ProgChef

Created by

Natsurang Areepongsa 5930197121

Wianna Phaithoonmongkol 5931060021

2110215 Programming Methodology

Semester 2 Year 2017

Chulalongkorn University

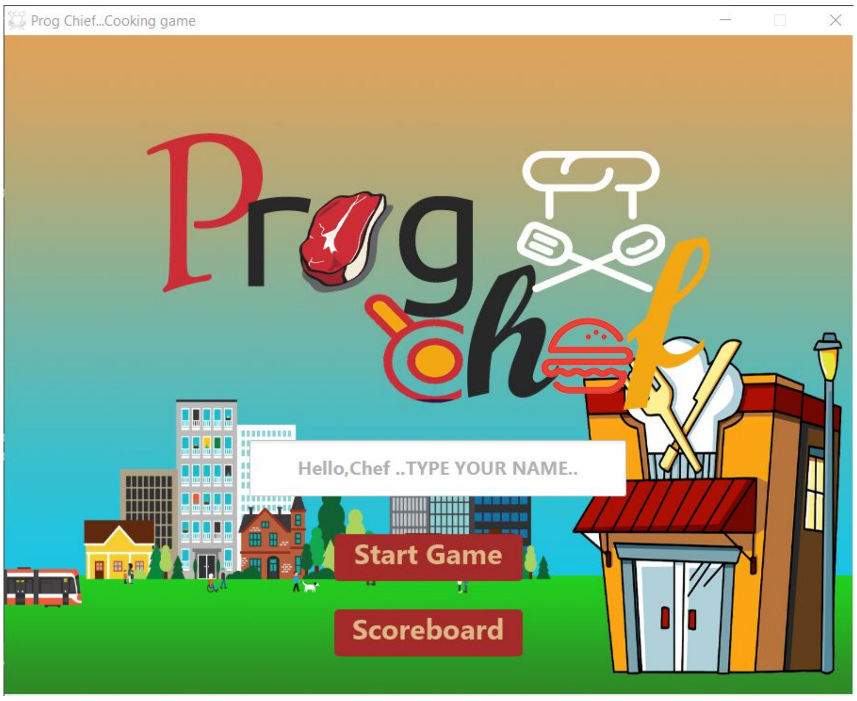
ProgChef

Introduction

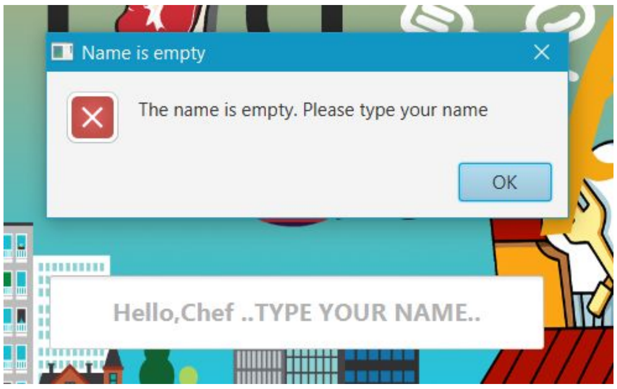
ProgChef is a cooking game which we have 3 ingredients(meat , vegetable , bread). We will have one state. We need to cook hamburger which have three form(meat+bread, vegetable+bread, vegetable+meat+bread) and it will have different point in each menu. This game the winner is the person who have high score.

User Material

Start scene



You need to fill the chef name before start a game.
If you don't fill, it will inform you like this.

