Team CookieClickerJavaEdition Documentation presents

“Cookie Clicker (Java Edition)”

created by
Kiattipoom Rojvanakarn 6130048921
Jirapat Phetlertanan 6130074121

2110215 Programming Methodology
Semester 1 Year 2019
Chulalongkorn University

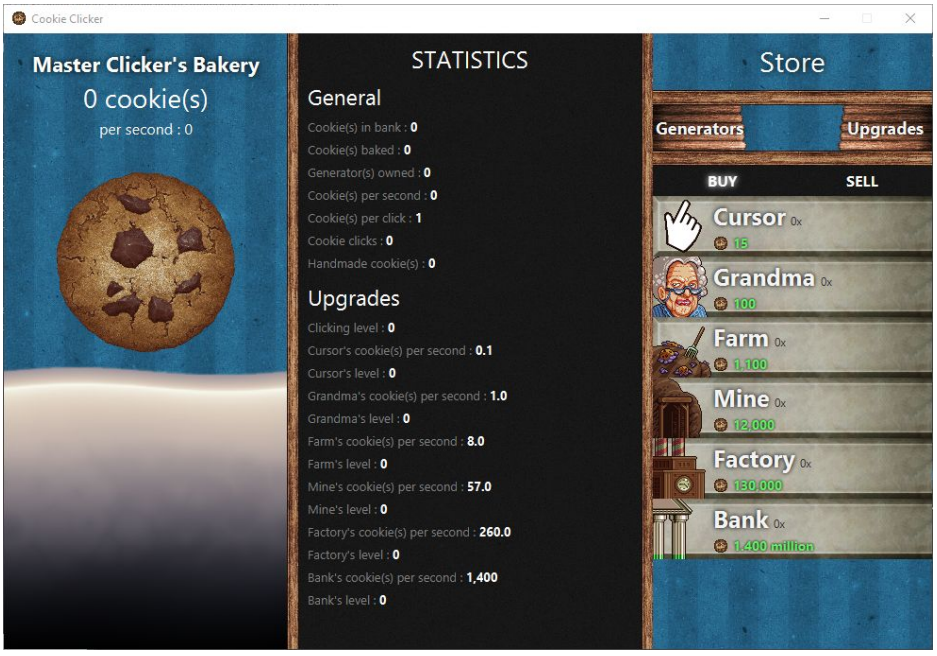
1. What is “Cookie Clicker (Java Edition)” ?

1.1 Introduction

Cookie Clicker (Java Edition) is an idle game, inspired by the game with the same name. You click the cookie on the screen, which will get one cookie per click. Then, you can spend cookies buying generators to automatically produce cookies for you. If you have enough generators, or click enough cookies, you can buy an upgrade to improve the efficiency of your clicking power and generators.

The original Cookie Clicker was coded in JavaScript. This game, which is called “Java Edition”, implements and modifies some of its existing features.

1.2 Gameplay

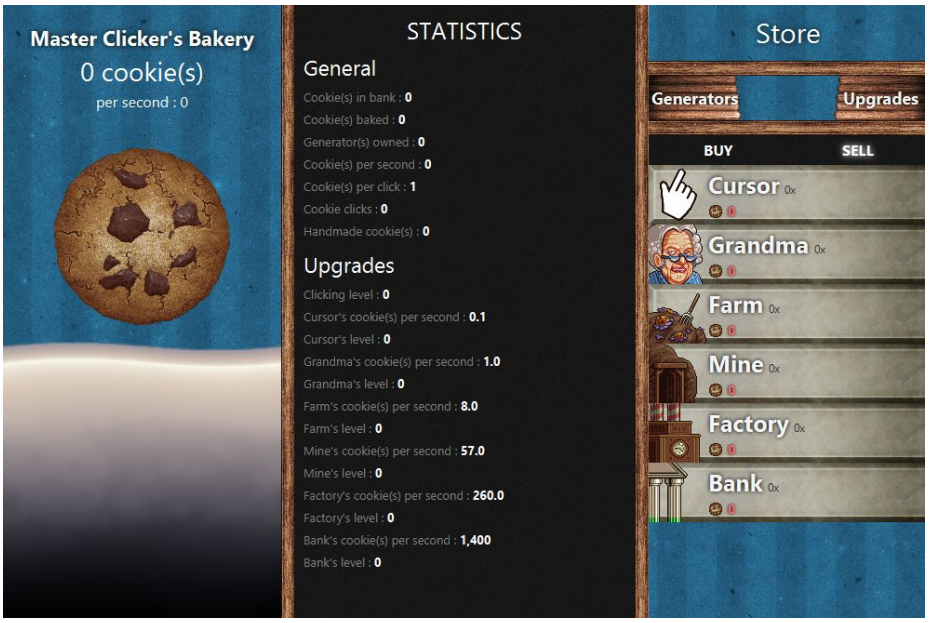


The GUI of the game.

The main game shows up when you launch the game. The GUI can be divided into three blocks from left to right : cookie block, stat block and the store block.

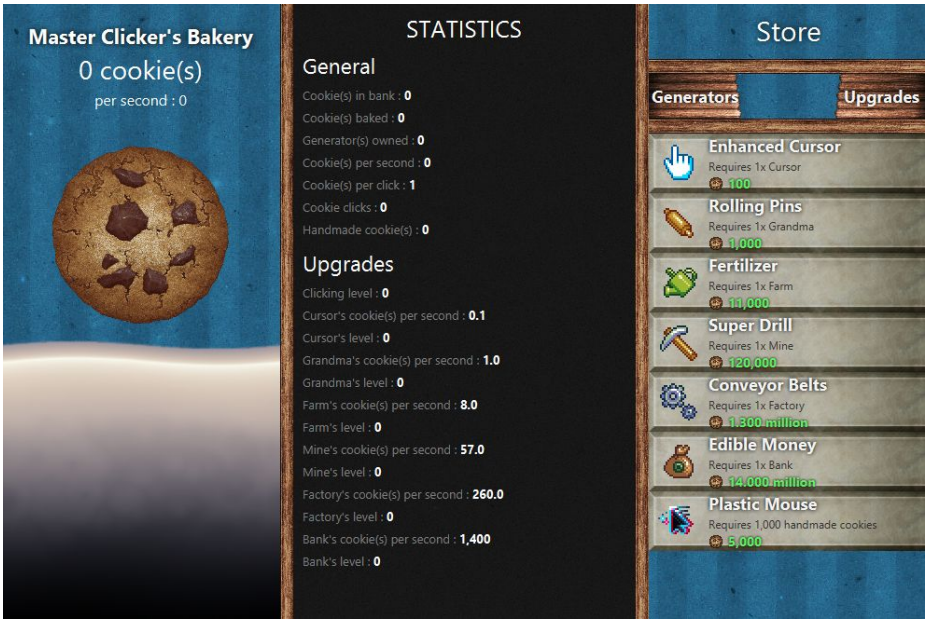
At first, the player clicks on the big cookie located in the cookie block to earn cookies (which will get one cookie per click). With cookies gained, the player can buy generators such as cursors, grandmas, farms, that automatically generate cookies from the store block. If the player has enough generators or clicks enough cookies, the player can purchase upgrades to increase cookies generated by generators or clicking. The game has no ending. The player can sell generators by clicking the

“SELL” button, which will turn the generators in the store block to sell mode. The player can revert to buy mode by clicking the “BUY” button.



The GUI after clicking the “SELL” button.

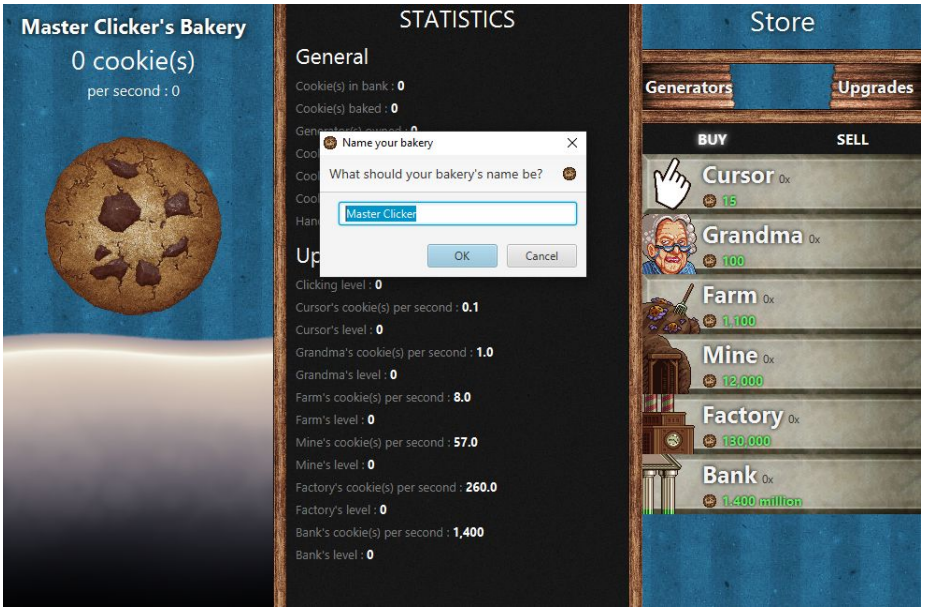
By default, the store block will show generators. If the player can simply switch to upgrades by clicking the “Upgrades” button. The player can switch back to generators by clicking the “Generators” button.



