.
`+ void startFever()`	Starts fever mode. Sets fever to true, and starts a thread that will set fever back to false.
`+ void killAdjacentEnemies(int position)`	This function is called when a Dynamite is clicked. It kills enemies(moles) adjacent to the block of the specified position. Draws images to bombEffects.
`+ void startStreak()`	Starts streak mode. Sets streak to true, shows StreakScreen and plays collectItem sound. Then, it starts a thread that sets streak back to false, hides StreakScreen, and adds score according to the count of StreakScreen after 5 seconds.
`- void startEffectsTimer()`	Starts a thread that clears bombEffects 0.5 seconds after drawing to it.

1.3 Class `EnterEventHandler implements EventHandler<KeyEvent>` (Inner class of `GameController`)

1.3.1 Methods

`+ void handle(KeyEvent e)`	Handles key events from enter and space bar. When the enter key is pressed, a bomb is used (if available), bomb sound is played, and bomb effects are drawn to bombEffects. Calls startEffectsTimer() to remove the effects 0.5 seconds afterwards. The enter key does nothing when the game is in streak mode (when streak is true). During the streak mode, when the spacebar is pressed, the count in StreakScreen is added.

1.4 Class `BlockEventHandler implements EventHandler<MouseEvent>` (Inner class of `GameController`)

1.4.1 Fields

`- Block block`	The block being clicked on

1.4.2 Constuctor

`+ BlockEventHandler(Block block)`	Initializes block field

1.4.3 Methods

`+ void handle(MouseEvent arg0)`	If the block is empty, return and do nothing. If the block contains an Enemy (mole), it calls takeDamage() from the Enemy, and plays sounds. If takeDamage of the Enemy returns false, it clears the block. If the block

	contains a PowerUp, it calls usePowerUp() of that power up, but if the PowerUp is a Collectible, it calls collect() of that PowerUp. For Bomb, if there are already 3 bombs collected, it will do nothing. For Streak, if streak is true, it will do nothing. For FeverStar, if fever is true, it will do nothing. It also stops any timer/thread running from the block.

1.5 Class `Main`

1.5.1 Fields

`- Scene scene`	The main scene of the application
`- SplashScreen splashScreen`	Welcome screen
`- StackPane root`	Root pane of the application
`- GameOver gameOver`	Game over screen
`+ StreakScreen streakScreen`	Streak screen
`+ BlockPane blockPane`	A pane containing all blocks
`+ ScorePane scorePane`	A pane that displays the score
`+ BombPane bombPane`	A pane that displays the amount of bombs collected
`+ HighScoreLabel highScoreLabel`	A pane that displays the high score
`+ Canvas bombEffects`	Canvas for effects from bombs
`+ Canvas feverEffects`	Canvas for effects during fever mode
`+ Canvas hitEffects`	Canvas for effects when an Enemy is hit
`+ GameController gameController`	The game controller

1.5.2 Methods

`+ void start(stage primaryStage)`	Initializes fields, creates components to add to root in appropriate order, and draws background images to a canvas, then add to root. Sets up the game controller, and calls setUpSplashScreenEvents() and setUpGameOverEvents(). Shows SplashScreen (Welcome screen).
`- void setUpSplashScreenEvents()`	Sets up the events of the buttons in SplashScreen. The start button starts the game, the reset high score button resets the high score, and the exit button exits the game.
`- void setUpGameOverEvents()`	Sets up the events of the buttons in GameOver. The play again button resets the values in the game controller, and starts the game again. The exit button exits the game.
`+ void showGameOver(int score)`	Shows the game over screen.
`+ void showStreakScreen()`	Shows the streak view
`+ void hideStreakScreen()`	Hides the streak view
`+ void main(String [] args)`	Starts the application.

