

# **Gotcooked! Documentation**

**Created by**

Tanakit Banluprasong 6731321421

Panat Lorchatchawankul 6731331721

**2110215 Programming Methodology**

**Semester 2 Year 2024**

**Chulalongkorn University**

# Gotcooked!

## Introduction

Gotcooked is inspired by the famous game, named Overcooked. We wanted to know how the game worked ,so we chose a cooking game.

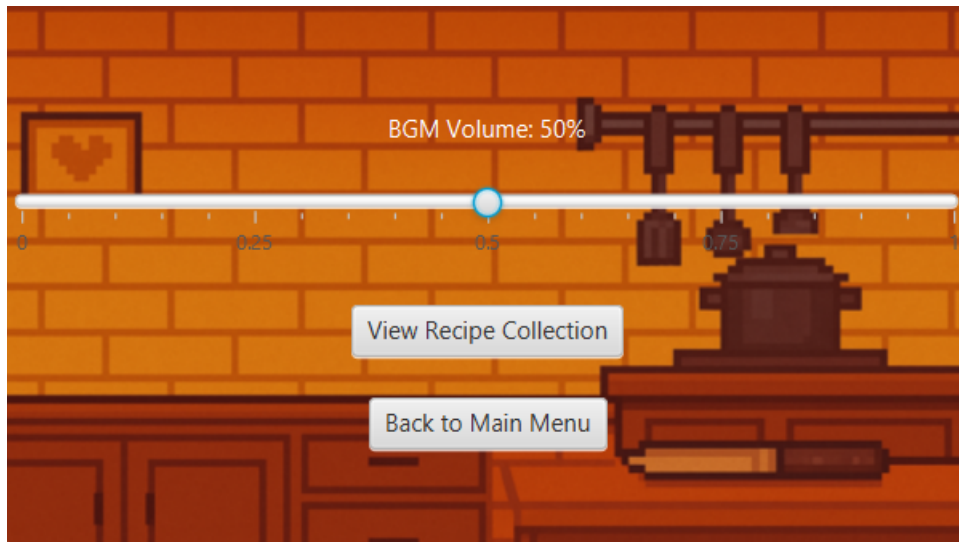
## Rule

The rule is simple: there is a time limiter which counts down from 5 minutes. Player has to make the menus from the random order and serve it. After serving the menu, the player will earn the score which menus have their own scores. Try to make as many scores as the player can in 5 minutes and be proud of your result.

## MainMenu Scene



- Start Button: Initialize the game.
- Options Button: Using for adjusting game sound and recipe collection.



## Recipe Collection.



Menu	Tool / Station	Ingredient (Add them in order)	Container for serve
Soup	Pot	- Water - Wagame (Processed) - Miso	Bowl
Pancake	Pan	- Egg - Flour - Sugar	Dish
Omurice	Pan	- Egg	Dish

		- Rice (cooked)	
Tuna Sushi	Rolling Mat	- Seaweed - Tuna (Chopped) - Rice (Cooked)	Dish
Ramen	Pot	- Soup - Noodle - Fried Pork	Bowl
Salmon Sushi	Rolling Mat	- Seaweed - Salmon (Chopped) - Rice (Cooked)	Dish
Fried Pork	Pan	- Pork	Dish
Rice	Pot	- Water - Raw Rice	Bowl
Bread	Pan	- Egg - Flour	Dish

## Game Example



- The top right corner will show Time, Score and Hand state.
- The top left corner will show the random order that the player has to finish it.












- This is a fridge that stores all ingredients.



- This selects the kitchen tools scene for making and serving food.

## - All station functions.

Station	Function
 Chiller	Choose the ingredients.
 Chopping Station	Chop the ingredient that required chopping (Unchopped).
 Table	Can put the container on it.
 Serving Station	Serve the menu.
 Trashcan	Remove item from hand.
 Rolling Mat Station	Using it to make tuna sushi and salmon sushi.
 Sink	Add water into the pot or wash some ingredients (Unwashed).
 Stove	Cook the food by using a pan or pot.
 Kitchenhold table	Choose the tool for processing the menu.

## - Cooking Example

- Our order is Rice. So let's pick the pot and add some water.



- Put the rice into the pot.

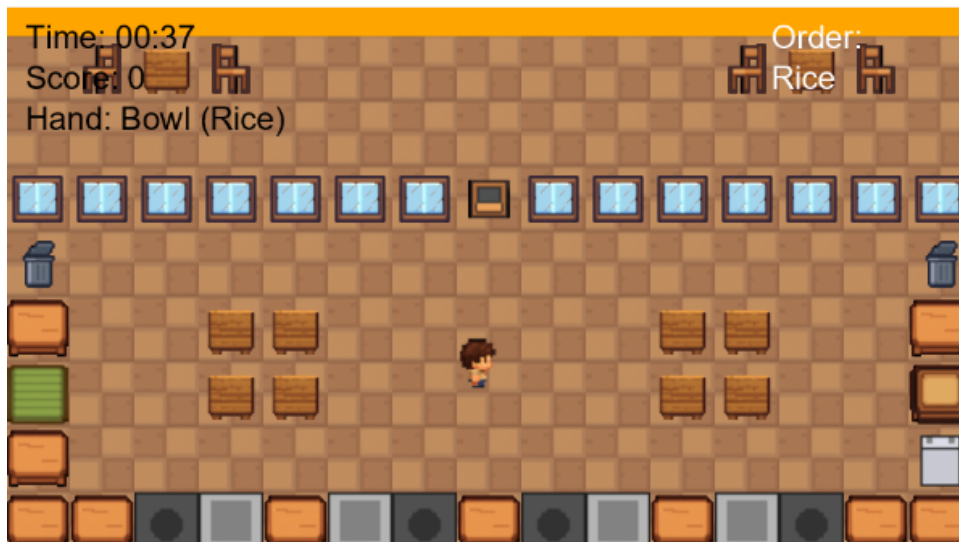


- Cook the rice on the stove

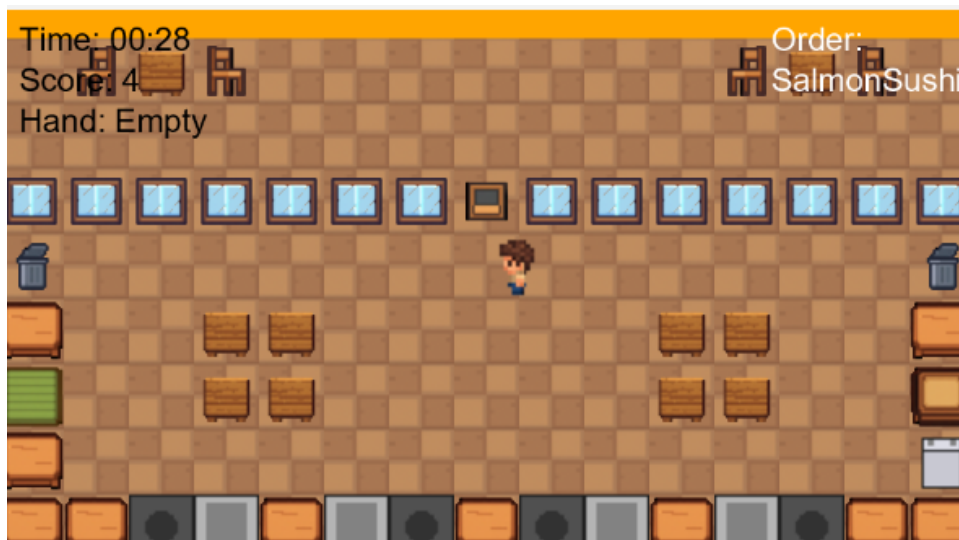


- After that, put the cooked rice into the bowl. (You need to prepare the bowl first)



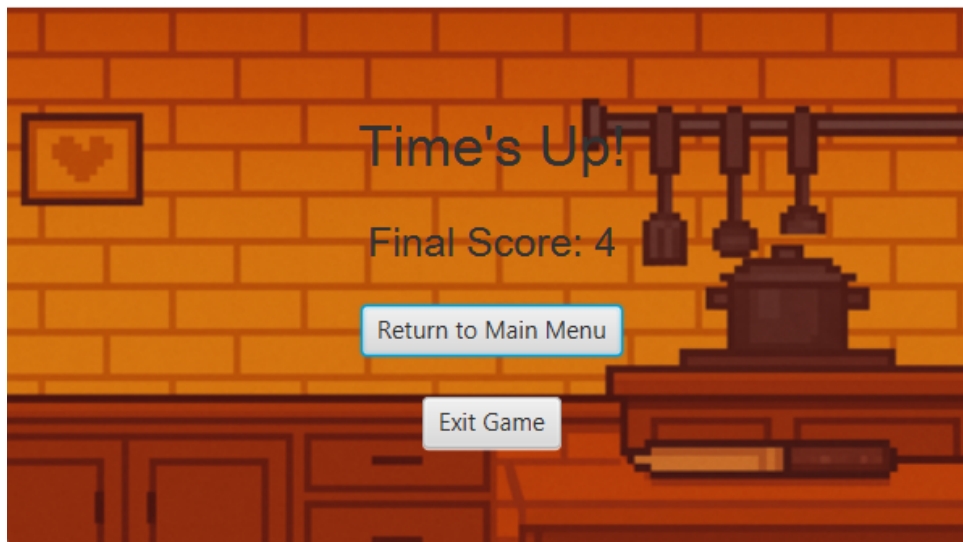


- Finally serve the order at the counter bar and receive the new order!!!