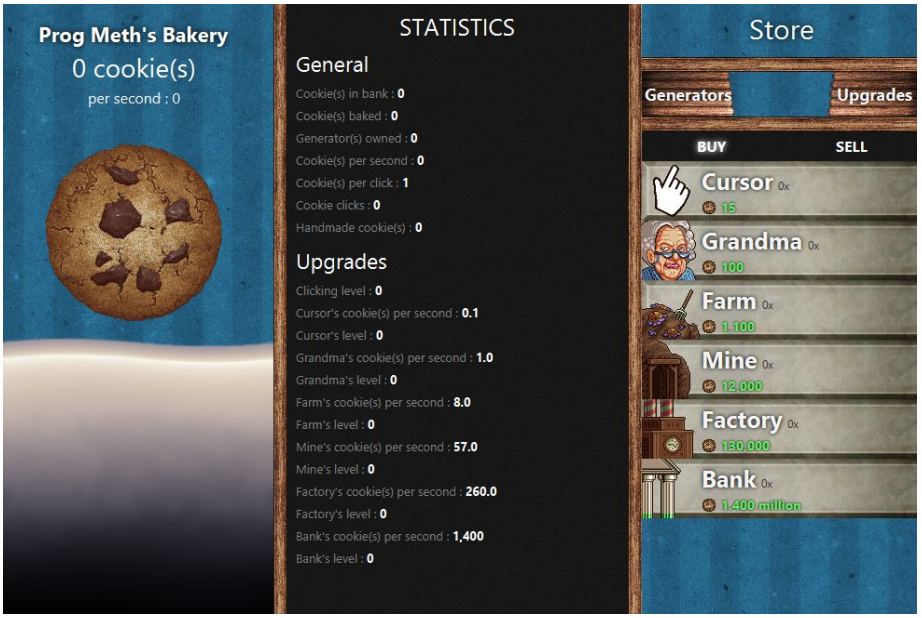
The GUI of the game after clicking the “Upgrades” button.

The stat box will update the player’s current information every second.

The player's default name will be "Master Clicker". The player can change his name by clicking the name in the cookie block, and the name changing window will appear.



Name changing dialog that appears after clicking the name in the cookie block.

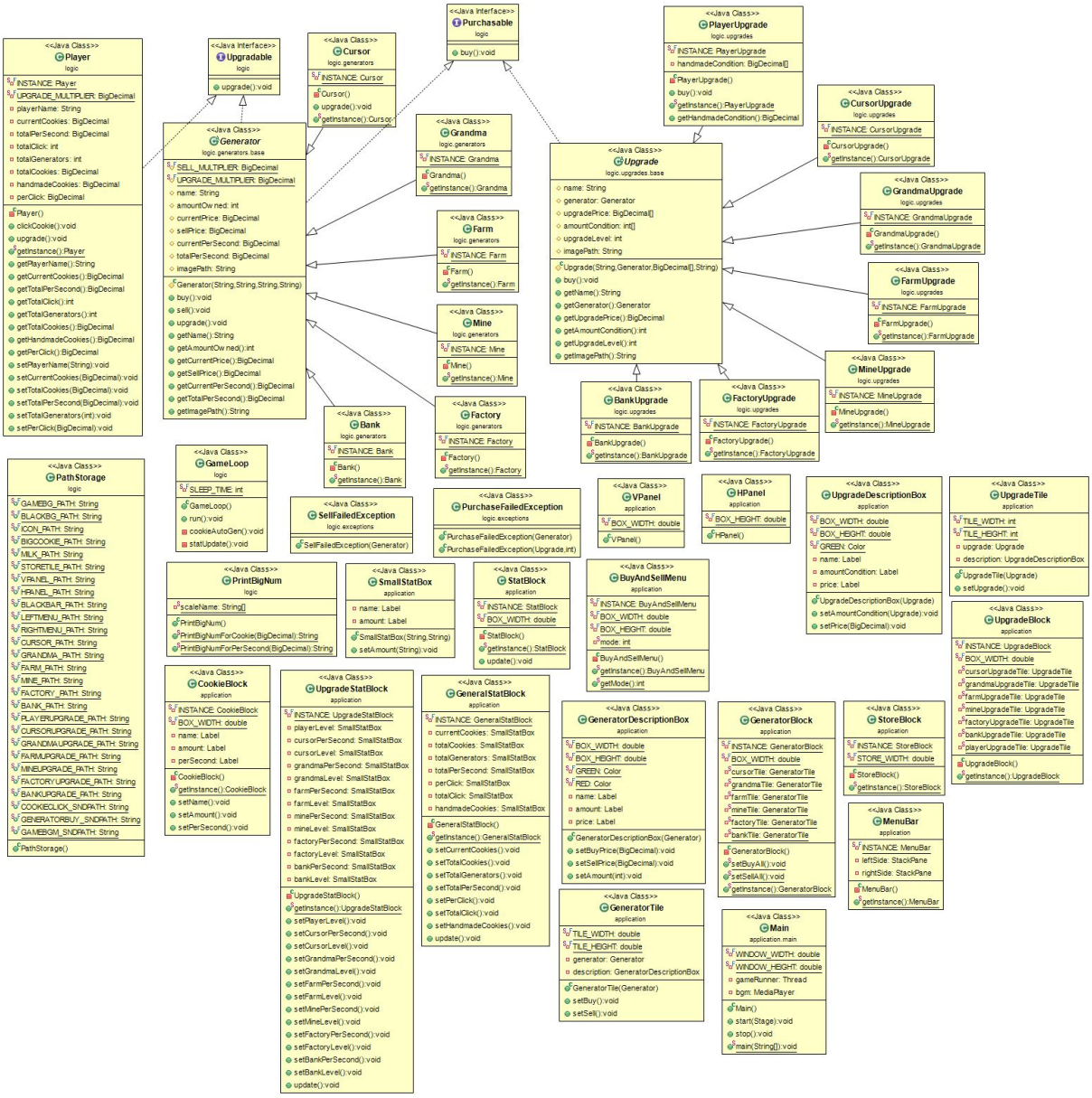


The GUI after changing the name to "Prog Meth".

1.3 Handling Overflow

Reaching large numbers (such as trillion, quadrillion, etc.) is very common in idle games. To handle large numbers in Java language, specifically in this project, we don't use primitive class such as integer or double. Instead, we use a class called "BigDecimal". *(For more information about BigDecimal class, check section 3)*

2. Implementation Details



* Noted that Access Modifier Notations can be listed below

+ (public)

(protected)

- (private)

<u>static will be underlined.</u>

abstract will be italic.

ALL_CAPITAL will be final variable.

2.1 Package `logic`

2.1.1 Interface `Purchasable`

This interface defines methods for objects that can be purchased.

2.1.1.1 Method

+ void buy() throws PurchaseFailedException	This method buys the current object.

2.1.2 Interface `Upgradable`

This interface defines methods for objects that can be upgraded.

2.1.2.1 Method

+ void upgrade()	This method upgrades the current object.

2.1.3 Class `PrintBigNum`

This class is used for converting BigDecimal numbers into String. It will be mainly used in package application, as most components need to display BigDecimal numbers.

2.1.3.1 Field

- String scaleName[] = {"million","billion","trillion","quadrillion","quintillion","sextillion","septillion"}	An array representing scale for large numbers.

2.1.3.2 Method

+ String PrintBigNumForCookie(BigDecimal num)	Return String in cookie amount format. This method will do following steps **in order** : - Gets rid of any decimal and rounds the number down, then instantiate an integer which represents to number's total digit. - If the digit is more than six, return String with this format : **x.xxx** quantifier (example : 1.000 million) - Else, print the first three digits at once with "," (If the digit is more than three), then print the last three digits from left to right, **digit by digit** (Gets rid of any decimal and rounds down for every process).