1.6 Class `Resources`

1.6.1 Fields

`+ final String BACKGROUND`	Background image.
`+ final String FRAME`	Image for the frame (leaves)
`+ final String SCORE`	Image for the score pane
`+ final String FEVER`	Image for fever effects
`+ final String NORMAL_ENEMY`	Image for NormalEnemy
`+ final String STRONG_ENEMY`	Image for StrongEnemy

+ final String FEVER_STAR	Image for FeverStar power up
+ final String STREAK_STAR	Image for StreakStar power up
+ final String BOMB	Image for Bomb power up
+ final String DYNAMITE	Image for Dynamite power up
+ final String BOMBS_0	Image for BombPane when 0 bombs are collected
+ final String BOMBS_1	Image for BombPane when 1 bomb is collected
+ final String BOMBS_2	Image for BombPane when 2 bombs are collected
+ final String BOMBS_3	Image for BombPane when 3 bombs are collected
+ final String BIG_BOOM	Image for when a Bomb is used
+ final String BOOM	Image for when a Dynamite is used
+ final String HIT	Image for when an Enemy is hit
+ final String SONG	Background music
+ final String BOMB_SOUND	Bomb sound
+ final String DIE	Sound when the Enemy dies
+ final String COLLECT	Sound when a Bomb and other PowerUp's (except Dynamite) is collected.
+ final String HIT_SOUND	Sound when the Enemy is hit.

2. Package component

2.1 Class `Block extends StackPane`

2.1.1 Fields

`- Item currentItem`	The item currently in this block
`- final int index`	The index of this block
`- Ihread thread`	The thread for when a power up is added to this block, and it has to disappear in 3 seconds.

2.1.2 Constructor

`+ Block(int index)`	Initializes index. Sets width and height of the block.

2.1.3 Methods

`+ void setCurrentItem(Item i) throws FillOccupiedBlockException`	Sets currentItem of this block and adds the item to this block. Throws FillOccupiedBlockException if the block is already occupied.
`+ void set CurrentItemWithTimer(Item i,int duration) throws FillOccupiedBlockException`	Sets currentItem of this block, adds the item to this block, and starts a thread that will make the item disappear and clear the block in 3 seconds. Throws FillOccupiedBlockException if the block is already occupied.
`+ boolean hasRunningTimer()`	If thread is running, return true. Else, return false.
`+ void stopTimer()`	Interrupt the thread.
`+ Item getCurrentItem()`	Returns currentItem
`+ boolean isEmpty()`	If the block does not have an item in it, return true. Else, return false.
`+ void clearBlock()`	Clears this block. currentItem is

	removed from the block.
`+ int getIndex()`	Returns the index of this block

2.2 Class `BlockPane extends StackPane`

2.2.1 Fields

`- TilePane tiles`	Pane to contain all the blocks
`- Canvas canvas`	Canvas to draw items in the position of the blocks.

2.2.2 Constructor

`+ BlockPane()`	Initializes tiles and canvas, and sets the properties of tiles so that it displays correctly. Adds tiles and canvas to itself.

2.2.3 Methods

`+ TilePane getTiles()`	Gets the TilePane of itself
`+ GraphicsContext getGC()`	Gets the graphics context of the canvas
`+ double getX(int index)`	Gets X position of the block relative to itself according to the specified index.
`+ double getY(int index)`	Gets Y position of the block relative to itself according to the specified index.
`+ void drawGC(int index,String imagePath)`	Draws image to its canvas at imagePath to the position of the block of the specified index.
`+ void clearGC(int index)`	Clears its canvas at the position of the block of the specified index.

2.3 Class `BombPane extends StackPane`

2.3.1 Fields

`- Canvas canvas`	Its canvas

2.3.2 Constructor

`+ BombPane()`	Initializes the canvas, calls drawBombPane(0), and adds “Press Enter to Use Bomb” label to itself. Adds the canvas to itself.

2.3.3 Methods

`+ void drawBombPane(int bombs)`	Draws bomb pane image to its canvas according to the number of bombs

2.4 Class `CustomButton extends Button`

2.4.1 Constructor

`+ CustomButton(String text)`	Sets the styling of the button and sets the text of the button according to the specified text.
`+ CustomButton(String text,int color)`	Sets the styling of the button and sets the text of the button according to the specified text. If color is 0, the background is Constants.ORANGE and text color is white. If color is 1, the background is white and the text color is Constants.ORANGE.

2.4.2 Methods

`- String style(int color)`	A method that returns the style of the button, according to the specified color option.