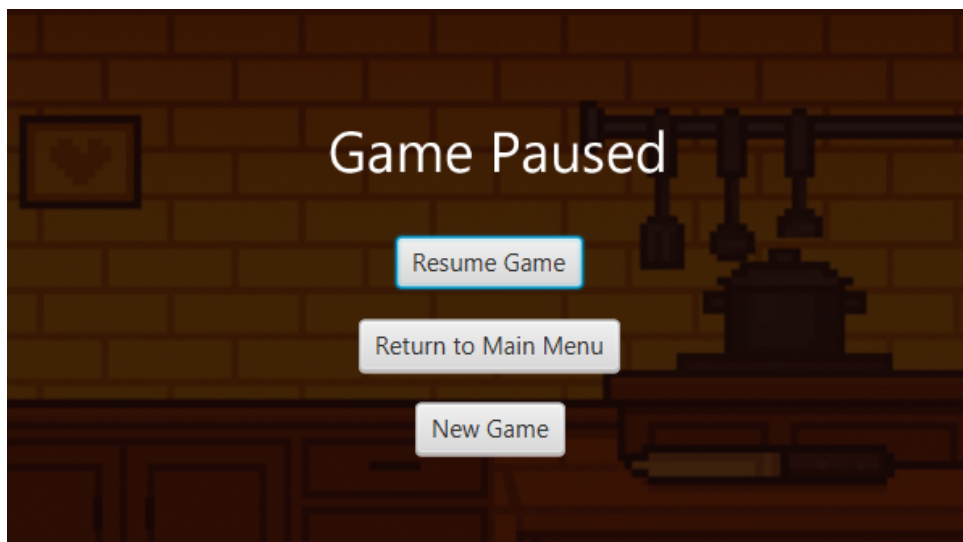


- When time is up after 5 mins, the result screen will appear and you will be proud of your cooking skill !!!

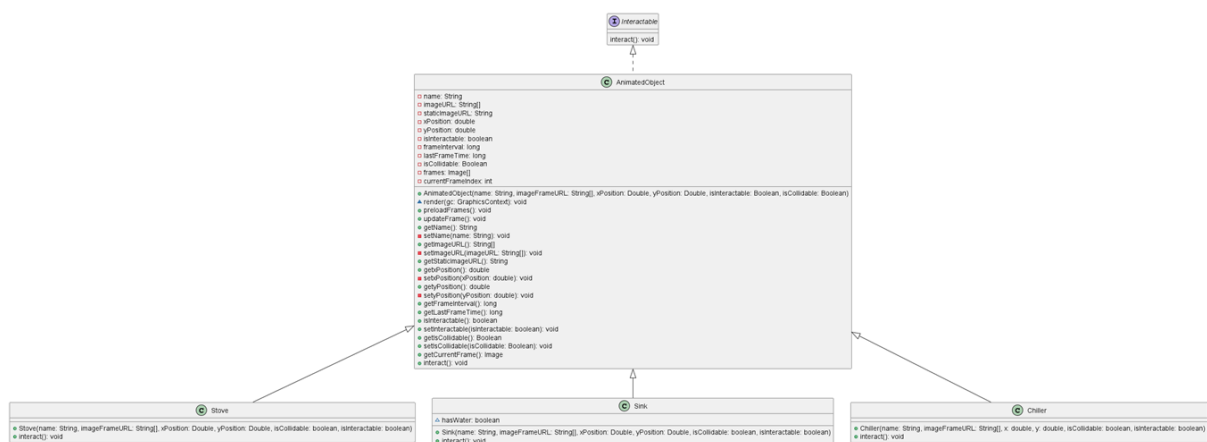
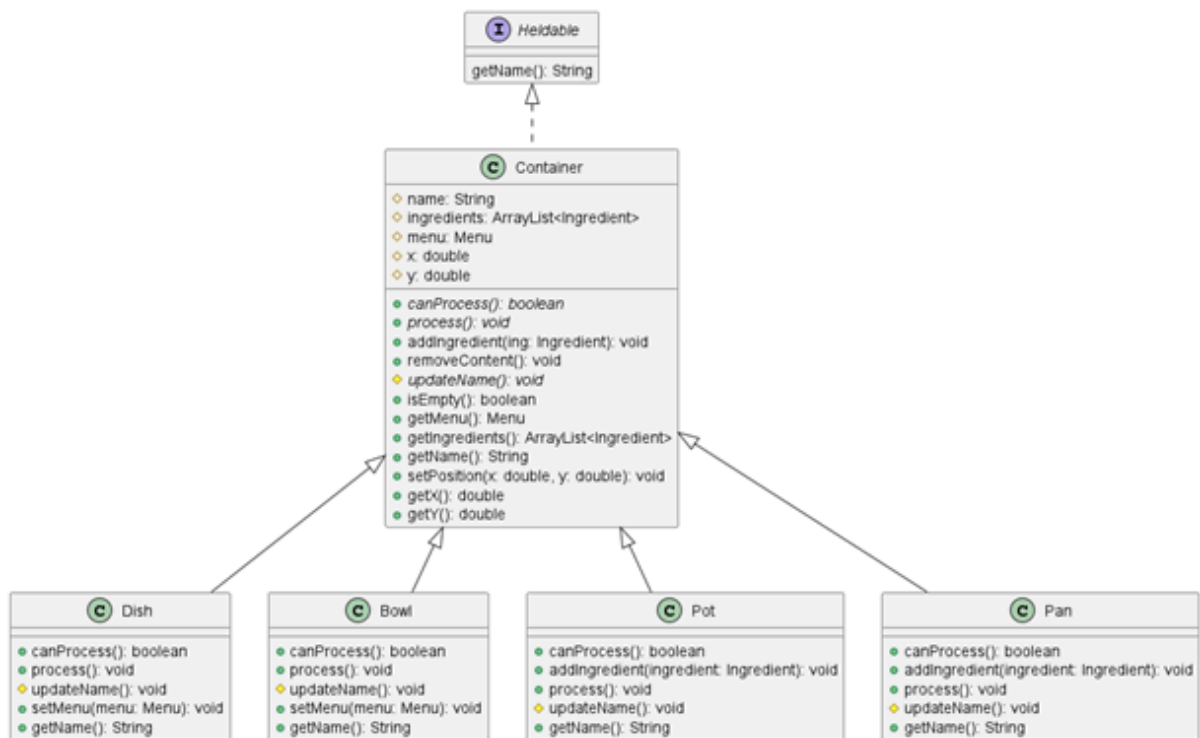


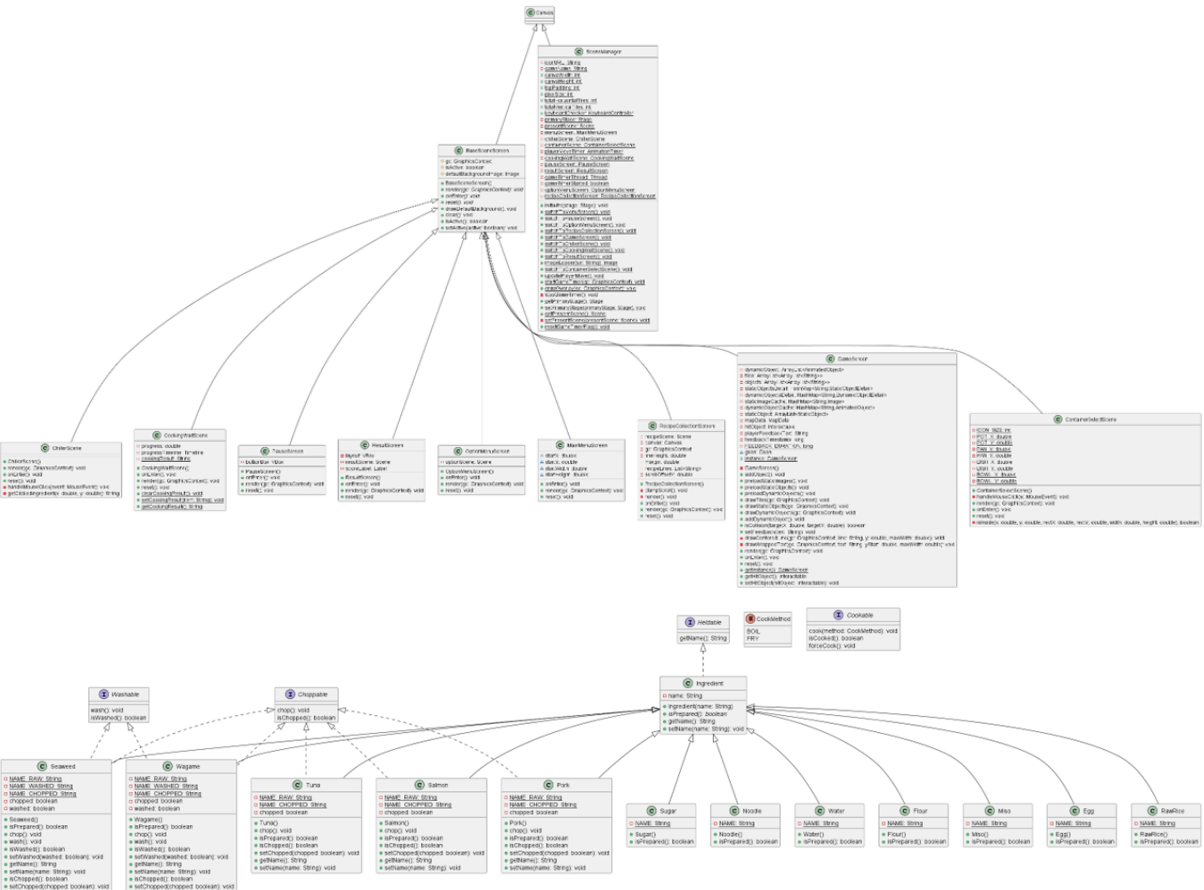


- This is a pause screen where the player can Resume the game, return to the main menu or new game.