+ String PrintBigNumForPerSecond(BigDecimal num)	Returns String in cookies per second format. This method will do following steps : - If the number is equal to zero or at least one thousand, return String in the same format with String in cookie amount format. - Else if the remainder of the number is equal to zero, return String in the same format with String in cookie amount format but with extra ".0" String. - Else, return String with one decimal, rounded down.

2.1.4 Class `Player` implements `Upgradable`

This class represents the player in the game.

2.1.4.1 Field

- Player INSTANCE = new Player()	An instance of the player.
- BigDecimal UPGRADE_MULTIPLIER = new BigDecimal("4")	Variable of the upgrade multiplier.
- String playerName	Player's name.
- BigDecimal currentCookies	Player's current cookies.
- BigDecimal totalCookies	Player's total cookies generated so far.
- BigDecimal totalPerSecond	Player's current cookies per second.
- int totalGenerators	Total generators owned.
- BigDecimal perClick	Amount of cookies that can be generated by one click.
- int totalClick	Player's total clicks so far.
- BigDecimal handmadeCookies	Total cookies generated by clicking.

2.1.4.2 Constructor

- Player()	Initializes the player. The player will have following attributes : - Name : Master Clicker - Cookies per click : 1

	- Other fields will be set to 0.

2.1.4.3 Method

- void clickCookie()	This method will be called when the player clicks the cookie. It will increase player's current cookies, total cookies, and handmade cookies by the amount of cookies per click.
- void upgrade()	Multiplies perClick by amount of UPGRADE_MULTIPLIER.
+ Getter method for instance.	
+ Getter method for every other field.	
+ Setter method for every field. (except instance, totalClick and handmadeCookies)	

2.1.5 Class `GameLoop` implements `Runnable`

This class is used for setting up logic for game loop thread, which will be set up in class Main.

2.1.5.1 Field

- int SLEEP_TIME = 1000;	Sleep time of the thread.

2.1.5.2 Method

+ void run()	Runs the loop. This method will do the following steps **by order**. - Auto generates the cookies. - Updates stat block. - Last, sleep for a duration of SLEEP_TIME. (The thread will only get interrupted when the application exits.)
- void cookieAutoGen()	Auto generates cookies by adding player's totalPerSecond to currentCookies and totalCookies.
- void statUpdate()	Updates all components in the stat block. *(For more information about stat block, check section 2.5.4)*

2.1.6 Class `PathStorage`

This class is used to store image and sound paths.

2.1.6.1 Field

+ String GAMEBG_PATH = ClassLoader.getSystemResource("img/bg.png").toString()	The path to image “bg.png”.
+ String BLACKBG_PATH = ClassLoader.getSystemResource("img/blackbg.png").toString()	The path to image “blackbg.png”.
+ String ICON_PATH = ClassLoader.getSystemResource("img/point.png").toString()	The path to image “point.png”.
+ String BIGCOOKIE_PATH = ClassLoader.getSystemResource("img/cookie.png").toString()	The path to image “cookie.png”.
+ String MILK_PATH = ClassLoader.getSystemResource("img/milkPlain.png").toString()	The path to image “milkPlain.png”.
+ String STORETILE_PATH = ClassLoader.getSystemResource("img/storeTile.png").toString()	The path to image “storeTile.png”.
+ String VPANEL_PATH = ClassLoader.getSystemResource("img/panelVertical.png").toString()	The path to image “panelVertical.png”.
+ String HPANEL_PATH = ClassLoader.getSystemResource("img/panelHorizontal.png").toString()	The path to image “panelHorizontal.png”.
+ String BLACKBAR_PATH = ClassLoader.getSystemResource("img/blackBar.png").toString()	The path to image “blackBar.png”.
+ String LEFTMENU_PATH = ClassLoader.getSystemResource("img/leftMenu.png").toString()	The path to image “leftMenu.png”.
+ String RIGHTMENU_PATH = ClassLoader.getSystemResource("img/rightMenu.png").toString()	The path to image “rightMenu.png”.
+ String CURSOR_PATH = ClassLoader.getSystemResource("img/gen	The path to generator image “cursor.png”.

erators/cursor.png").toString()	
+ String GRANDMA_PATH = ClassLoader.getSystemResource("img/generators/grandma.png").toString()	The path to generator image “grandma.png”.
+ String FARM_PATH = ClassLoader.getSystemResource("img/generators/farm.png").toString()	The path to generator image “farm.png”.
+ String MINE_PATH = ClassLoader.getSystemResource("img/generators/mine.png").toString()	The path to generator image “mine.png”.
+ String FACTORY_PATH = ClassLoader.getSystemResource("img/generators/factory.png").toString()	The path to generator image “factory.png”.
+ String BANK_PATH = ClassLoader.getSystemResource("img/generators/bank.png").toString()	The path to generator image “bank.png”.
+ String PLAYERUPGRADE_PATH = ClassLoader.getSystemResource("img/upgrades/player.png").toString()	The path to upgrade image “player.png”.
+ String CURSORUPGRADE_PATH = ClassLoader.getSystemResource("img/upgrades/cursor.png").toString()	The path to upgrade image “cursor.png”.
+ String GRANDMAUPGRADE_PATH = ClassLoader.getSystemResource("img/upgrades/grandma.png").toString()	The path to upgrade image “grandma.png”.
+ String FARMUPGRADE_PATH = ClassLoader.getSystemResource("img/upgrades/farm.png").toString()	The path to upgrade image “farm.png”.
+ String MINEUPGRADE_PATH = ClassLoader.getSystemResource("img/upgrades/mine.png").toString()	The path to upgrade image “mine.png”.
+ String FACTORYUPGRADE_PATH = ClassLoader.getSystemResource("img/upgrades/factory.png").toString()	The path to upgrade image “factory.png”.
+ String BANKUPGRADE_PATH = ClassLoader.getSystemResource("img/upgrades/bank.png").toString()	The path to upgrade image “bank.png”.
+ String COOKIECLICK_SNDPATH = ClassLoader.getSystemResource("snd/coo	The path to sound “cookieClick.mp3”.

kieClick.mp3").toString()	
+ String GENERATORBUY_SNDPATH = ClassLoader.getSystemResource("snd/generatorBuy.mp3").toString()	The path to sound “generatorBuy.mp3”.
+ String GAMEBGM_SNDPATH = ClassLoader.getSystemResource("snd/bgm.mp3").toString()	The path to sound “bgm.mp3”.

2.2 Package `logic.generators`

2.2.1 Package `logic.generators.base`

2.2.1.1 *Class* `Generator` *implements* `Purchasable, Upgradable`

This class represents a generator in the game. Generators are objects that increase cookies per second. Each generator has its own CpS (cookies per second) and the price increases by 15% each time you buy one (the price is always rounded down). When selling a generator, you are refunded by 25% of the current generator price.

2.2.1.1.1 Field

# BigDecimal BUY_MULTIPLIER = new BigDecimal("1.15")	Variable of the buy multiplier.
# BigDecimal SELL_MULTIPLIER = new BigDecimal("0.25")	Variable of the sell multiplier.
# BigDecimal UPGRADE_MULTIPLIER = new BigDecimal("2")	Variable of the upgrade multiplier.
# String name	Generator’s name.
# int amountOwned	Total amount of this type of generator that player owned.
# BigDecimal currentPrice	Current price of this generator.
# BigDecimal sellPrice	Current sell price of this generator.
# BigDecimal currentPerSecond	Current cookies per second that this type of generator can generate.
# BigDecimal totalPerSecond	Total cookies per second that this type of generator can generate.
# String imagePath	The image path of this generator.

2.2.1.1.2 Constructor

# Generator(String name, String price, String perSecond, String imagePath)	Initializes the generator. Unmentioned fields will be set to 0.

2.2.1.1.3 Method

+ void buy() throws PurchaseFailedException	This method buys the generator. It should : - Subtracts the player's current cookie by the generator's current buy price. - Add the player's cookies per second and the generator's total cookies per second by the generator's current cookies per second. - Updates the player's and the generator's total amount. - Updates current buy price by multiplying by the amount of BUY_MULTIPLIER (the result is always rounded down). - Updates current sell price by multiplying buy price by the amount of SELL_MULTIPLIER (the result is always rounded down). If the current cookies are less than the required cookies, the exception called "PurchaseFailedException" will be thrown.
+ void sell() throws SellFailedException	This method sells the generator. It should : - Adds the player's current cookie by the generator's current buy price. - Subtract the player's cookies per second and the generator's total cookies per second by the generator's current cookies per second. - Updates the player's and the generator's total amount. - Updates current buy price by dividing by the amount of BUY_MULTIPLIER (the result is always rounded up). - Updates current sell price by multiplying buy price by the amount of SELL_MULTIPLIER (the result is always rounded down).