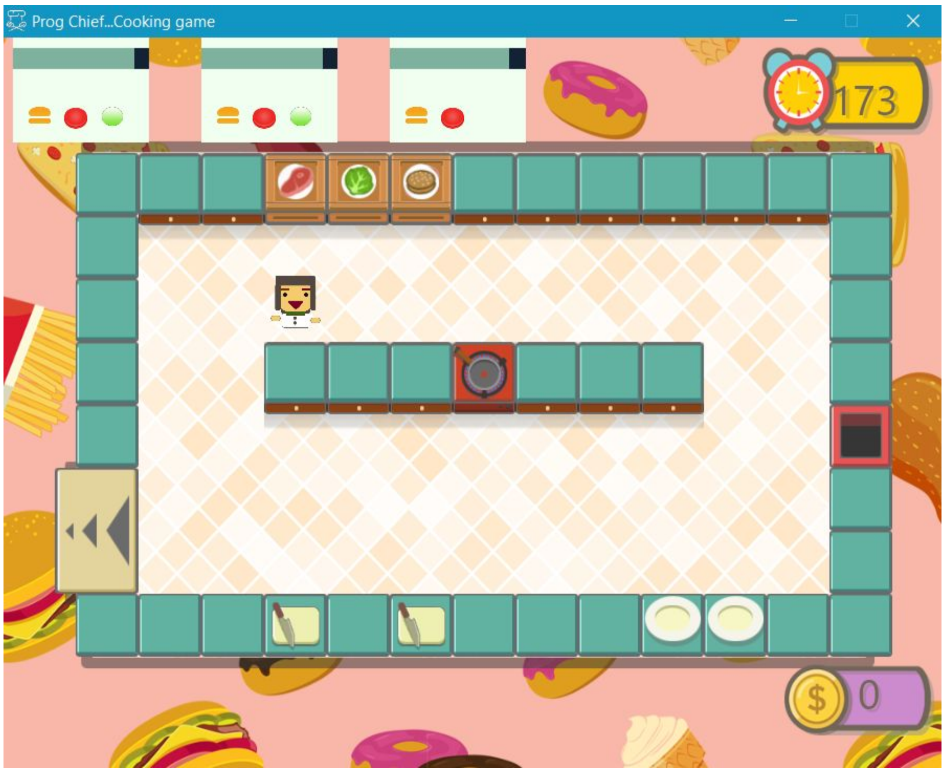


Name is empty

The name is empty. Please type your name

If you fill name, you can start a game.

Game Scene



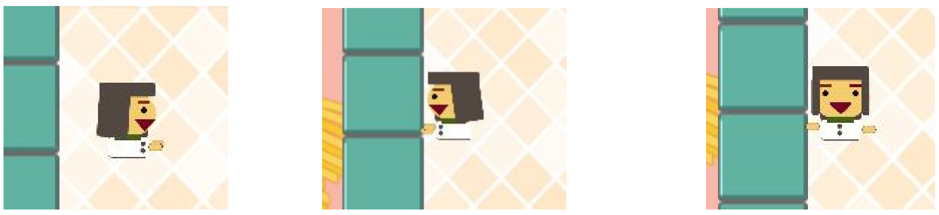
The time at the top right side. It starts at 179 minutes.

The point at the bottom right side. It start at 0.

The menu that you need to cook is on the top. Each menu have their own time. You can watch green. If it run out of time, it will disappear.

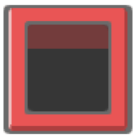
Ingredient

1. bread from bread chest
2. vegetable from vegetable chest
3. meat from meat chest



When we want to move, we use arrow to go right, left, up and down. We can’t surpass the counters.

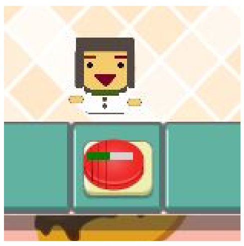
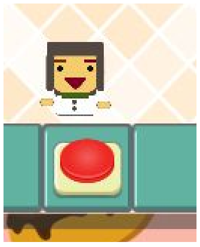
We press ‘A’ to put the ingredient and plate up and down in chest and to put it to counter, stove and cashier but we can’t put it on the floor and if you don’t want it, you can leave it to the garbage.


garbage counter(Plate can’t leave on this)

- We have a rule

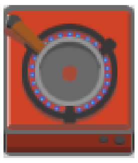
1. You need to face the counter that you will put the ingredient or plate.
2. The plate can’t place on stove and chopping board and the ingredients should be done in each state of them before place on the plate.
3. The bread can’t place on stove and chopping board.
4. The vegetable can’t place on stove.
5. In each counter, you can place only one ingredient.

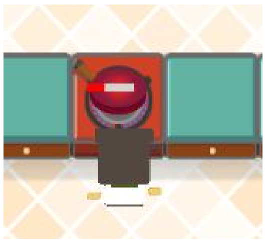
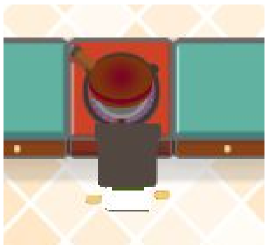
chopping counter

In the chopping state

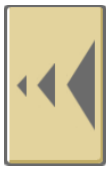
We press ‘S‘ to chop and we can chop meat and vegetable in chopping board. The ingredients. If it is done the green time will disappear and the size will smaller.