## Class Diagram.







## 1.Package Main

### 1.1 Class Main extends Application

#### 1.1.1 Methods

+ Static void main(String[] args)	Initialize the game.
+ void start(Stage PrimaryStage)	-Create Scene from SceneManager and call initialize method from SceneManager in which argument is primaryStage -Set Game end to false

## 2.Package Logic

### 2.1 Class AnimatedObject - this Class represents AnimatedObject blueprint

#### 2.1.1 Fields

- String name	Name of object obtain from data
- String[] imageURL	Array of imageURL for objects that have animations.
- String staticImageURL	ImageURL for objects that have one Image.
- double xPosition	xPosition of object (this position means position on the 2D array grid not on Canvas itself)
- double yPosition	YPosition of object (this position means position on the 2D array grid not on Canvas itself)
- boolean isInteractable	Boolean flag indicating whether the object supports user interaction, such as being picked up, opened, or activated.
- long frameInterval	frameInterval for each animation = 100_000_000 default 10 fps
- long lastFrameTime	Last Frametime that object is on

- Boolean isCollidable	Boolean flag indicating whether the object supports user collision
- Image[] frames	Keep Image Object in array
- int currentFrameIndex	use to pick Image from array (->->->)

## 2.1.2 Constructor

+ AnimatedObject(String name,String[] imageFrameURL,Double xPosition,Double yPosition,Boolean isInteractable,Boolean isCollidable)	Constructor for create AnimatedObject with required parameter
--	---

## 2.1.3 Methods

+ render(GraphicsContext gc)	draw currentFrame Image
+ void preloadFrames()	add Image object of imageURL into frames array
+ void updateFrame()	if 10 frame pass change FrameIndex by one it can be used to change picture animations
+ String getName()	getter for name
- void setName(String name)	setter for name
+ String[] getImageURL()	getter for imageURL array
- void setImageURL(String[] imageURL)	setter for imageURL
+ String getStaticImageURL()	getter for staticImageURL
+ double getXPosition()	getter for xPosition
- void setXPosition(double xPosition)	setter for xPosition
+ double getYPosition()	getter for yPosition
- void setYPosition(double yPosition)	setter fot yPosition
+ long getFrameInterval()	getter for frameInterval

+ long getLastFrameTime()	getter for lastFrameTime
+ boolean isInteractable()	getter for isInteraction
+ void setInteractable(boolean isInteractable)	setter for isInteraction
+ Boolean getIsCollidable()	getter for isCollidable
+ void setIsCollidable(Boolean isCollidable)	setter for isCollidable
+ Image getCurrentFrame()	getter for CurrentFrame

## 2.2 Class Enum Direction

### 2.2.1 Enum

UP	Represent the player face up direction.
DOWN	Represent the player face down direction.
LEFT	Represent the player face to the left direction.
RIGHT	Represent the player face to the right direction.

2.3 Class Map Data - This class is made so that JsonParsing can be done single time and assign field without declare reader or type

### 2.3.1 Fields

- ArrayList<ArrayList<String>> tiles	A 2D array for keeping tile object
- ArrayList<ArrayList<String>> objects	A 2D array for keeping objects
- HashMap<String, StaticObjectDetail> staticObjectsDetail	Hashmap to make static object can be search fast from name



- HashMap<String, DynamicObjectDetail> dynamicObjectsDetail	Hashmap to make dynamic object can be search fast from name
--	---

### 2.3.2 Methods

+ ArrayList<ArrayList<String>> getTiles()	getter for tiles 2D array
+ ArrayList<ArrayList<String>> getObjects()	getter for objects 2D array
+ HashMap<String, StaticObjectDetail> getStaticObjectsDetail()	getter for staticObjectsDetail hashmap
+ HashMap<String, DynamicObjectDetail> getDynamicObjectsDetail()	getter for dynamicObjectsDetail hashmap

## 2.4 class DynamicObjectDetail - This is a blueprint for keeping DynamicObject Data

### 2.4.1 Fields

- String[] imageFrameURL	Store url of the image.
- double x	xPosition of object (this position means position on the 2D array grid not on Canvas itself)
- double y	yPosition of object (this position means position on the 2D array grid not on Canvas itself)
- boolean isInteractable	Boolean flag indicating whether the object supports user interaction, such as being picked up, opened, or activated.
- boolean isCollidable	Boolean flag indicating whether the object supports user collision
- long frameInterval	frameInterval for each animation = 100_000_000

	default 10 fps
- String name	Name of the object.

## 2.4.2 Method

Getters & Setters	getter and setter for field above
-------------------	-----------------------------------

2.5 class DynamicObjectFactory - This class is used to create DynamicObject instances and classify them if they are of a special type (e.g., Stove, Sink, Chiller).

## 2.5.1 Method

<u>+ AnimatedObject createDynamicObject(String name, DynamicObjectDetail detail)</u>	This method is used to create and return a DynamicObject using the specified name and its associated detail parameters.
--	---

2.6 class GameManager - This class is used to manage gameplay logic part on GameScreen

## 2.6.1 Fields

<u>- boolean isGameEnded</u>	Default End game state in a false state.
<u>- boolean gameStarted</u>	Default Start game state in a false state.
<u>- int score</u>	Default score to 0.
<u>- int timeLeft</u>	Default time to 300 seconds.

## 2.6.2 Method

<u>+ double positionX(double playerXPos)</u>	Player X axis calculator method to convert to axis on Canva.
<u>+ double positionY(double</u>	Player Y axis calculator method to

<u>playerYPos)</u>	convert to axis on Canva.
<u>+ boolean isIllegalMove(double newPlayerXPos, double newPlayerYPos)</u>	Check the collision(Calling the GameScreen. <i>getInstance().isCollision</i> method) prevents the player from moving out of the scene.
<u>+ void startGame()</u>	Start the new game by setting everything to default state.
<u>+ void resetGame()</u>	Reset all states to default.
<u>+ void addScore(int value)</u>	add Score with value used to update value score which will used in ScoreBoard
<u>+ void decrementTime()</u>	each time this method is call (if time > 0) decrement it by one
Getters & Setters	getter and setter for field above

## 2.7 class KeyboardController - This class is used to check KeyboardInput

### 2.7.1 Fields

- boolean upPressed	State of W button press.
- boolean downPressed	State of S button press.
- boolean leftPressed	State of A button press.
- boolean rightPressed	State of D button press.
- boolean interactionTriggered	State of E button press.
- boolean escPressed	State of ESC button press.

### 2.7.2 Constructor

+ KeyboardController()	Call the keyboardChecker method.
------------------------	----------------------------------

### 2.7.3 Method

+ void keyboardChecker()	Set on key press event(For checking keyboard press). CASE W: setupPressed to true.
--------------------------	---

	CASE S: downPressed to true. CASE A: leftPressed to true. CASE D: rightPressed to true. CASE E: interactionTriggered to true. CASE ESC: escPressed to true. Set on key release event(For return default state to button when release button). CASE W: setupPressed to false. CASE S: downPressed to false. CASE A: leftPressed to false. CASE D: rightPressed to false. CASE E: interactionTriggered to false. CASE ESC: escPressed to false.
+ boolean isInteraction	check if interactionTriggered is true if is true change it to false (This is made so that Triggered only one time not pressed E continuous)
Getters & Setters	getter and setter for field above

## 2.8 class SoundManager - This is class is used to manage sound

### 2.8.1 Fields

- <u>AUDIOCLIP WALKSOUND</u>	Load walk sound.
- <u>AUDIOCLIP COOKSOUND</u>	Load cook sound.
- <u>AUDIOCLIP CHILLERSOUND</u>	Load chiller sound.
- <u>AUDIOCLIP SERVESOUND</u>	Load serve sound.
- <u>AUDIOCLIP MENUBGM</u>	Load menu theme sound.
- <u>AUDIOCLIP GAMEBGM</u>	Load game theme sound.
- <u>boolean isGameBgmPlaying</u>	Set default state to false.
- <u>boolean isMenuBgmPlaying</u>	Set default state to false.
- <u>double globalVolume</u>	Default sound volume of every sound is 50%

## 2.8.2 Methods

<u>+ void setGlobalVolume(double volume)</u>	globalVolume is between 0-1 setVolume of everyBGM/Sound to match globalVolume field
<u>+ void playWalk()</u>	Method that sets volume to global volume and plays walk sound.
<u>+ void stopPlayWalk()</u>	Method that stops playing walk sound.
<u>+ void playCook()</u>	Method that sets volume to global volume and plays cook sound.
<u>+ void stopCook()</u>	Method that stops playing cook sound.
<u>+ void playChillerOpen()</u>	Method that sets volume to global volume and plays interacting sound.
<u>+ void playServe()</u>	Method that sets volume to global volume and plays serving sound.
<u>+ void stopServe()</u>	Method that stops serving sound.
<u>+ void playMenuBGM()</u>	Method that sets volume to global volume and plays menu sound and checks if the sound is already played or not.
<u>+ void stopMenuBGM()</u>	Method that stops menu sound.
<u>+ void playGameBgm()</u>	Method that sets volume to global volume and plays game sound and checks if the sound is already played or not.
<u>+ void stopGameBGM()</u>	Method that stops game sound.