	If player has no generator of that type, an exception called “SellFailedException” will be thrown.
+ void upgrade()	Multiplies the generator’s current cookies per second by amount of UPGRADE_MULTIPLIER, then updates the generator’s total cookies per second.
+ Getter method for every field	

2.2.2 Class `Cursor` extends `Generator`

“Autoclicks once every 10 seconds. Such wow 100%”

The **Cursor** is the first generator in the game. It has starting price of 15 cookies and will automatically generate a cookie every 10 seconds.

2.2.2.1 Field

- Cursor INSTANCE = new Cursor()	An instance of the cursor.

2.2.2.2 Constructor

- Cursor()	Initializes the cursor. It will have the image path of the generator image “cursor.png”.

2.2.2.3 Method

+ void upgrade()	*This method overrides its parent.* Behaves like a normal generator’s upgrade method, but also multiplies player’s perClick by amount of generator’s UPGRADE_MULTIPLIER.
+ Getter method for instance.	

2.2.3 Class `Grandma` extends `Generator`

“A nice grandma to bake more cookies. Yummy !”

The **Grandma** is the second generator in the game. She has starting price of 100 cookies and will automatically generate 1 cookie every second.

2.2.3.1 Field

- Grandma INSTANCE = new Grandma()	An instance of the grandma.

2.2.3.1 Constructor

+ Grandma()	Initializes the grandma. It will have the image path of the generator image “grandma.png”.

2.2.3.3 Method

+ Getter method for instance.	

2.2.4 Class `Farm` extends `Generator`

“Grows cookie plants from cookie seeds. How cool is that ?”

The **Farm** is the third generator in the game. It has starting price of 1,100 cookies and will automatically generate 8 cookies every second.

2.2.4.1 Field

- Farm INSTANCE = new Farm()	An instance of the farm.

2.2.4.2 Constructor

- Farm()	Initializes the farm. It will have the image path of the generator image “farm.png”.

2.2.4.3 Method

+ Getter method for instance.	

2.2.5 Class `Mine` extends `Generator`

“Mines out cookie dough and chocolate chips. Let the pickaxe swing.”

The **Mine** is the fourth generator in the game. It has starting price of 12,000 cookies and will automatically generate 57 cookies every second.

2.2.5.1 Field

- Mine INSTANCE = new Mine()	An instance of the mine.

2.2.5.2 Constructor

- Mine()	Initializes the mine. It will have the image path of the generator image “mine.png”.

2.2.5.3 Method

+ Getter method for instance.	

2.2.6 Class `Factory` extends `Generator`

"Produces large quantities of cookies. Hail Cookies !"

The **Factory** is the fifth generator in the game. It has starting price of 130,000 cookies and will automatically generate 260 cookies every second.

2.2.6.1 Field

- Factory INSTANCE = new Factory()	An instance of the factory.

2.2.6.2 Constructor

- Factory()	Initializes the factory. It will have the image path of the generator image "factory.png".

2.2.6.3 Method

+ Getter method for instance.	

2.2.7 Class `Bank` extends `Generator`

"Generates cookies from interest. Aw man, here we go again."

The **Bank** is the sixth and the last generator in this game. It has starting price of 1.4 million cookies and will automatically generate 1,400 cookies every second.

2.2.7.1 Field

- Bank INSTANCE = new Bank()	An instance of the bank.

2.2.7.2 Constructor

- Bank()	Initializes the bank. It will have the image path of the generator image "bank.png".

2.2.7.3 Method

+ Getter method for instance.	

2.3 Package `logic.upgrades`

2.3.1 Package `logic.upgrades.base`

2.3.1.1 *Class `Upgrade` implements `Purchasable`*

This class represents an upgrade in the game. Each upgrade has its own level, starting from 0 to 5. In each level, you have to meet the conditions of that upgrade before it can be purchased. By default, the condition is the amount of that generator type.

2.3.1.1.1 Field

# String name	Upgrade’s name
# Generator generator	Generator type of this upgrade.
# BigDecimal[] upgradePrice	An array representing the price of this upgrade in each level.
# int[] amountCondition	An array representing the condition of this upgrade in each level.
# int upgradeLevel	Current level of this upgrade.
# String imagePath	The image path of this upgrade.

2.3.1.1.2 Constructor

# Upgrade(String name, Generator generator, BigDecimal[] upgradePrice, String imagePath)	Initializes the upgrade with the level starting from 0. The condition for each level is : - 0 : 1 generator - 1 : 5 generators - 2 : 25 generators - 3 : 50 generators - 4 : 100 generators

2.3.1.1.3 Method

+ void buy() throws PurchaseFailedException	This method buys the upgrade. It should : - Subtracts the player's current cookie by the current level’s upgrade price. - Upgrades the generator. - Updates the current level. There are many cases for the method to throw the exception called “PurchasedFailedException” : - If the upgrade level is already maxed out, the exception with error code 0 will be thrown. - If the upgrade level is not maxed out and the conditions are not met, the exception with error code 1 will be thrown. - If the current cookies are less than the required cookies, the exception with error code 2 will be thrown.
+ BigDecimal getUpgradePrice()	Returns current level’s upgrade price.

	If the upgrade's level is maxed out, it will return 0.
+ int getAmountCondition()	Returns current level's upgrade condition. If the upgrade's level is maxed out, it will return 0.
+ Getter method for other fields.	

2.3.2 Class `CursorUpgrade` extends `Upgrade`

The cursor upgrade is called **Enhanced Cursor**. When purchased, both the player's clicking power and the cursor will become **twice** as efficient.

2.3.2.1 Field

- CursorUpgrade INSTANCE = new CursorUpgrade()	An instance of the cursor upgrade.

2.3.2.2 Constructor

+ CursorUpgrade()	Initializes the cursor upgrade. It will have the image path of the upgrade image "cursor.png". The upgrade cost for each level is : - 0 : 100 cookies - 1 : 500 cookies - 2 : 1,000 cookies - 3 : 5,000 cookies - 4 : 10,000 cookies

2.3.2.3 Method

+ Getter method for instance.	

2.3.3 Class `GrandmaUpgrade` extends `Upgrade`

The grandma upgrade is called **Rolling Pins**. When purchased, the grandma will become **twice** as efficient.

2.3.3.1 Field

- GrandmaUpgrade INSTANCE = new GrandmaUpgrade()	An instance of the grandma upgrade.

2.3.3.2 Constructor

- GrandmaUpgrade()	Initializes the grandma upgrade. It will have the image path of the upgrade image “grandma.png”. The upgrade cost for each level is : - 0 : 1,000 cookies - 1 : 5,000 cookies - 2 : 10,000 cookies - 3 : 50,000 cookies - 4 : 100,000 cookies

2.3.3.3 Method

+ Getter method for instance.	

2.3.4 Class `FarmUpgrade` extends `Upgrade`

The farm upgrade is called **Fertilizer**. When purchased, the farm will become **twice** as efficient.

2.3.4.1 Field

- FarmUpgrade INSTANCE = new FarmUpgrade()	An instance of the farm upgrade.

2.3.4.2 Constructor

- FarmUpgrade()	Initializes the farm upgrade. It will have the image path of the upgrade image “farm.png”. The upgrade cost for each level is : - 0 : 11,000 cookies - 1 : 55,000 cookies - 2 : 110,000 cookies - 3 : 550,000 cookies - 4 : 1.1 million cookies

2.3.4.3 Method

+ Getter method for instance.	

2.3.5 Class `MineUpgrade` extends `Upgrade`

The mine upgrade is called **Super Drill**. When purchased, the mine will become **twice** as efficient.

2.3.5.1 Field

- MineUpgrade INSTANCE = new MineUpgrade()	An instance of the mine upgrade.

2.3.5.2 Constructor

- MineUpgrade()	Initializes the mine upgrade. It will have the image path of the upgrade image “mine.png”. The upgrade cost for each level is : - 0 : 120,000 cookies - 1 : 600,000 cookies - 2 : 1.2 million cookies - 3 : 6.0 million cookies - 4 : 12.0 million cookies

2.3.5.3 Method

+ Getter method for instance.	

2.3.6 Class `FactoryUpgrade` extends `Upgrade`

The factory upgrade is called **Conveyor Belts**. When purchased, the factory will become **twice** as efficient.

2.3.6.1 Field

- FactoryUpgrade INSTANCE = new FactoryUpgrade()	An instance of the factory upgrade.