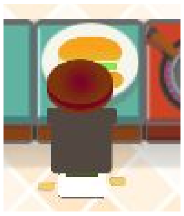
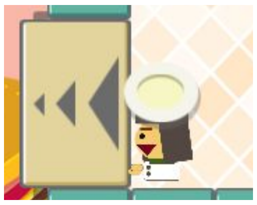
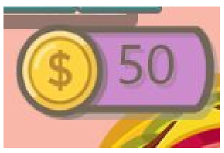

stove

In the ripe state

We press ‘A’ to ripe and we can ripe only meat and wait when it finish the red time will disappear and the color of the meat will change.


cashier

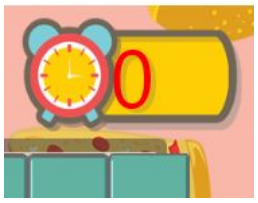




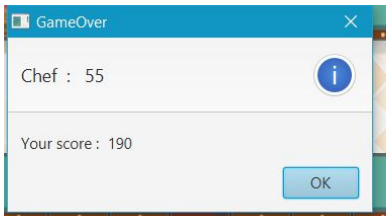
In finish state

When we finish cooking, we can put it on the plate. We will send to the cashier and the point will increase and the menu that is done, will disappear.

- vegetable+bread is 40 points.
- meat+bread is 50 points.
- vegetable+meat+bread is 70 points.

But if it don’t match to the menu at the top, the point won’t be increase and your food that you sent ,will disappear.