## 2.9 class StaticObject - This is blueprint for create Static Object instances

### 2.9.1 Fields

- String name	Name of the object.
- String imageUrl	Image url of the object.

- boolean isCollidable	Boolean flag indicating whether the object supports user collision
- boolean isInteractable	Boolean flag indicating whether the object supports user interaction, such as being picked up, opened, or activated.
- double x	xPosition of object (this position means position on the 2D array grid not on Canvas itself)
- double y	YPosition of object (this position means position on the 2D array grid not on Canvas itself)

## 2.9.2 Constructor

+ StaticObject(String name, String imageUrl, double x, double y, boolean isCollidable, boolean isInteractable)	Constructor for create general StaticObject
--	---

## 2.9.3 Method

Getters & Setters	getter and setter for field above
-------------------	-----------------------------------

## 2.10 StaticObjectDetail - This is a blueprint for keeping StaticObject Data

### 2.10.1 Fields

- String name	Name of the object.
- String staticImageUrl	Image url of the object.
- boolean isCollidable	State that check allowed collision.

- boolean isInteractable	State that check allowed interaction.
--------------------------	---------------------------------------

## 2.10.2 Methods

Getters & Setters	getter and setter for field above
-------------------	-----------------------------------

2.11 class StaticObjectFactory - This class is used to create StaticObject instances and classify them if they are of a special type (e.g., Stove, Sink, Chiller).

### 2.11.1 Method

<u>+ StaticObject</u> <u>createStaticObject(String name,</u> <u>StaticObjectDetail detail, double x,</u> <u>double y)</u>	This method is used to create and return a StaticObject using the specified name and its associated detail parameters.
--	--

2.12 class Task - This class is blueprint to keep each task(menu that need to be served in order to obtain score) detail

### 2.12.1 Field

- Menu expectedMenu	The menu that needs to be delivered.
- Class<? extends Container> requiredContainer	The required container of the menu.
- int score	Score after serving the menu.

### 2.12.2 Constructor

+ Task(Menu expectedMenu, Class<? extends Container> requiredContainer, int score)	Set the menu, the required container and score of the menu.
--	---



### 2.12.3 Methods

+ boolean isCompletedBy(Heldable held)	The method that checks the correction of name, required menu and required container.
Getters Method	getter for field above

2.13 class TaskManager - This class to used to manage Task cycle (random generate,finish)

#### 2.13.1 Field

- <u>Task currentTask</u>	The current task.
- <u>boolean taskGenerated</u>	The state checking generated task.

#### 2.13.2 Method

+ <u>void generateTask()</u>	Check if the task has been generated yet. If not random the menu from the list "Ramen", "Soup", "FriedPork", "SalmonSushi", "Omurice", "TunaSushi", "Rice", "Pancake", "Bread", the required container and score of the menu.
+ <u>Task getCurrentTask()</u>	Return the current task.
+ <u>void finishCurrentTask()</u>	If finished the task, set the current task to null and taskGenerated to false.
- <u>Class&lt;? extends Container&gt; getRequiredContainerFor(String menuName)</u>	Return the required container of the menus.
- <u>getScoreForMenu(String menuName)</u>	Return the score of the menus.

## 3. Package GUI

3.1 Abstract Class BaseSceneScreen extends Canvas -  
This class is a base to all SceneScreen (that show on player pov)

### 3.1.1 Fields

# GraphicsContext gc	gc for inherited class can use
# boolean isActive	boolean that show that scene is active or not
# final Image defaultBackgroundImage	this image is obtain from this URL image/basescene/pixel_art_kitchen_480x270.png (By using Scenemanager.imageLoader method)

### 3.1.2 Constructor

+ BaseSceneScreen()	- Set width by Canva width(get from SceneManager) - Set height by Canva height(get from SceneManager)
---------------------	--

### 3.1.3 Method

+ void render(GraphicsContext gc)	used to render in inherited class
+ void onEnter()	method used in inherited class to do what when enter that scene
+ void reset()	method used in inherited class to reset the scene to initial state
+ void clear()	method is used to clear the canvas
+ void drawDefaultBackground	drawImage of defaultBackground at 0 0 with size of Canvas Width and Canvas Height
+ boolean isActive()	getter for isActive
+ void setActive(boolean active)	setter for isActive

## 3.2 Class ResultScreen extends BaseSceneScreen -

This class is used to build a result screen when game is ended

### 3.2.1 Fields

- VBox layout	Arranges UI components vertically with spacing.
- Scene resultScene	Holds the layout and background image, shown in the JavaFX window.
- Label scoreLabel	Displays the final score when the result screen appears.
- Image backgroundImage	Hold the background image.

### 3.2.2 Constructor

+ ResultScreen()	Initialize the result screen by showing timed up text, also contains a button that can return to the main menu or exit the game.(This use stackpane that has children is Canva and VBox Layout)
------------------	---

### 3.2.3 Method

+ void onEnter()	Called when the result screen is shown.Updates the score label with the final score. Sets the current scene to resultScene.
+ void render(GraphicsContext gc)	Do nothing
+public void reset()	Do nothing

## 3.3 Class PauseScreen extends BaseSceneScreen -

This class is used to create a pause screen when player pressed ESC

### 3.3.1 Field

- VBox buttonBox	buttonBox represent VBox pane
------------------	-------------------------------

### 3.3.2 Constructor

+ PauseScreen()	Creates VBox pane and contains resume button, main menu button, new game button.
-----------------	--

### 3.3.3 Method

+ void onEnter()	Call the render method and add a button.
+ void render(GraphicsContext gc)	Draw the game pause scene.
+ void reset()	Remove the VBox pane.

## 3.4 Class CookingWaitScene extends BaseSceneScreen

### 3.4.1 Fields

- double progress	Set the default progress to 0.0.
- Timeline progressTimeline	progressTimeline represents timeline.

### 3.4.2 Method

+ void onEnter()	Create the cooking time bar. After 5 seconds return the cooked ingredient.
+ void render(GraphicsContext gc)	Draw the waiting screen.
+ void reset()	Reset the time bar.

## 3.5 Class ContainerSelectScene extends BaseSceneScreen

### 3.5.1 Fields

- <u>int ICON_SIZE</u>	Set the size of icon to 80
------------------------	----------------------------

<u>- double POT_X</u>	Set the position of the pot in the x-axis to 130.
<u>- double POT_Y</u>	Set the position of the pot in the y-axis to 60.
<u>- double PAN_X</u>	Set the position of the pan in the x-axis to 270.
<u>- double PAN_Y</u>	Set the position of the pam in the y-axis to 60.
<u>- double DISH_X</u>	Set the position of the dish in the x-axis to 130.
<u>- double DISH_Y</u>	Set the position of the pot in the y-axis to 160.
<u>- double BOWL_X</u>	Set the position of the bowl in the x-axis to 270.
<u>- double BOWL_Y</u>	Set the position of the bowl in the y-axis to 160.

### 3.5.2 Constructor

+ ContainerSelectScene()	Set on mouse click.
--------------------------	---------------------

### 3.5.3 Method

- void handleMouseClicked(MouseEvent e)	If clicking at Pot, Pan, DISH, BOWL, the player will get an item into hand. Stop game sound.
+ void render(GraphicsContext gc)	Draw the image of the table.
+ void onEnter()	Print "Entered ContainerSelectScene".
+ void reset()	Print "Reset ContainerSelectScene"
- boolean isInside(double x, double y, double rectX, double rectY, double width, double height)	Check if it is inside the square.

## 3.6 Class ChillerScene extends BaseSceneScreen

### 3.6.1 Constructor

+ ChillerScene()	Add mouse listener detecting click.
------------------	-------------------------------------

### 3.6.2 Method

+ void update(GraphicsContext gc)	-
+ void render(GraphicsContext gc)	Clear Canvas and draw a chiller image.
+ void onEnter()	Render this canvas and SetActive to true.
+ void reset()	SetActive to false.
- void handleClick(MouseEvent event)	Check the clicked ingredient and setHeldItem of the player according to player clicks and close the scene.
- String getClickedIngredient(double x, double y)	Create the box containing the ingredients. Check the position of the mouse, click which one and return the ingredient.