2.3.6.2 Constructor

- FactoryUpgrade()	Initializes the factory upgrade. It will have the image path of the upgrade image “factory.png”. The upgrade cost for each level is : - 0 : 1.3 million cookies - 1 : 6.5 million cookies - 2 : 13.0 million cookies - 3 : 65.0 million cookies - 4 : 130.0 million cookies

2.3.6.3 Method

+ Getter method for instance.	

2.3.7 Class `BankUpgrade` extends `Upgrade`

The bank upgrade is called **Edible Money**. When purchased, the bank will become **twice** as efficient.

2.3.7.1 Field

- BankUpgrade INSTANCE = new BankUpgrade()	An instance of the bank upgrade.

2.3.7.2 Constructor

- BankUpgrade()	Initializes the bank upgrade. It will have the image path of the upgrade image “bank.png”. The upgrade cost for each level is : - 0 : 14.0 million cookies - 1 : 70.0 million cookies - 2 : 140.0 million cookies - 3 : 700.0 million cookies - 4 : 1.4 billion cookies

2.3.7.3 Method

+ Getter method for instance.	

2.3.8 Class `PlayerUpgrade` extends `Upgrade`

The player upgrade is called **Enhanced Cursor**, and doesn’t have any generator type. When purchased, the player's clicking power will become **quadrice** as efficient. Unlike other upgrades which check on the amount of generators, this upgrade checks on total cookies generated by clicking.

2.3.8.1 Field

- PlayerUpgrade INSTANCE = new PlayerUpgrade()	An instance of the player upgrade.
- BigDecimal[] handmadeCondition	An array representing the condition of this upgrade in each level.

2.3.8.2 Constructor

- PlayerUpgrade()	Initializes the player upgrade. It will have the image path of the upgrade image “player.png”. The upgrade cost for each level is : - 0 : 1,000 cookies

	- 1 : 5,000 cookies - 2 : 10,000 cookies - 3 : 50,000 cookies - 4 : 100,000 cookies The condition for each level is : - 0 : 5,000 handmade cookies - 1 : 50,000 handmade cookies - 2 : 500,000 handmade cookies - 3 : 5.0 million handmade cookies - 4 : 50.0 million handmade cookies

2.3.8.3 Method

+ void buy() throws PurchaseFailedException	*This method overrides its parent.* Behaves like normal upgrade with few noticeable differences : - It will check on handmadeCondition instead of amountCondition. - Upgrades the player instead of generators. - If the upgrade level is not maxed out and the conditions are not met, the exception called "PurchaseFailedException" will be thrown with error code 3.
+ BigDecimal getHandmadeCondition()	Returns current level's upgrade condition. If the upgrade's level is maxed out, it will return 0.
+ Getter method for instance.	

2.4 Package `logic.exceptions`

2.4.1 Class `PurchaseFailedException` extends `Exception`

This class is an **Exception** object thrown when failing to purchase the Purchasable object.

2.4.1.1 Constructor

+ PurchaseFailedException(Generator generator)	Initializes the exception thrown by generator's method. It will print the String with the following format : **Error while buying generator called** Generator name **- Not enough cookies!"**
+ PurchaseFailedException(Upgrade upgrade, int errorType)	Initializes the exception thrown by upgrade's method. Depending on errorType, it will print different String format. - 0 : **Error while buying upgrade called** Upgrade name **- Upgrade level already maxed!** - 1 : **Error while buying upgrade called** Upgrade name **- Not enough generators!** - 2 : **Error while buying upgrade called** Upgrade name **- Not enough cookies!** - 3 : **Error while buying upgrade called** Upgrade name **- Not enough handmade cookies!** - etc. : **Error while buying upgrade called** Upgrade name **- Unknown Error!**

2.4.2 Class `SellFailedException` extends `Exception`

This class is an **Exception** object thrown when failing to sell the generator.

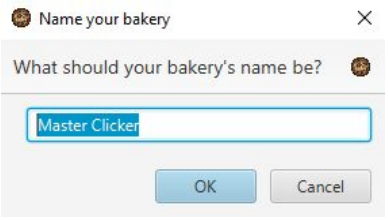
2.4.2.1 Constructor

+ SellFailedException(Generator generator)	Initializes the exception. It will print the String with the following format : **Error while selling** Generator name **- No** Generator name **to sell!**

2.5 Package `application`

2.5.1 Class `CookieBlock` extends `VBox`

The GUI of the cookie block.



The window popped up after clicking the name label.

2.5.1.1 Field

- CookieBlock INSTANCE = new CookieBlock()	An instance of the cookie block.
+ double BOX_WIDTH = 300	Variable of the box width.
- Label name	The label displaying player's name.
- Label amount	The label displaying current cookies.
- Label perSecond	The label displaying current cookies per second.

2.5.1.2 Constructor

- CookieBlock()	Initializes the cookie block. This method should : - Sets width with given BOX_WIDTH variable, and sets alignment to be CENTER.

	- Instantiates the name label and sets the font size to be 22, bold weight, WHITE text color and adds drop shadow effect.
	- Instantiates the amount label and sets the font size to be 28, WHITE text color.
	- Instantiates the perSecond label and sets the font size to be 16, WHITE text color.
	- Update all labels.
	- Instantiates an image of “cookie.png” and sets the size to be 200x200.
	- Instantiates an image of “milkPlain.png” and sets the size to be 300x300.
	- Sets event handler to the name label to :
	  - When hovered, add glow effect with level of 1.0 (while retaining the drop shadow effect), and when mouse exited, remove that glow effect.
	  - When clicked, display a text input dialog with the default value of the player's name when the mouse is released. Sets the title to be “Name your bakery”, header text to be “What should your bakery’s name be?” and set all graphics in the dialog to be “point.png”. If the player changes the name (the result should not be empty String), update the name.
	- Sets event handler to the cookie image to :
	  - When the mouse is pressed, change the size to 220x220. Change the size back to normal size and update current cookies when the mouse is released.
	- Instantiate three VBoxes
	  - Sets height of the first box to

	be 150, alignment to be CENTER, and add all three labels to the box **in order**. - Sets height of the first box to be 240, alignment to be CENTER, and adds the cookie image to the box. - Sets height of the last box to be 300 and adds the milk image to the box. - Add instantiated VBoxes to this box.

2.5.1.3 Method

+ Getter method for instance.	
+ setName()	Updates the name label to display the player's name. The String format is : " Player name**'s Bakery** "
+ setAmount()	Updates the amount label to display the current cookies. The String format is : Current cookies **cookie(s)**
+ setPerSecond()	Updates the perSecond label to display the current cookies per second. The String format is : **per second :** cookies per second

2.5.2 Class `HPanel` extends `HBox`

The GUI of a HPanel.

2.5.2.1 Field

+ double BOX_HEIGHT = 16	Variable of the box height.

2.5.2.2 Constructor

+ HPanel()	Initializes the HPanel. This method should : - Sets height with given BOX_HEIGHT variable. - Sets background image to be "panelHorizontal.png"

2.5.3 Class `VPanel` extends `VBox`

The GUI of a VPanel.

2.5.3.1 Field

+ double BOX_WIDTH = 16	Variable of the box width.

2.5.3.2 Constructor

+ VPanel()	Initializes the VPanel. This method should : - Sets width with given BOX_WIDTH variable. - Sets background image to be “panelVertical.png”

2.5.4 Class `SmallStatBox` extends `HBox`

The GUI of a small stat box.

2.5.4.1 Field

- Label name	The label displaying name of the stat.
- Label amount	The label displaying amount of the stat.

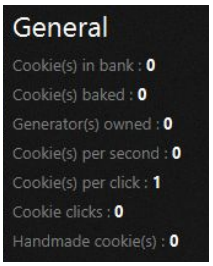
2.5.4.2 Constructor

+ SmallStatBox(String name)	Initializes the stat box. This method should : - Instantiates name label with given name and sets the font size to be 13, GRAY text color. - Instantiates amount label with empty string and sets the font size to be 13, bold weight, WHITE text color. - Sets the padding of 3 from all sides. - Add both labels to the box.

2.5.4.3 Method

+ setAmount(String amount)	Sets the amount label to display the given parameter.

2.5.5 Class `GeneralStatBlock` extends `VBox`



The GUI of the general stat block.

2.5.5.1 Field

- GeneralStatBlock INSTANCE = new GeneralStatBlock()	An instance of the general stat block.
- SmallStatBox currentCookies	The stat box displaying current cookies.
- SmallStatBox totalCookies	The stat box displaying total cookies.
- SmallStatBox totalGenerators	The stat box displaying total generators.
- SmallStatBox totalPerSecond	The stat box displaying total cookies per second.
- SmallStatBox perClick	The stat box displaying cookies per click.
- SmallStatBox totalClick	The stat box displaying total clicks.
- SmallStatBox handmadeCookies	The stat box displaying handmade cookies.