2.5 Class `HighScoreLabel extends VBox`

2.5.1 Fields

`- Label scoreLabel`	Label to show the high score

2.5.2 Constructor

`+ HighScoreLabel()`	Adds 2 labels to itself: 1. The label that says “High score:” and 2. The label that shows the high score.

2.5.3 Methods

`+ void setHighScoreText(int i)`	Sets the text of scoreLabel

2.6 Class `ScorePane extends StackPane`

2.6.1 Fields

`- Label label`	Label to show the score

2.6.2 Constructor

`+ ScorePane(String text)`	Initializes the label, and adds the label and a canvas with image of the score pane to itself.

2.6.3 Methods

`+ void setScoreLabelText(int score)`	Sets the text of the label

3. Package item

3.1 Class `Bomb extends PowerUp implements Collectible`

3.1.1 Constructor

`+ Bomb()`	Initilizes Item with image of Resources.BOMB

3.1.2 Methods

`+ void collect()`	Calls addBomb() method from the game controller
`+ void usePowerUp()`	Calls clearBoard() method from the game controller

3.2 Interface `Collectible`

3.2.1 Methods

`+ void collect()`	Calls when the item is collected

3.3 Class `Dynamite extends PowerUp`

3.3.1 Fields

`- int blockIndex`	The index of the block that this Dynamite is in

3.3.2 Constructor

`+ Dynamite(int index)`	Initilizes Item with image of Resources.DYNAMITE and initializes blockIndex according to the provided index

3.3.3 Methods

`+ void usePowerUp()`	Calls killAdjacentEnemies(this.blockIndex) from the game controller

3.4 Class `Enemy extends Item`

3.4.1 Fields

`- int health`	The health of this Enemy

3.4.2 Constructor

`+ Enemy(int health,String image)`	Initilizes health according to the provided health, and initializes Item with the provided image

3.4.3 Methods

`+ void setHealth(int health)`	Sets the health of this Enemy
`+int getHealth()`	Returns the health of this Enemy
`+ boolean takeDamage()`	Calls when the enemy takes damage

3.5 Class `FeverStar extends PowerUp`

3.5.1 Constructor

`+ FeverStar()`	Initilizes Item with image of Resources.FEVER_STAR

3.5.2 Method

`+ void usePowerUp()`	Calls startFever() method from the game controller

3.6 Class `Item extends StackPane`

3.6.1 Fields

`- String image`	Image path for this item

3.6.2 Constructor

`+ Item(String image)`	Initilizes image, sets width and height, and sets mouse transparent to true.

3.6.3 Methods

`+ String getImage()`	Gets the image path of this item
`+ void setImage(String image)`	Sets the image path of this item

3.7 Class `NormalEnemy extends Enemy`

3.7.1 Constructor

`+ NormalEnemy()`	Initilizes Item with image of Resources.NORMAL_ENEMY

3.7.2 Methods

`+ boolean takeDamage()`	Return false

3.8 Class *`PowerUp extends Item`*

3.7.1 Constructor

`+ PowerUp(String image)`	Initializes Item with the image of the provided image

3.7.2 Methods

+ void usePowerUp()	Calls when this PowerUp is used

3.9 Class `Streak extends PowerUp`

3.9.1 Constructor

`+ Streak()`	Initilizes Item with image of Resources.STREAK_STAR

3.9.2 Method

`+ void usePowerUp()`	Calls startStreak() from the game controller

3.10 Class `StrongEnemy extends Enemy`

3.10.1 Constructor

`+ StrongEnemy()`	Initilizes Item with image of Resources.STRONG_ENEMY and sets the health to 3.

3.10.2 Method

`+ boolean takeDamage()`	Decreases health by one. If health > 0, return true. Else, return false.