## 3.7 class MainMenuScreen extends BaseSceneScreen

### 3.7.1 Field

+ double startX	The x-coordinate (left edge) of the "Start" button. Set to 169.
+ double startY	The y-coordinate (top edge) of the "Start" button. Set to 86.
+ double startWidth	The width of the button. Set to 141.
+ double startHeight	The height of the button. Set to 18.

### 3.7.2 Method

+ void onEnter()	Draw the image of mainMenuScreen.png. Set mouse on click by 3 choices (Start, Option,
------------------	---

	Exit button). When the mouse enters the area of the button, change the cursor to HAND.
+ void render(GraphicsContext gc)	-
+ void reset()	-

## 3.8 class GameScreen extends BaseSceneScreen

### 3.8.1 Fields

- ArrayList<AnimatedObject> dynamicObject	Array that collects dynamic object
- ArrayList<ArrayList<String>> tiles	2D Array that collects Tile Object (keep in Column and Row form)
- ArrayList<ArrayList<String>> objects	2D Array that collects General Objects (keep in Column and Row form)
-HashMap<String,StaticObjectDetail> > staticObjectsDetail	Hashmap that use to quickly search that static object detail by name
- HashMap<String, DynamicObjectDetail> dynamicObjectsDetail	Hashmap that use to quickly search that dynamic object detail by name
- HashMap<String, Image> staticImageCache	Hashmap that is used to keep static object image Cache to prevent lag from repeatedly downloading to draw scene
- HashMap<String, AnimatedObject> dynamicObjectCache	Hashmap that is used to keep dynamic object image Cache to prevent lag from repeatedly downloading to draw scene
- MapData mapData	mapdata blueprint that used in extracting data
- Interactable hitObject	general object that can be Interactable
- String playerFeedbackText	A text that send to player at center screen to show status ex.cannot interact or complete order



- long feedbackTimestamp	default timeStamp used in calculating time passed in feedBackText
- final long FEEDBACK_DURATION	A duration which text stay on the screen
- GameScreen instance	A singleton instance of GameScreen
+ Gson gson	Gson library use to extract data from json file

### 3.8.2 Constructor

- GameScreen()	Call super() method.
----------------	----------------------

### 3.8.3 Method

+ void addObject()	This method extract json data from map//map1.json and keep in mapData , tiles , objects, staticObjectsDetail,dynamicObjectsDetail
+ void preloadStaticImages()	this method loop through staticObjectsDetail HashMap keyset add pre download image object to staticImageCache
+ void preloadStaticObjects()	for loop through objects 2D Array it will load only static object by adding StaticObject instance that create from StaticObjectFactory in staticObject field
+ void preloadDynamicObjects()	for loop through objects 2D Array it will load only dynamic object by adding AnimatedObject instance that create from DynamicObjectFactory in dynamicObjectCache field
+ void drawTiles(GraphicsContext gc)	draw tile by for loop through tiles array (draw position from col and row from array that must be convert by GameManager.positionX / Y )

+ void drawStaticObjects(GraphicsContext gc)	draw StaticObject by for loop through tiles array (draw position from col and row from array that must be convert by GameManager.positionX / Y ) ** it will check if it have container on that object (for Counter and RollingMat) if have it will draw a stroke to show that there is object on that object
+ void drawDynamicObjects(GraphicsContext gc)	draw dynamic object from dynamicObjectCache hashmap value (draw position from col and row from array that must be convert by GameManager.positionX / Y )
+ void addDynamicObject()	for loop through mapData.getDynamicObjectsDetail.keySet() and add AnimatedObject that is create by DynamicObjectFactory.createDynamicObject to dynamicObject field
+ boolean isCollision(double targetX, double targetY)	check collision by using AABB technique
+ void setFeedback(String text)	setFeedbackText and set new feedbackTimeStamp to time that this text has been set
- void drawCenteredLine(GraphicsContext gc, String line, double y, double maxWidth)	draw centered text on Canvas
- void drawWrappedText(GraphicsContext gc, String text, double yStart, double maxWidth)	draw text if text graphic length is more than canvas create a new sentence
+ void render(GraphicsContext gc)	drawTile,staticObjects,DynamicObjects,setOrange to Toppading and set Feedbacktext it will disappear if time pass more than time limit of a text
+ void onEnter()	preloadStaticObjects

+ void reset()	if it is counter setContainerOnCounter null if it is RollingMat set it to null and random new Task
+ <u>GameScreen getInstance()</u>	if there is no object of GameScreen create a new one else return that object
Getters & Setters	Getter and Setter for all field

### 3.9 class OptionMenuScreen extends BaseSceneScreen

#### 3.9.1 Field

- Scene optionScene	Store option scene.
---------------------	---------------------

#### 3.9.2 Constructor

+ OptionMenuScreen()	Create the scene, volume bar to adjust the sound in the game and the button switch to recipe scene or back to main menu.
----------------------	---

#### 3.9.3 Methods

+ void onEnter()	Activate a scene when this method gets called.
+ void render(GraphicsContext gc)	Render the scene.
+ void reset()	-

### 3.10 class RecipeCollectionScreen extends BaseSceneScreen

#### 3.10.1 Fields

- Scene recipeScene	Store recipe scene.
- Canvas canvas	Store the canvas
- GraphicsContext gc	Named gc of type GraphicsContext

- DOUBLE LINEHEIGHT	Vertical space (in pixels) between each line of recipe text.
- DOUBLE MARGIN	Top and bottom padding from the edge of the canvas
- List<String> recipeLines	Array for recipe text.
- double scrollOffsetY	This variable controls how far the list of recipes is scrolled vertically.

### 3.10.2 Constructor

+ RecipeCollectionScreen()	Prepare the recipe text in the recipeLines. Create canvas and scene and also allow the mouse scroll. Add back button to the scene.
----------------------------	--

### 3.10.3 Method

- void clampScroll()	Limit the scroll in the vertical.
- void render()	Draw the scene, add the text and scroll.
+ void onEnter()	When calling this method, set this scene.
+ void render(GraphicsContext gc)	call render method.
+ void reset()	Reset the scrollOffsetY and render again.

## 3.11 class SceneManager extends Canvas

### 3.11.1 Fields

- <u>STRING ICONURL</u>	Icon file path.
- <u>STRING GAMENAME</u>	The game name.
- <u>int canvaWidth</u>	Default canvas width.
- <u>int canvaHeight</u>	Default canvas height.
- <u>INT TOPPADDING</u>	Top padding size.

- <u>INT PIXELSIZE</u>	Pixel size of all images.
- <u>INT TOTALHORIZONTALTILES</u>	Number of horizontal tiles.
- <u>INT TOTALVERTICALTILES</u>	Number of vertical tiles.
- <u>KeyboardController</u> <u>keyboardChecker</u>	Keyboard checker for checking keyboard button press.
- <u>Stage primaryStage</u>	Stores the main Stage.
- <u>Scene presentScene</u>	Track of the current scene being displayed on the primaryStage.
- <u>MAINMENUSCREEN</u> <u>MENUSCREEN</u>	Store main menu screen scene.
- <u>CHILLERSCENE CHILLERSCENE</u>	Store chiller screen scene.
- <u>CONTAINERSELECTSCENE</u> <u>CONTAINERSCENE</u>	Store container selector scene.
- <u>AnimationTimer playerMoveTimer</u>	An instance of that timer, specifically for handling player movement, interaction, and rendering
- <u>COOKINGWAITSCENE</u> <u>COOKINGWAITSCENE</u>	Store cooking scene.
- <u>PAUSESCREEN PAUSESCREEN</u>	Store pause screen scene.
- <u>RESULTSCREEN</u> <u>RESULTSCREEN</u>	Store result screen scene.
- <u>Thread gameTimerThread</u>	Background thread that handles the game's countdown timer.
- <u>boolean gameTimerStarted</u>	State that checks the activity of the timer.

- <u>OptionMenuScreen</u> <u>optionMenuScreen</u>	Store menu screen scene.
- <u>RecipeCollectionScreen</u> <u>recipeCollectionScreen</u>	Store recipe collection scene.

### 3.11.2 Methods

+ void initialize(Stage stage)	Initialize the game screen, add menu screen and preload the object, random menu in the game scene . Also set the window game icon and title. Start the menu sound.
+ void <u>switchToMenuScreen()</u>	Stop both game and timer thread after that switch to menu screen. Also stop all the sounds.
+ void <u>switchToPauseScreen()</u>	Stop both game and timer thread after that switch to pause screen. Also stop all the sounds.
+ void <u>switchToOptionMenuScreen()</u>	Switch to the option menu screen.
+ void <u>switchToRecipeCollectionScreen()</u>	Switch to the recipe collection screen.
+ void <u>switchToGameScreen()</u>	Switch to game screen, activate generate task, set the present game scene, render player, activate the keyboard, activate the player thread, start game timer and start game sound. (Disable other sounds prevent wrong sound while changing scene)
+ void <u>switchToChillerScene()</u>	Stop both game and timer thread (Ignore the picking time from the game to make the game more easier.) after that switch to chiller screen.
+ void <u>switchToCookingWaitScene()</u>	Stop both game and timer thread (Ignore the cooking time from the game to make the game more easier.) after that switch to cooking screen.
+ void <u>switchToResultScreen()</u>	Stop player thread and disable game sound. Switch to the result screen.
+ Image <u>imageLoader(String url)</u>	Load the image by using the url string.
+ void <u>switchToContainerSelectScene()</u>	Stop the player thread and switch to container select scene.