2.5.5.2 Constructor

- GeneralStatBlock()	Initializes the general stat box. This method should : - Instantiates a label with text “General” and sets the font size to be 23, WHITE text color, and sets the padding of 5 from the bottom. - Instantiate all boxes in the field and update the stat boxes. - Instantiates a VBox, sets the padding of 10 from the bottom, and sets the spacing of 5. Then, add all boxes in the field to this box **in order**. - Add instantiated label and VBox to this box.

2.5.5.3 Method

+ Getter method for instance.	
+ setCurrentCookies()	Updates currentCookies box to display current cookies.
+ setTotalCookies()	Updates totalCookies box to display total cookies.
+ setTotalGenerators()	Updates totalGenerators box to display total generators.
+ setTotalPerSecond()	Updates totalPerSecond box to display total cookies per second.
+ setPerClick()	Updates perClick box to display cookies per click.
+ setTotalClick()	Updates totalClick box to display total clicks.
+ setHandmadeCookies()	Updates handmadeCookies box to display handmade cookies.
+ void update()	Updates all stat boxes.

2.5.6 Class `UpgradeStatBlock` extends `VBox`

The GUI of the upgrade stat block.

2.5.6.1 Field

- UpgradeStatBlock INSTANCE = new UpgradeStatBlock()	An instance of the upgrade stat block.
- SmallStatBox playerLevel	The stat box displaying player's clicking level.
- SmallStatBox cursorPerSecond	The stat box displaying cursor's cookies per second.
- SmallStatBox cursorLevel	The stat box displaying cursor's level.
- SmallStatBox grandmaPerSecond	The stat box displaying grandma's cookies per second.
- SmallStatBox grandmaLevel	The stat box displaying grandma's level.
- SmallStatBox farmPerSecond	The stat box displaying farm's cookies per second.
- SmallStatBox farmLevel	The stat box displaying farm's level.
- SmallStatBox minePerSecond	The stat box displaying mine's cookies per second.
- SmallStatBox mineLevel	The stat box displaying mine's level.
- SmallStatBox factoryPerSecond	The stat box displaying factory's cookies per second.
- SmallStatBox factoryLevel	The stat box displaying factory's level.
- SmallStatBox bankPerSecond	The stat box displaying bank's cookies per second.
- SmallStatBox bankLevel	The stat box displaying bank's level.

2.5.6.2 Constructor

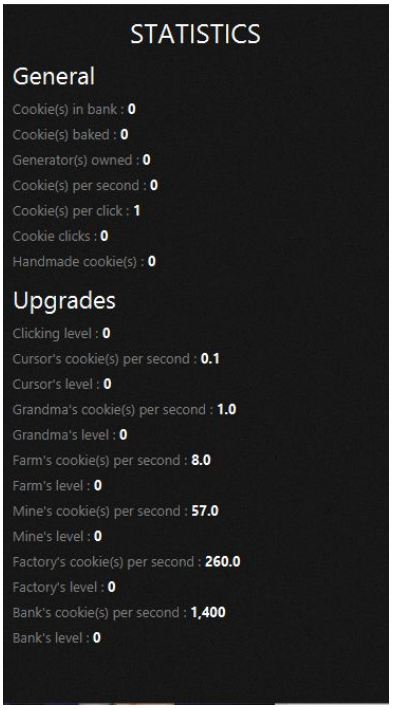
- UpgradeStatBlock()	Initializes the upgrade stat box. This method should : - Instantiates a label with text “Upgrades” and sets the font size to be 23, WHITE text color, and sets the padding of 5 from the bottom. - Instantiate all boxes in the field and update the stat boxes. - Instantiates a VBox, sets the padding of 10 from the bottom, and sets the spacing of 5. Then, add all boxes in the field to this box **in order**. - Add instantiated label and VBox to this box.

2.5.6.3 Method

+ Getter method for instance.	
+ void setPlayerLevel()	Updates playerLevel box to display player’s clicking level.
+ void setCursorPerSecond()	Updates cursorPerSecond box to display cursor’s cookies per second.
+ void setCursorLevel()	Updates cursorLevel box to display cursor’s level.
+ void setGrandmaPerSecond()	Updates grandmaPerSecond box to display grandma’s cookies per second
+ void setGrandmaLevel()	Updates grandmaLevel box to display grandma’s level.
+ void setFarmPerSecond()	Updates farmPerSecond box to display farm’s cookies per second
+ void setFarmLevel()	Updates farmLevel box to display farm’s level.
+ void setMinePerSecond()	Updates minePerSecond box to display mine’s cookies per second
+ void setMineLevel()	Updates mineLevel box to display mine’s level.

+ void setFactoryPerSecond()	Updates factoryPerSecond box to display factory's cookies per second
+ void setFactoryLevel()	Updates factoryLevel box to display factory's level.
+ void setBankPerSecond()	Updates bankPerSecond box to display bank's cookies per second
+ void setBankLevel()	Updates bankLevel box to display bank's level.
+ void update()	Updates all stat boxes.

2.5.7 Class `StatBlock` extends `VBox`



The GUI of the stat block.

2.5.7.1 Field

- StatBlock INSTANCE = new StatBlock()	An instance of the stat block.
- double BOX_WIDTH = 368	Variable of the box width.

2.5.7.2 Constructor

- StatBlock()	Initializes the stat block. This method should :

	- Sets the padding of 10 from all sides, sets width with given BOX_WIDTH variable, and sets its background to be “blackbg.png". - Instantiates a label with text “STATISTICS” and sets the font size to be 25, WHITE text color. - Instantiates a VBox, sets the alignment to be CENTER, and sets the padding of 5 from the bottom. Then, add the label to this box - Add the instantiated VBox, general stat box and upgrade stat box to the box **in order**.

2.5.7.3 Method

+ Getter method for instance.	
+ void update()	Updates all components in general stat box and upgrade stat box.

2.5.8 Class `MenuBar` extends `BorderPane`

The GUI of the menu bar.

2.5.8.1 Field

- MenuBar INSTANCE = new MenuBar()	An instance of the menu bar.
- StackPane leftSide	The left side of the menu bar.
- StackPane rightSide	The right side of the menu bar.

2.5.8.2 Constructor

- MenuBar()	Initializes the menu bar. This method should : - Instantiates the leftSide pane, along with an image of “leftMenu.png” - Instantiates a label with text “Generators”, sets the size to 18, bold weight, WHITESMOKE text color, and adds a drop shadow

	effect with the radius of 10 and BLACK color. - Add the leftMenu image and the label to the leftSide pane. - Instantiates the rightSide pane, along with an image of “rightMenu.png” - Instantiates a label with text “Upgrades”, sets the size to 18, bold weight, WHITESMOKE text color, and adds a drop shadow effect with the radius of 10 and BLACK color. - Add the rightMenu image and the label to the rightSide pane. - Sets the leftSide pane to the left side of this pane, and sets the rightSide pane to the right side of this pane.

2.5.8.3 Method

+ Getter method for instance.	

2.5.9 Class `BuyAndSellMenu` extends `HBox`

The GUI of the buy and sell menu, in buy mode.

The GUI of the buy and sell menu, in sell mode.

2.5.9.1 Field

- BuyAndSellMenu INSTANCE = new BuyAndSellMenu()	An instance of the buy and sell menu.
- double BOX_WIDTH = 300	Variable of the box width.
- double BOX_HEIGHT = 32	Variable of the box height.
- int mode = 0	Current mode of the menu. - 0 : buy mode - 1 : sell mode

2.5.9.2 Constructor

- BuyAndSellMenu()	Initializes the buy and sell menu. This method should : - Sets the alignment to be CENTER. - Instantiates each node : - `buy` is a label with text “BUY”. - `sell` is a label with text “SELL”. - Sets the font size of both nodes to be 16, bold weight. Then, set both nodes’ height to be given BOX_HEIGHT variable, the width to be half of given BOX_WIFTH variable, and the alignment to be CENTER. - Adds glow effect with level of 1.0 to `buy`. - Sets event handlers for both labels to : - When hovered, add glow effect with level of 1.0 if current mode doesn’t match the name that mouse hovered to, and when mouse exited, remove the effect. - When clicked, set the mode to match the name of the label, set the generator block to match current mode, add glow effect with level of 1.0, and remove effect from the other label when the mouse is released. *(For more information about generator block, check section 2.5.12)*

2.5.9.3 Method

+ Getter method for instance and mode.	