4. Package logic

4.1 Class `FillOccupiedBlockException extends Exception`

4.1.1 Fields

`- final long serialVersionUID`	The default serial version uid

4.1.2 Constructors

`+ FillOccupiedBlockException()`	Initilizes the exception
`+ FillOccupiedBlockException( String message)`	Initializes the exception according to the provided message
`+ FillOccupiedBlockException(Throwble cause)`	Initializes the exception according to the provided cause
`+ FillOccupiedBlockException(String message, Throwable cause)`	Initializes the exception according to the provided message and cause

4.2 Class `HighScoreLogic`

4.2.1 Fields

`- Path path`	Path of the file that stores the high score (highscore.txt).
`- int highScore`	The high score

4.2.2 Method

`- void getFile()`	Get the file according to path. If the file does not exist, it calls createNewFile().
`- void createNewFile()`	Creates a new file called highscore.txt

- void writeHighScore(int score)	Writes the high score into the file in path. If the score written is less than or equal to the current high score, do nothing.
- void resetHighScore()	Writes 0 to the file in path
- void readHighScore()	Reads the score from the file in path and stores in highscore
- int getHighScore()	Returns highscore

4.3 Class RandomLogic

4.3.1 Fields

+ final int[] randomProb	An array of integers to be utilized by the methods to create uneven probability for each item.

4.3.2 Method

+ int randomPosition(ArrayList<Integer> available)	Randoms a position of the block from a list of indices of available blocks provided
+ int randomItemAtOnce(int speed)	Randoms how many items will be randomed at once
+ Item randomItem(int speed, int position,boolean fever)	Randoms the type of item. At the beginning of the game (when speed (interval) is slow), only NormalEnemy appears. As the game progresses, other types of items start to appear. NormalEnemy has the greatest probability of appearing in game, followed by StrongEnemy, and each type of PowerUp has the same probability of appearing. FeverStar will not appear if the game is in fever mode.

4.4 Class `ScoreLogic`

4.4.1 Fields

`- int score`	Current score
`- int scoreMultiplier`	Current score multiplier

4.4.2 Methods

`+ int addScore(int added)`	Score is added by added * scoreMultiplier. Returns the added score.
`+ int getScore()`	Returns current score
`+ int getScoreMultiplier()`	Returns current score multiplier
`+ void setScoreMultiplier()`	Sets the score multiplier
`+ int resetScore()`	Reset score to 0. Returns the reset score.

5. Package view

5.1 Class `GameOver extends VBox`

5.1.1 Fields

`- Button startButton`	The button to restart the game
`- Button exitButton`	The button to exit the game
`- Label scoreLabel`	The label that shows the achieved score
`- Label highScore`	The label that tells the player if their achieved score is their new high score.

5.1.2 Constructor

`+ GameOver(int score)`	Initializes all fields and adds to itself. Sets the properties of itself to display correctly.

5.1.3 Methods

`+ String style(int fontSize)`	Returns the style of the label
`+ void setScore(int score)`	Set the text of scoreLabel
`+ Button getStartButton()`	Returns startButton
`+ Button getExitButton()`	Returns exitButton

5.2 Class `SplashScreen extends VBox`

5.2.1 Fields

`- Button startButton`	The button to start the game
`- Button exitButton`	The button to exit the game
`- Button resetButton`	The button to reset the high score

5.2.2 Constructor

`+ SplashScreen()`	Initializes all fields and adds to itself. Sets the properties of itself to display correctly. Initializes Title and adds to itself.

5.2.3 Methods

`+ Button getStartButton()`	Returns startButton
`+ Button getResetButton()`	Returns resetButton
`+ Button getExitButton()`	Returns exitButton

5.3 Class `Title extends VBox` (inner class of `SplashScreen`)

5.3.1 Constructor

`+ Title`	Intializes labels for displaying the title of the game. Sets properties of itself to display correctly.

5.3.2 Methods

`- String style(int fontSize)`	Returns the style for the label according to the provided font size

5.4 Class `StreakScreen extends VBox`

5.4.1 Fields

`- Label countLabel`	Label to display count
`- int count`	The count to be displayed in countLabel

5.4.2 Constructor

`+ StreakScreen()`	Initializes all fields and adds to itself. Sets the properties of itself to display correctly.

5.4.3 Methods

`+ void reset()`	Sets count to 0 and sets the text of countLabel to “0”
`+ int getCount()`	Returns the current count
`+ void addCount()`	Add to current count by 1. Sets the text of countLabel according to current count.
`+ String style(int fontSize)`	Returns the style for the label according to the provided font size