+ <u>void updatePlayerMove()</u>	Using as the player thread that keeps tracking the player input keyboard and keeping render player direction. <b>Also contains polymorphism at the interaction button and plays the interaction sound.</b>
+ <u>void startGameTimer(GraphicsContext gc)</u>	Start the game timer counting down. Also contains a method that pauses the timer.
+ <u>void drawOverlay(GraphicsContext gc)</u>	Drawing the text that appears on the game screen (Timer, Score, Hand status and order).
- <u>void stopGameTimer()</u>	Stop the game timer when switching to some scenes.
- Getters & Setters	Getter and setter for all fields

## 4.Package ingredients

### 4.1 class Egg extends Ingredient

#### 4.1.1 Field

- <u>String NAME</u>	Set the string Name to "Egg"
----------------------	------------------------------

#### 4.1.2 Constructor

+ Egg()	Set the name of the object.
---------	-----------------------------

#### 4.1.3 Method

+ boolean isPrepared()	Return true (No need to be prepared).
------------------------	---------------------------------------

### 4.2 class Flour extends Ingredient

#### 4.2.1 Field

- <u>String NAME</u>	Set the string Name to "Flour"
----------------------	--------------------------------



### 4.2.2 Constructor

+ Flour()	Set the name of the object.
-----------	-----------------------------

### 4.2.3 Method

+ boolean isPrepared()	Return true (No need to be prepared).
------------------------	---------------------------------------

## 4.3 abstract class Ingredient implements Heldable

### 4.3.1 Field

- String name	Name of the object.
---------------	---------------------

### 4.3.2 Constructor

+ Ingredient(String name)	Set the name of the ingredient.
---------------------------	---------------------------------

### 4.3.3 Method

+ abstract boolean isPrepared()	Tell the state of the ingredient.
+ String getName()	Get the name of the object.
+ void setName(String name)	Set the name of the object.

## 4.4 class IngredientFactory

### 4.4.1 Field

- <u>Map&lt;String, Supplier&lt;Heldable&gt;&gt; registry</u>	Map that keep string key and Supplier to create that Ingredient object when use string key
---	--

### 4.4.2 Method

+ static	static Initializer for registry field
+ static Heldable createIngredient(String name)	create ingredient by obtain object ingredient from registry.get (in Supplier)

## 4.5 class Miso extends Ingredient

### 4.5.1 Field

- <u>String NAME</u>	Set the string Name to "Miso".
----------------------	--------------------------------

### 4.5.2 Constructor

+ Miso()	Set the name of the object.
----------	-----------------------------

### 4.5.3 Method

+ boolean isPrepared()	Return true (No need to be prepared).
------------------------	---------------------------------------

## 4.6 class Noodle extends Ingredient

### 4.6.1 Field

- <u>String NAME</u>	Set the string Name to "Noodle".
----------------------	----------------------------------

### 4.6.2 Constructor

+ Noodle()	Set the name of the object.
------------	-----------------------------

### 4.6.3 Method

+ boolean isPrepared()	Return true (No need to be prepared).
------------------------	---------------------------------------

## 4.7 class Pork extends Ingredient implements Choppable

### 4.7.1 Field

- <u>String NAME_RAW</u>	Set the default name to "Pork (Unchopped)".
- <u>String NAME_CHOPPED</u>	Set the name after chop to "Pork".
- boolean chopped	Default chop state to false.

### 4.7.2 Constructor

+ Pork()	Set the name of the object to default.
----------	--

### 4.7.3 Method

+ void chop()	If the pork hasn't been chopped, setChopped to true and change name to chopped state.
+ boolean isPrepared()	The method that checks preparation.
+ boolean isChopped()	Check the chopping state.
+ setChopped(boolean chopped)	Set chopped state.
+ String getName()	Get the name of the object.
+ setName(String name)	Set the name of the object.

## 4.8 class RawRice extends Ingredient

### 4.8.1 Field

- <u>String NAME_RAW</u>	Set the default name to "Pork (Unprepared)".
--------------------------	--

### 4.8.2 Constructor

+ RawRice()	Set the name of the object to default.
-------------	--

### 4.8.3 Method

+ boolean isPrepared()	Return false (Need to be prepared).
------------------------	-------------------------------------

## 4.9 class Salmon extends Ingredient implements Choppable

### 4.9.1 Field

- <u>String NAME_RAW</u>	Set the default name to "Salmon (Unchopped)".
- <u>String NAME_CHOPPED</u>	Set the name after chop to "Salmon".
+ boolean chopped	Default chop state to false.

## 4.9.2 Constructor

+ Salmon()	Set the name of the object to default.
------------	--

## 4.9.3 Method

+ void chop()	If the salmon hasn't been chopped, setChopped to true and change name to chopped state.
+ boolean isPrepared()	The method that checks preparation.
+ boolean isChopped()	The method that checks chop state.
+ setChopped(boolean chopped)	Set the chop state.
+ String getName()	Get the name of the object.
+ setName(String name)	Set the name of the object.

## 4.10 class Seaweed extends Ingredient implements Choppable, Washable

### 4.10.1 Field

- <u>String NAME_RAW</u>	The default name of seaweed that hasn't been chopped or washed.
- <u>String NAME_WASHED</u>	The name of seaweed after washing.
- <u>String NAME_CHOPPED</u>	The name of the seaweed after chopping.
- boolean chopped	State that check chop.
- boolean washed	State that check wash.

### 4.10.2 Constructor

+ Seaweed()	Set the default name to raw state.
-------------	------------------------------------

### 4.10.3 Method

+ boolean isPrepared()	The method that checks preparation.
+ void chop()	If Seaweed hasn't been washed, the

	player has to wash it first. If Seaweed hasn't been chopped, setChopped to true.
+ void wash()	If Seaweed hasn't been washed, set the seaweed state setWashed to true.
+ boolean isWashed()	The wash state.
+ void setWashed(boolean washed)	Set the wash state.
+ String getName()	Get the name of the object.
+ void setName(String name)	Set the name of the object.
+ boolean isChopped()	The check chop state method.
+ void setChopped(boolean chopped)	Set chop state method.

## 4.11 class Sugar extends Ingredient

### 4.11.1 Field

- <u>String NAME_RAW</u>	Default name to "Sugar"
--------------------------	-------------------------

### 4.11.2 Constructor

+ Sugar()	Set the name to the default name.
-----------	-----------------------------------

### 4.11.3 Method

+ boolean isPrepared()	Set the prepared state to true.
------------------------	---------------------------------

## 4.12 class Tuna extends Ingredient implements Choppable

### 4.12.1 Field

- <u>String NAME_RAW</u>	Default name to "Tuna (Unchopped)"
- <u>String NAME_CHOPPED</u>	Set the name after chop to "Tuna".
- boolean chopped	Default chop state to false.

## 4.12.2 Constructor

+ Tuna()	Set the name to the default name.
----------	-----------------------------------

## 4.12.3 Method

+ void chop()	If Tuna has not been chopped, setChopped to true and setName to NAME_CHOPPED
+ boolean isPrepared()	Set the prepared state according to isChopped.
+ boolean isChopped()	Get chop state.
+ void setChopped(boolean chopped)	Set chop state.
+ String getName()	Get name.
+ void setName(String name)	Set name.

## 4.13 class Wagame extends Ingredient implements Choppable,Washable

### 4.13.1 Field

- <u>String NAME_RAW</u>	Default name to "Wagame (Unchopped,Unwashed)"
- <u>String NAME_WASHED</u>	Set the name after wash to "Wagame (Unchopped)".
- <u>String NAME_CHOPPED</u>	Set the name after chop to "Wagame".
- boolean chopped	Default chop state to false.
- boolean washed	Default wash state to false.

### 4.13.2 Constructor

+ Wagame()	Set the name to the default name.
------------	-----------------------------------

### 4.13.3 Method

+ boolean isPrepared()	Set the state of preparation according to isChopped.
+ void chop()	If the wagame hasn't been washed, it can't be chopped. After wash setChopped to true and setName to NAME_CHOPPED
+ void wash()	setWashed to true and set name to NAME_WASHED.
+ boolean isWashed()	The wash state.
+ void setWashed(boolean washed)	Set the wash state.
+ String getName()	Get the name.
+ void setName(String name)	Set the name.
+ boolean isChopped()	Get the chop state.
+ void setChopped(boolean chopped)	Set the chop state.