2.5.10 Class `GeneratorDescriptionBox` extends `VBox`

The GUI of a generator description box.

2.5.10.1 Field

- double BOX_WIDTH = 190	Variable of the box width.
- double BOX_HEIGHT = 50	Variable of the box height.
- Color GREEN = Color.web("#61F161FF")	Variable of the buy price color.
- Color RED = Color.web("#FF6666FF")	Variable of the sell price color.
- Label name	The label displaying generator's name.
- Label amount	The label displaying generator's amount.
- Label price	The label displaying generator's price.

2.5.10.2 Constructor

+ GeneratorDescriptionBox()	Initializes the generator description box. This method should : - Sets height with given BOX_HEIGHT variable, width with given BOX_WIDTH variable, alignment to be CENTER_LEFT. - Instantiates name label and sets the font size to be 25, bold weight, WHITESMOKE text color. Then, add a drop shadow effect with the radius of 5 and BLACK color. - Instantiates amount label and sets the font size to be 12. - Instantiates price label and sets the font size to be 14, bold weight. Then, add a drop shadow effect with the radius of 3 and BLACK color. - Update the amount and price label. - Instantiates two HBoxes and an image of "point.png" - Sets the alignment of the first box to be BASELINE_LEFT, and sets spacing of 5. Then, add name and amount label to the box. - Sets the alignment of the first

	box to be CENTER_LEFT, and sets spacing of 5. Then, add instantiated image and price label to the box. - Add instantiated HBoxes to this box.

2.5.10.3 Method

+ void setBuyPrice(BigDecimal price)	Sets the price label to display the given parameter and sets the color.
+ void setSellPrice(BigDecimal price)	Sets the price label to the display given parameter and sets the color.
+ void setAmount(int amount)	Sets the amount label to display the given parameter. The String format is : amount**x** (example : **15x**)

2.5.11 Class `GeneratorTile` extends `HBox`

The GUI of a generator tile.

2.5.11.1 Field

- double BOX_WIDTH = 300	Variable of the box width.
- double BOX_HEIGHT = 64	Variable of the box height.
- Generator generator	Generator type of this tile.
- GeneratorDescriptionBox description	Generator description box of this tile.

2.5.11.2 Constructor

+ GeneratorTile(Generator generator)	Initializes the generator tile. This method should : - Sets height with given BOX_HEIGHT variable, width with given BOX_WIDTH variable, alignment to be CENTER_LEFT and sets background to be “storeTile.png”. - Instantiate every field and updates buy price.

	- Instantiate an image of the generator and an audio clip of “generatorBuy.mp3”.
	- Sets event handler to this box to :
	  - When the mouse is pressed, add an inner shadow effect with radius of 30.
	  - When the mouse is released, remove the effect. If the current mode is buy mode, try buying the generator. If success, play the sound and update the price, current cookies and cookies per second. But if it is sell mode, try selling the generator instead of buying.
	- Add instantiated image and description box to this box.

2.5.11.3 Method

+ void setBuy()	Updates buy price.
+ void setSell()	Updates sell price.

2.5.12 Class `GeneratorBlock` extends `VBox`

The GUI of the generator block.

2.5.12.1 Field

- GeneratorBlock INSTANCE = new GeneratorBlock()	An instance of the generator block.

- double BOX_WIDTH = 300	Variable of the box width.
- GeneratorTile cursorTile	The generator tile of the cursor.
- GeneratorTile grandmaTile	The generator tile of the grandma.
- GeneratorTile farmTile	The generator tile of the farm.
- GeneratorTile mineTile	The generator tile of the mine.
- GeneratorTile factoryTile	The generator tile of the factory.
- GeneratorTile bankTile	The generator tile of the bank.

2.5.12.2 Constructor

- GeneratorBlock()	Initializes the generator block. This method should : - Sets width with given BOX_WIDTH variable. - Instantiate all generator tiles. - Add buy and sell menu, and all generator tiles to the box **in order**.

2.5.12.3 Method

+ void setBuyAll()	Sets all tiles to be in buy mode.
+ void setSellAll()	Sets all tiles to be in sell mode.
+ Getter method for instance.	

2.5.13 Class `UpgradeDescriptionBox` extends `VBox`

The GUI of an upgrade description box.

2.5.13.1 Field

- double BOX_WIDTH = 190	Variable of the box width.
- double BOX_HEIGHT = 50	Variable of the box height.
- Color GREEN = Color.web("#61F161FF")	Variable of the price color.
- Label name	The label displaying upgrade's name.

- Label amountCondition	The label displaying upgrade's condition.
- Label price	The label displaying upgrade's price.

2.5.13.2 Constructor

+ UpgradeDescriptionBox()	Initializes the upgrade description box. This method should : - Sets height with given BOX_HEIGHT variable, width with given BOX_WIDTH variable, alignment to be CENTER_LEFT. - Instantiates name label and sets the font size to be 18, bold weight, WHITESMOKE text color. Then, add a drop shadow effect with the radius of 5 and BLACK color. - Instantiates amount condition label and sets the font size to be 12. - Instantiates price label and sets the font size to be 14, bold weight. Then, add a drop shadow effect with the radius of 3 and BLACK color. - Update the amount condition and price label. - Instantiates a HBox and an image of "point.png" - Sets the alignment of the first box to be CENTER_LEFT, and sets spacing of 5. Then, add instantiated image and price label to the box. - Add name and amount condition label, and instantiated HBox to this box.

2.5.13.3 Method

+ void setAmountCondition(Upgrade upgrade)	Sets the price label to display the condition. The String format is : - If the upgrade level is already maxed out : **This upgrade is maxed out** - If the upgrade level is not maxed out and it is not player upgrade :

	Requires amount condition**x** generator name - If the upgrade level is not maxed out and it is player upgrade : **Requires** amount condition **handmade cookies**
+ setPrice(BigDecimal price)	Sets the price label to the display given parameter and sets the color.

2.5.14 Class `UpgradeTile` extends `HBox`

The GUI of an upgrade tile.

2.5.14.1 Field

- double BOX_WIDTH = 300	Variable of the box width.
- double BOX_HEIGHT = 64	Variable of the box height.
- Upgrade upgrade	Upgrade type of this tile.
- UpgradeDescriptionBox description	Upgrade description box of this tile.

2.5.14.2 Constructor

+ UpgradeTile()	Initializes the upgrade tile. This method should :
	- Sets height with given BOX_HEIGHT variable, width with given BOX_WIDTH variable, alignment to be CENTER_LEFT and sets background to be “storeTile.png”.
	- Instantiate every field and updates upgrade price.
	- Instantiate an image of the upgrade and an audio clip of “generatorBuy.mp3”.
	- Sets event handler to this box to :
	  - When the mouse is pressed, add an inner shadow effect with radius of 30.
	  - When the mouse is released,

	remove the effect. Then, try purchasing the generator. If success, play the sound and update the price, current cookies and cookies per second. - Add instantiated image and description box to this box.

2.5.14.3 Method

+ setUpgrade()	Update upgrade's amount condition and price.

2.5.15 Class `UpgradeBlock` extends `VBox`

The GUI of the upgrade block.

2.5.15.1 Field

- UpgradeBlock INSTANCE = new UpgradeBlock()	An instance of the upgrade block.
- double BOX_WIDTH = 300	Variable of the box width.
- UpgradeTile cursorUpgradeTile	The upgrade tile of the cursor upgrade.
- UpgradeTile grandmaUpgradeTile	The upgrade tile of the grandma upgrade.
- UpgradeTile farmUpgradeTile	The upgrade tile of the farm upgrade.
- UpgradeTile mineUpgradeTile	The upgrade tile of the mine upgrade.

- UpgradeTile factoryUpgradeTile	The upgrade tile of the factory upgrade.
- UpgradeTile bankUpgradeTile	The upgrade tile of the bank upgrade.
- UpgradeTile playerUpgradeTile	The upgrade tile of the player upgrade.