GameOver

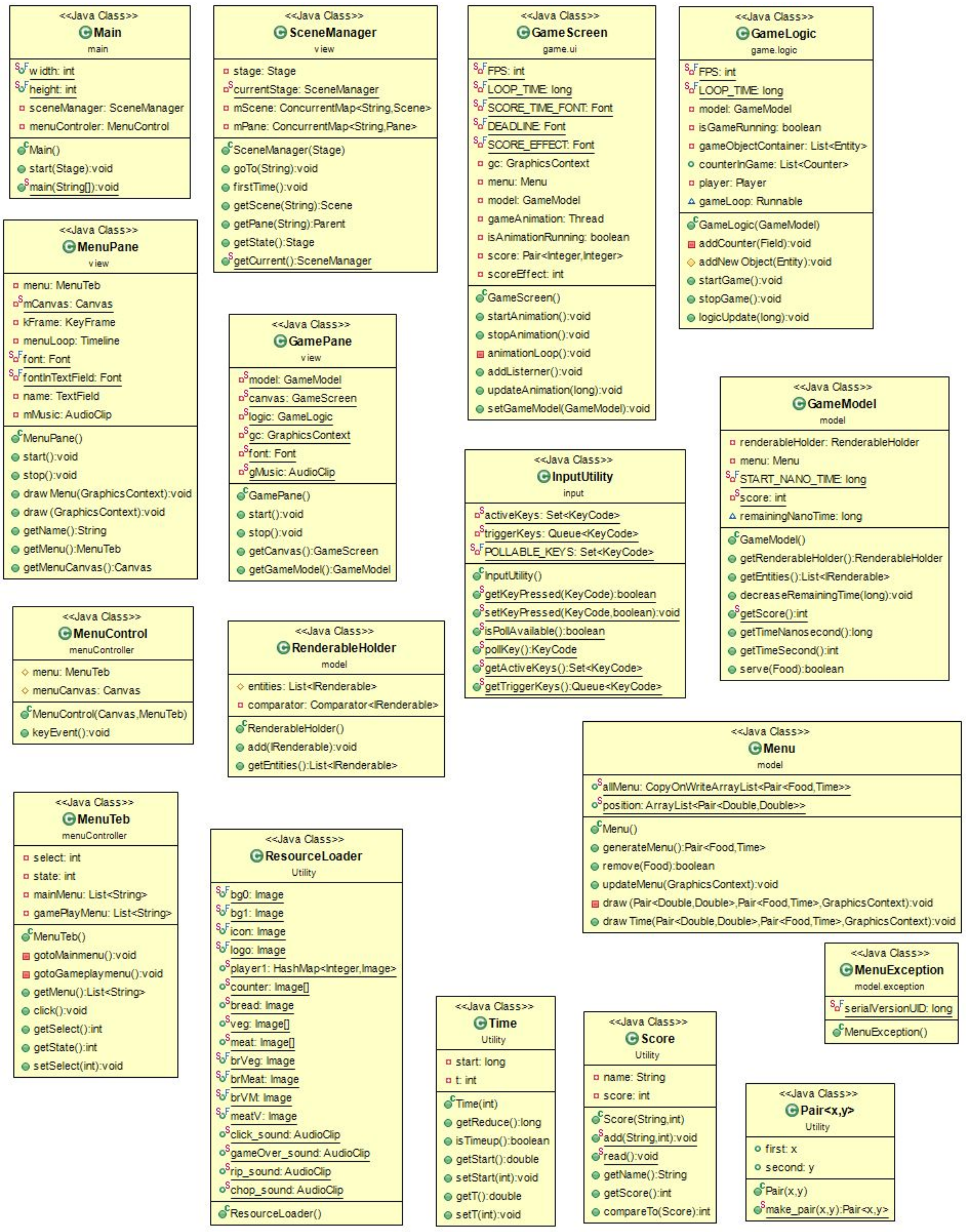
Chef : 55

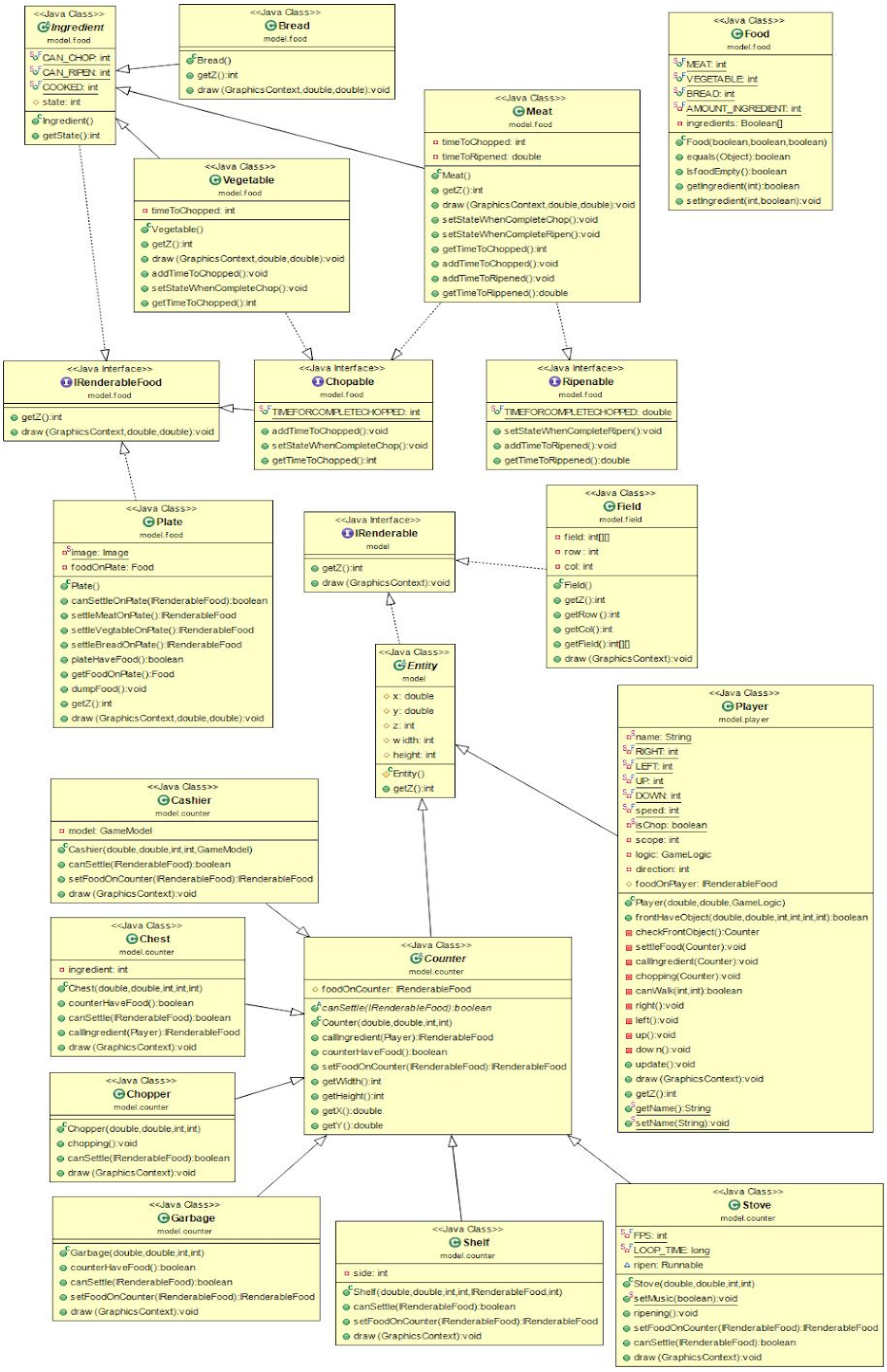
Your score : 190

Time up

The clock will be zero and it will inform your score.