## 4.14 class Water extends Ingredient

### 4.14.1 Field

- <u>String NAME</u>	Default name to "Water"
----------------------	-------------------------

### 4.14.2 Constructor

+ Water()	Set the name to the default name.
-----------	-----------------------------------

### 4.14.3 Method

+ boolean isPrepared()	Return true (No need to be prepared).
------------------------	---------------------------------------

## 5 Package interfaces

### 5.1 interface Choppable

#### 5.1.1 Method

+ void chop()	Chop method.
---------------	--------------

+ boolean isChopped()	Chop state method.
-----------------------	--------------------

## 5.2 interface Cookable

### 5.2.1 Method

+ void cook(CookMethod method)	Cook method.
+ boolean isCooked(CookMethod method)	Cook state method.
+ void forceCook()	Force cook method.

## 5.3 enum CookMethod

BOIL	Represent the boiling method.
FRY	Represent the frying method.

## 5.3 interface Heldable

### 5.3.1 Method

+ String getName()	Get name method.
--------------------	------------------

## 5.4 interface Interactable

### 5.4.1 Method

+ void interact()	Interact method.
-------------------	------------------

## 5.5 interface Washable

### 5.5.1 Method

+ void wash()	Wash method.
+ boolean isWashed()	Wash state method.

## 5.6 Enum CookMethod

BOIL	Represent the boil process.
FRY	Represent the fry process.



## 6. Package Menu

### 6.1 class Bread extends Menu implements Cookable

#### 6.1.1 Fields

- <u>String NAME_UNCOOKED</u>	Default name of bread to "Bread (Uncooked)".
- <u>String NAME_COOKED</u>	Name after cooking process "Bread".
- boolean cooked	Default cook state to false.

#### 6.1.2 Constructor

+ Bread()	Set name and menu recipe.Using MenuRecipe.ALL_RECIPES.get("Bread").getIngredients()
-----------	---

#### 6.1.3 Method

+ void cook(CookMethod method)	Set Cooked to true and setName to NAME_COOKED
+ boolean isCooked(CookMethod method)	Method that returns cook state.
+ boolean isUsableAsIngredient()	Set as false. (Bread cannot be merged other menus)

### 6.2 class FriedPork extends Menu implements Cookable

#### 6.2.1 Fields

- <u>String NAME_UNCOOKED</u>	Default name of pork when in pan to "FriedPork (Uncooked)"
- <u>String NAME_COOKED</u>	Name after cooking process "FriedPork".
+ boolean cooked	Default cook state to false.

#### 6.2.2 Constructor

+ FriedPork()	Set name and menu recipe.Using
---------------	--------------------------------

	MenuRecipe.ALL_RECIPES.get("Fried Pork").getIngredients()
--	---

### 6.2.3 Method

+ void cook(CookMethod method)	Set Cooked to true and setName to NAME_COOKED
+ boolean isCooked(CookMethod method)	Method that returns cook state.
+ boolean isUsableAsIngredient()	Set as true. (Can be merged with another menu)
+ void setCooked(boolean cooked)	Set cooked.

## 6.3 class Menu implements Heldable

### 6.3.1 Fields

- HASHMAP<STRING,INTEGER> INGREDIENTREQUIRED	HashMap of required ingredients.
- String name	Name of the object.

### 6.3.2 Constructor

+ Menu(String name,HashMap<String,Integer> ingredientRequired)	Set name and required ingredients.
--	------------------------------------

### 6.3.3 Method

+ String getName()	Get name.
+ void setName(String name)	Set name.
+ HashMap<String, Integer> getIngredientRequired()	Hashmap that stores required ingredients.
+ <i>boolean isUsableAsIngredient()</i>	State that checks merging with other menus.

## 6.4 class MenuFactory

### 6.4.1 Fields

<u>- HASHMAP&lt;STRING, CLASS&lt;? EXTENDS MENU&gt;&gt; MENU_MAP</u>	Menu_map by HashMap containing string name and class.
--	---

## 6.4.2 Construction

static	Add all of these to HashMap. ("Soup", Soup.class), ("SalmonSushi", SalmonSushi.class), ("TunaSushi", TunaSushi.class), ("Ramen", Ramen.class), ("Omurice", Omurice.class), ("Pancake", Pancake.class), ("Bread", Bread.class), ("Rice", Rice.class), ("FriedPork", FriedPork.class).
--------	---

## 6.4.3 Method

<u>+ Menu createMenu(String menuName)</u>	Create Menu from name by using menu_map.get(menuName).getDeclaredConstructor().newInstance();
<u>+Ingredient createIngredientFromMenu(Menu menu)</u>	Create Ingredient from menu that is Usable as Ingredient (Override isPrepared to true)

## 6.5 class MenuRecipe

### 6.5.1 Field

<u>- HASHMAP&lt;STRING, RECIPE&gt; ALL_RECIPES</u>	Menu_map by HashMap containing string menu name and menu class.
--	---

### 6.5.2 Constructor

static	Add all names and recipes to HashMap. (friedPorkRecipe, soupRecipe, cookedRiceRecipe, salmonSushiRecipe, tunaSushiRecipe, ramenRecipe, omuriceRecipe, pancakeRecipe, breadRecipe).
--------	--

## 6.6 class MergeMenu

### 6.6.1 Method

+ <u>String</u> <u>findMatchingMenu(ArrayList&lt;Ingredient&gt; inputIngredients, Class&lt;? extends Container&gt; inputContainer)</u>	Counts how many times each ingredient (by name) appears in the container. Loop through all recipes. Check Container Match. Check Ingredients Match. If matched returns the menu name ,else return null.
---	---

## 6.7 class Omurice extends Menu implements Cookable

### 6.7.1 Field

- <u>STRING NAME_UNCOOKED</u>	Default name to "Omurice (Uncooked)".
- <u>STRING NAME_COOKED</u>	Name after cooking "Omurice".
- boolean cooked	Default cook state to false.

### 6.7.2 Constructor

+ Omurice()	Set name and menu recipe. Using MenuRecipe.ALL_RECIPES.get("Omurice").getIngredients()
-------------	--

### 6.7.3 Method

+ void cook(CookMethod method)	Set name to NAME_COOKED and cooked to true.
+ boolean isCooked(CookMethod method)	Set cook state.
+ boolean isUsableAsIngredient()	Set to false. (Can not be merged with other menus.)

## 6.8 class Pancake extends Menu implements Cookable

### 6.8.1 Fields

- <u>STRING NAME_UNCOOKED</u>	Default name is "Pancake (Uncooked)".
- <u>STRING NAME_COOKED</u>	Name after cooking "Pancake".

- boolean cooked	Default cook state to false.
------------------	------------------------------

## 6.8.2 Constructor

+ Pancake()	Set name and menu recipe.Using MenuRecipe.ALL_RECIPES.get("Pancake").getIngredients()
-------------	---

## 6.8.3 Method

+ void cook(CookMethod method)	Set name to NAME_COOKED and cooked to true.
+ boolean isCooked(CookMethod method)	Check cook state.
+ boolean isUsableAsIngredient()	Set to false. (Can not be merged with other menus.)

## 6.9 class Ramen extends Menu

### 6.9.1 Field

- <u>STRING NAME</u>	Default name is "Ramen".
----------------------	--------------------------

### 6.9.2 Constructor

+ Ramen()	Set name and menu recipe.Using MenuRecipe.ALL_RECIPES.get(NAME).getIngredients()
-----------	--

### 6.9.3 Method

+ boolean isUsableAsIngredient()	Set to false. (Can not be merged with other menus.)
----------------------------------	---

## 6.10 class Recipe

### 6.10.1 Field

- HashMap<String, Integer> ingredients	HashMap that collects the name and number of ingredients.
- Class<? extends Container>	Classes represent the used

containerUsed	containers.
---------------	-------------

## 6.10.2 Constructor

+ Recipe(HashMap<String, Integer> ingredients, Class<? extends Container> containerUsed)	Set ingredients and containers.
--	---------------------------------

## 6.10.3 Method

+ HashMap<String, Integer> getIngredients()	Return HashMap of ingredients.
+ Class<? extends Container> getContainerUsed()	Return the used container class.

# 6.11 class Rice extends Menu implements Cookable

## 6.11.1 Field

- <u>STRING NAME_UNCOOKED</u>	Default name is "Rice (Uncooked)".
- <u>STRING NAME_COOKED</u>	Name after cooking "Rice".
- boolean cooked	Default cook state to false.

## 6.11.2 Constructor

+ Rice()	Set name and menu recipe.Using MenuRecipe.ALL_RECIPES.get("Rice").getIngredients()
----------	--

## 6.11.3 Method

+ void cook(CookMethod method)	Set name to NAME_COOKED and cooked to true.
+ boolean isCooked(CookMethod method)	Check cook state.
+ boolean isUsableAsIngredient()	Set to true. (Can be merged with other menus.)
+ void setCooked(boolean cooked)	Set cook state.

## 6.12 class SalmonSushi extends Menu

### 6.12.1 Field

- <u>STRING NAME</u>	Default name is "SalmonSushi".
----------------------	--------------------------------

### 6.12.2 Constructor

+ SalmonSushi()	Set name and menu recipe.Using MenuRecipe.ALL_RECIPES.get(NAME).getIngredients()
-----------------	--

### 6.12.3 Method

+ boolean isUsableAsIngredient()	Set to false. (Can not be merged with other menus.)
----------------------------------	---

## 6.13 class Soup extends Menu implements Cookable

### 6.13.1 Fields

- <u>STRING NAME_UNCOOKED</u>	Default name is "Soup (Uncooked)".
- <u>STRING NAME_COOKED</u>	Name after cooking "Soup".
- boolean cooked	Default cook state to false.