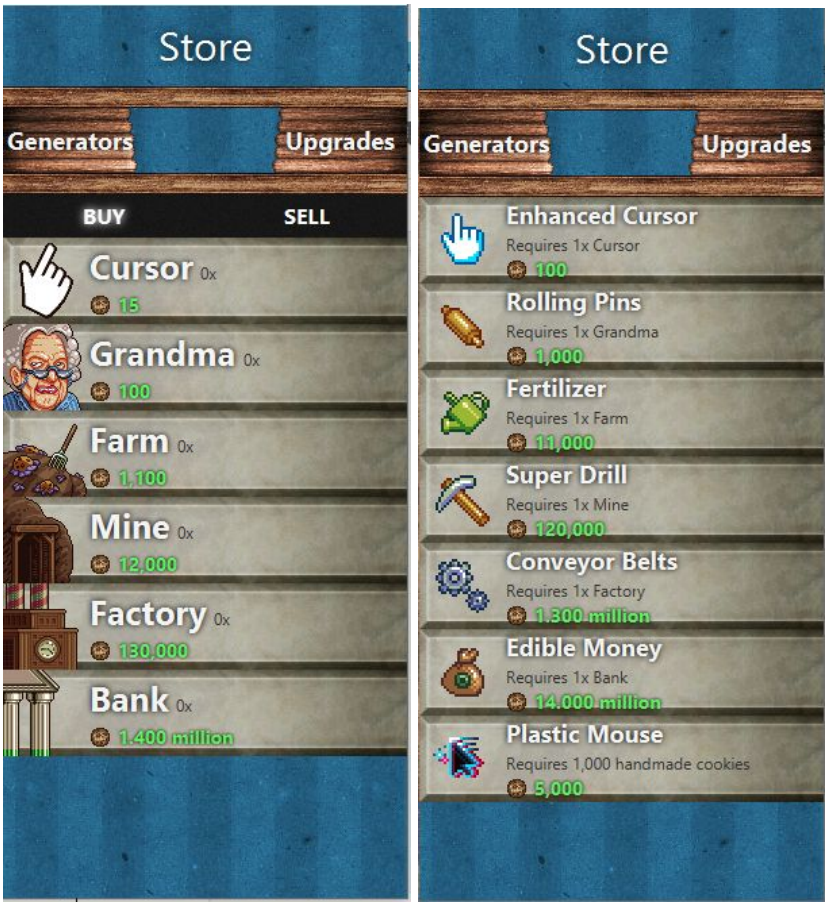
2.5.15.2 Constructor

- UpgradeBlock()	Initializes the upgrade block. This method should : - Sets width with given BOX_WIDTH variable. - Instantiate all upgrade tiles. - Add all upgrade tiles to the box **in order**.

2.5.15.3 Method

+ Getter method for instance.	

2.5.16 Class `StoreBlock` extends `VBox`



The GUI of the store block.

2.5.16.1 Field

- StoreBlock INSTANCE = new StoreBlock()	An instance of the store block.
- double BOX_WIDTH = 300	Variable of the box width.

2.5.16.2 Constructor

- StoreBlock()	Initializes the store block. This method should : - Sets width with given BOX_WIDTH variable and sets the alignment to be TOP_CENTER. - Instantiates a label with text “Store” and sets the font size to be 30, its height to be 60, WHITE text color, and adds a drop shadow effect. - Instantiate two HPanels. - Sets event handler to the left side of the menu bar to display only generator block when clicked (only when the mouse is released). - Sets event handler to the right side of the menu bar to display only upgrade block when clicked (only when the mouse is released). - Add the “Store” label, first HPanel, menu bar, second HPanel, and generator block to the box.

2.5.16.3 Method

+ Getter method for instance.	

2.6 Package `application.main`

2.6.1 Class `Main` extends `Application`

2.6.1.1 Field

- double WINDOW_WIDTH = 1000	Variable of the application width.
- double WINDOW_HEIGHT = 690	Variable of the application height.
- Thread gameRunner = new Thread(new GameLoop())	The thread that will be used as a game loop.

- MediaPlayer bgm	The media player of application.

2.6.1.2 Method

+ void start(Stage primaryStage)	The main entry point of the JavaFX application. This method should : - Instantiate two VPanels and a `root` container using HBox, then sets the background of the root container to be “bg.png”. - Add the cookie block, first VPanel, stat block, second VPanel, store block to the root container **in order**. - Instantiates the media player with “bgm.mp3”, sets cycle to be indefinite, and plays the media player. - Sets the stage’s width to be WINDOW_WIDTH, its height to be WINDOW_HEIGHT, and makes it not resizable. - Sets the appropriate scene with the title "Cookie Clicker" and sets the icon to be “point.png”, then shows the stage. - Starts the game loop thread.
+ void main(String[] args)	An entry point of the application.
+ void stop() throws Exception	The override method that will stop all thread when we stop the application.

3. Appendix (About BigDecimal)

3.1 What is BigDecimal ?

`BigDecimal` *(`java.math.BigDecimal`)* is a subclass of `Number` *(`java.lang.Number`)*. It can handle very large or very small numbers with great precision.

A `BigDecimal` consists of an arbitrary precision integer *unscaled value* and a 32-bit integer *scale*. If zero or positive, the scale is the number of digits to the right of the decimal point. If negative, the unscaled value of the number is multiplied by ten to the power of the negation of the scale. The value of the number represented by the `BigDecimal` is therefore ($unscaledValue \times 10^{-scale}$).

The `BigDecimal` class provides operations for arithmetic, scale manipulation, rounding, comparison, hashing, and format conversion. The `toString()` method provides a canonical representation of a BigDecimal.

The `BigDecimal` class gives its user complete control over rounding behavior (*using the enumeration values of the `RoundingMode` enum, such as `RoundingMode.HALF_UP`*). If no rounding mode is specified and the exact result cannot be represented, an exception is thrown; otherwise, calculations can be carried out to a chosen precision and rounding mode by supplying an appropriate `MathContext` object to the operation. In either case, eight rounding modes are provided for the control of rounding.

Besides a logical exact result, each arithmetic operation has a preferred scale for representing a result. The preferred scale for each operation is listed in the table below.

Operation	**Preferred Scale of Result**
Add	max(addend.scale(), augend.scale())
Subtract	max(minuend.scale(), subtrahend.scale())
Multiply	multiplier.scale() + multiplicand.scale()
Divide	dividend.scale() - divisor.scale()
Square root	radicand.scale()/2

These scales are the ones used by the methods which return exact arithmetic results; except that an exact divide may have to use a larger scale since the exact result may have more digits. For example, 1/32 is 0.03125.

3.2 BigDecimal and RoundingMode features used in the project

3.2.1 Field

+ BigDecimal ZERO	The value 0, with a scale of 0.
+ BigDecimal ONE	The value 1, with a scale of 0.
+ RoundingMode UP	Rounding mode to round away from zero.
+ RoundingMode DOWN	Rounding mode to round towards zero.
+ RoundingMode HALF_UP	Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up.

3.2.2 Constructor

+ BigDecimal(int val)	Translates an `int` into a `BigDecimal`.
+ BigDecimal(String val)	Translates the String representation of a `BigDecimal` into a `BigDecimal`.

3.2.3 Method

+ int compareTo(BigDecimal val)	Compares this `BigDecimal` with the specified `BigDecimal`. Returns : - -1 if this `BigDecimal` is numerically less than `val`. - 0 if this `BigDecimal` is numerically equal to `val`. - 1 if this `BigDecimal` is numerically greater than `val`.
+ BigDecimal add(BigDecimal augend)	Returns a `BigDecimal` whose value is `(this + augend)`, and whose scale is `max(this.scale(), augend.scale())`.
+ BigDecimal subtract(BigDecimal subtrahend)	Returns a `BigDecimal` whose value is `(this - subtrahend)`, and whose scale is `max(this.scale(), subtrahend.scale())`.
+ BigDecimal multiply(BigDecimal multiplicand)	Returns a `BigDecimal` whose value is `(this × multiplicand)`, and whose

	scale is `(this.scale() + multiplicand.scale())`.
+ BigDecimal divide(BigDecimal divisor)	Returns a `BigDecimal` whose value is `(this / divisor)`, and whose preferred scale is `(this.scale() - divisor.scale())`; if the exact quotient cannot be represented (because it has a non-terminating decimal expansion) an `ArithmeticException` is thrown.
+ BigDecimal divide(BigDecimal divisor, RoundingMode roundingMode)	Returns a `BigDecimal` whose value is `(this / divisor)`, and whose scale is `this.scale()`.
+ BigDecimal remainder(BigDecimal divisor)	Returns a `BigDecimal` whose value is `(this % divisor)`.
+ BigDecimal scaleByPowerOfTen(int n)	Returns a `BigDecimal` whose numerical value is equal to $(this \times 10^{n})$.
+ BigDecimal setScale(int newScale, RoundingMode roundingMode)	Returns a `BigDecimal` whose scale is the specified value, and whose unscaled value is determined by multiplying or dividing this `BigDecimal`'s unscaled value by the appropriate power of ten to maintain its overall value.
+ String toString()	Returns the string representation of this `BigDecimal`, using scientific notation if an exponent is needed.