### 6.13.2 Constructor

+ Soup()	Set name and menu recipe.Using MenuRecipe.ALL_RECIPES.get("Soup").getIngredients()
----------	--

### 6.13.3 Methods

+ void cook(CookMethod method)	Set name to NAME_COOKED and cooked to true by checking method condition.
+ boolean isCooked(CookMethod method)	Check cook state.
+ void setCooked(boolean cooked)	Set cook state.
+ boolean isUsableAsIngredient()	Set to true. (Can be merged with other menus.)

## 6.14 class TunaSushi extends Menu

### 6.14.1 Field

- <u>STRING NAME</u>	Default name is "TunaSushi".
----------------------	------------------------------

### 6.14.2 Constructor

+ TunaSushi()	Set name and menu recipe.Using MenuRecipe.ALL_RECIPES.get(NAME).getIngredients()
---------------	--

### 6.14.3 Method

+ boolean isUsableAsIngredient()	Set to false. (Can not be merged with other menus.)
----------------------------------	---

## 7. Package container

### 7.1 class Bowl extends Container

#### 7.1.1 Methods

+ boolean canProcess()	Default to false.
+ void process()	-
# void updateName()	Update name when put menu in.
+ void setMenu(Menu menu)	Set the menu name with the bowl.
+ String getName()	Get the name of all bowl's states.

### 7.2 class Chiller extends AnimatedObject

#### 7.2.1 Constructor

+ Chiller(String name, String[] imageFrameURL, double x, double y, boolean isCollidable, boolean isInteractable)	Set name, frame, position, coliable and interactable state.
--	---

#### 7.2.2 Method



+ void interact()	Switch to Chiller Scene.
-------------------	--------------------------

## 7.3 class Container implements Heldable

### 7.3.1 Fields

# String name	Display name according to state.
# ArrayList<Ingredient> ingredients	Ingredients array collect ingredients.
# Menu menu	Menu after process.
# double x, y	Position of container.
+ boolean canProcess()	State that check allowed process.
+ void process()	Process the food. (boil, fry)

### 7.3.2 Method

+ void addIngredient(Ingredient ing)	Add the ingredient into arraylist and update name.
+ void removeContent()	Clear all state of container.
# void updateName()	Update name according to state.
+ boolean isEmpty()	Check the empty container.
+ Menu getMenu()	Return menu.
Getters & Setters	-

## 7.4 class Counter extends StaticObject implements Interactable

### 7.4.1 Field

- Container containerOnCounter	Default state to null.
--------------------------------	------------------------

### 7.4.2 Constructor

+ Counter(String name, String staticImageUrl, Double xPosition, Double yPosition, boolean	Set name, image url, position, collidable and interactable.
---	---

isCollidable, boolean isInteractable)	
---------------------------------------	--

### 7.4.3 Methods

+ void interact()	CASE 1: Player is holding nothing, setHeld item to the container. CASE 2: Player is holding an Ingredient, add the ingredient into the container. CASE 3: Player is holding a Container then transfers the menu to another container. CASE4: Player is holding something else, return log message.
- void mergeContainer(Container from, Container to)	Check the allowed merge between two containers.
Getters & Setters	-

## 7.5 class CounterWithChoppingBoard extends StaticObject implements Interactable

### 7.5.1 Constructor

+ CounterWithChoppingBoard(String name, String imageUrl, double x, double y, boolean isCollidable, boolean isInteractable)	Set name, image url, position, collidable and interactable state.
--	---

### 7.5.2 Method

+ void interact()	Chop the HeldItem if it can be chopped.
-------------------	---

## 7.6 class Dish extends Container

### 7.6.1 Methods

+ boolean canProcess()	Default to false.
+ void process()	-
# protected void updateName()	Merge menu and dish.

+ void setMenu(Menu menu)	Set menu by calling updateName.
+ String getName()	Get the name of the dish.

## 7.7 class FoodDeliveryChannel extends StaticObject implements Interactable

### 7.7.1 Constructor

+ FoodDeliveryChannel(String name, String imageUrl, double x, double y, boolean isCollidable, boolean isInteractable)	Set name, image url, position, collidable and interactable state.
---	---

### 7.7.2 Method

+ void interact()	Check if the HeldItem of the player matches the order or not after that update score and setHeld to null.
-------------------	---

## 7.8 class Pan extends Container

### 7.8.1 Methods

+ boolean canProcess()	Check mergeable state.
+ void addIngredient(Ingredient ingredient)	Check if the ingredient is allowed to fry after that call process method.
+ void process()	Process the ingredients and update the name.
# void updateName()	Change the name of the pan according to the ingredient in it.
+ String getName()	Get the pan name.

## 7.9 class Pot extends Container

### 7.9.1 Field

- Image potImage	Use SceneManager.imageLoader to load the image url.
------------------	---

## 7.9.2 Methods

+ boolean canProcess()	Check if ingredients can be merged to the menu.
+ addIngredient(Ingredient ingredient)	Add the ingredients into the pan.
+ void process()	Call updateName method to merge the pot and ingredient.
# void updateName()	Set the name of the pot matching the state.
+ String getName()	Get the name of merging ingredients inside the pot.

## 7.10 class RollingMat extends StaticObject implements Interactable

### 7.10.1 Fields

- ArrayList<Ingredient> placedIngredients	Create ArrayList for collecting placed ingredients.
- boolean confirmMerging	Default merging state to false.
- Dish dishOnMat	Default to null.

### 7.10.2 Constructor

+ RollingMat(String name, String imageUrl, double x, double y, boolean isCollidable, boolean isInteractable)	Set name, image url, position, collidable and interactable state.
--	---

### 7.10.3 Methods

+ void interact()	Transfer the container to the dish.
+ Dish getDishOnMat()	Get dish.
+ void clearIngredients()	Clear the ingredients from the rolling mat.

## 7.11 class Sink extends AnimatedObject

### 7.11.1 Field

+ boolean hasWater	Default to false.
--------------------	-------------------

### 7.11.2 Constructor

+ Sink(String name, String[] imageFrameURL, Double xPosition, Double yPosition, boolean isCollidable, boolean isInteractable)	Set name, image url, position, collidable and interactable state.
---	---

### 7.11.3 Method

+ void interact()	Add water to the allowed containers or wash some ingredients.
-------------------	---

## 7.12 class Stove extends AnimatedObject

### 7.12.1 Constructor

+ Stove(String name, String[] imageFrameURL, Double xPosition, Double yPosition, boolean isCollidable, boolean isInteractable)	Set name, image url, position, collidable and interactable state.
--	---

### 7.12.2 Method

+ void interact()	Cook the ingredients with allowed containers.
-------------------	---

## 7.13 class Table extends StaticObject implements Interactable

### 7.13.1 Constructor

+ Table(String name, String staticImageURL, Double xPosition,	Set name, image url, position, collidable and interactable state.
---	---

Double yPosition, boolean isCollidable, boolean isInteractable)	
---	--

### 7.13.2 Method

+ interact()	Place the container on the table.
--------------	-----------------------------------

## 7.14 class Trashcan extends StaticObject implements Interactable

### 7.14.1 Field

- boolean isConfirming	Default to false.
------------------------	-------------------

### 7.14.2 Constructor

+ Trashcan(String name, String imageUrl, double x, double y, boolean isCollidable, boolean isInteractable)	Set name, image url, position, collidable and interactable state.
--	---

### 7.14.3 Method

+ void interact()	Turn the heldItem of player to null by confirming.
-------------------	--

## 8. Package component

### 8.1 Player

#### 8.1.1 Fields

- double playerXPos, playerYPos	Player position.
- final int WIDTH, HEIGHT , startXPos, startYPos	Default player width to 32, Height to 32, start position x to 7, start position y to 3.
- double speed	Default speed to 0.5.
- Image playerImg	Player image.
- Heldable heldItem	Default hand item to null.
- <u>Player instance</u>	Instance of player

- MAP<DIRECTION, IMAGE> ANIMATIONS	Map that collects all player's moves according to direction.
---------------------------------------	--

## 8.1.2 Constructor

- Player()	Default to reset position.
------------	----------------------------

## 8.1.3 Method

+ void move(Direction direction)	CASE 1: UP direction increases player in Y axis with the player speed. CASE 2: DOWN direction decreases player in Y axis with the player speed. CASE 3: RIGHT direction increases player in X axis with the player speed. CASE 4: LEFT direction decreases player in X axis with the player speed.
+ void updateAnimation(Direction direction)	Change the player image according to player direction.
+ void render(GraphicsContext gc)	Draw the image of the player with player position, width, height and player image.
+ void reset()	Default all player status. (Position in X and Y axis, speed, default animation and setHeldItem to null.
+ static Player getInstance()	If there are no player instance create new one else return that instance
+ boolean isHolding(Class<?> clazz)	Check the class of player's HeldItem.
Getters / Setters	Getter and Setter for all fields