Implemention detail





<<Java Class>> Ingredient model.food
CAN_CHOP: int
CAN_RIPEN: int
COOKED: int
state: int
Ingredient()
getState():int
<<Java Class>> Bread model.food
Bread()
getZ():int
draw(GraphicsContext,double,double):void
<<Java Class>> Food model.food
MEAT: int
VEGETABLE: int
BREAD: int
AMOUNT_INGREDIENT: int
ingredients: Boolean[]
Food(boolean,boolean,boolean)
equals(Object):boolean
IsfoodEmpty():boolean
getIngredient(int):boolean
setIngredient(int,boolean):void
<<Java Class>> Meat model.food
timeToChopped: int
timeToRipened: double
Meat()
getZ():int
draw(GraphicsContext,double,double):void
setStateWhenCompleteChop():void
setStateWhenCompleteRipen():void
getTimeToChopped():int
addTimeToChopped():void
addTimeToRipened():void
getTimeToRippened():double
<<Java Class>> Vegetable model.food
timeToChopped: int
Vegetable()
getZ():int
draw(GraphicsContext,double,double):void
addTimeToChopped():void
setStateWhenCompleteChop():void
getTimeToChopped():int
<<Java Interface>> IRenderableFood model.food
getZ():int
draw(GraphicsContext,double,double):void
<<Java Interface>> Chopable model.food
TIMEFORCOMPLETECHOPPED: int
addTimeToChopped():void
setStateWhenCompleteChop():void
getTimeToChopped():int
<<Java Interface>> Ripenable model.food
TIMEFORCOMPLETECHOPPED: double
setStateWhenCompleteRipen():void
addTimeToRipened():void
getTimeToRippened():double
<<Java Class>> Plate model.food
image: Image
foodOnPlate: Food
Plate()
canSettleOnPlate(IRenderableFood):boolean
settleMeatOnPlate():IRenderableFood
settleVegtableOnPlate():IRenderableFood
settleBreadOnPlate():IRenderableFood
plateHaveFood():boolean
getFoodOnPlate():Food
dumpFood():void
getZ():int
draw(GraphicsContext,double,double):void
<<Java Interface>> IRenderable model
getZ():int
draw(GraphicsContext):void
<<Java Class>> Field model.field
field: int[][]
row: int
col: int
Field()
getZ():int
getRow():int
getCol():int
getField():int[][]
draw(GraphicsContext):void
<<Java Class>> Entity model
x: double
y: double
z: int
width: int
height: int
Entity()
getZ():int
<<Java Class>> Player model.player
name: String
RIGHT: int
LEFT: int
UP: int
DOWN: int
speed: int
isChop: boolean
scope: int
logic: GameLogic
direction: int
foodOnPlayer: IRenderableFood
Player(double,double,GameLogic)
frontHaveObject(double,double,int,int,int,int):boolean
checkFrontObject():Counter
settleFood(Counter):void
callIngredient(Counter):void
chopping(Counter):void
canWalk(int,int):boolean
right():void
left():void
up():void
down():void
update():void
draw(GraphicsContext):void
getZ():int
getName():String
setName(String):void
<<Java Class>> Cashier model.counter
model: GameModel
Cashier(double,double,int,int,GameModel)
canSettle(IRenderableFood):boolean
setFoodOnCounter(IRenderableFood):IRenderableFood
draw(GraphicsContext):void
<<Java Class>> Chest model.counter
ingredient: int
Chest(double,double,int,int,int)
counterHaveFood():boolean
canSettle(IRenderableFood):boolean
callIngredient(Player):IRenderableFood
draw(GraphicsContext):void
<<Java Class>> Counter model.counter
foodOnCounter: IRenderableFood
canSettle(IRenderableFood):boolean
Counter(double,double,int,int)
callIngredient(Player):IRenderableFood
counterHaveFood():boolean
setFoodOnCounter(IRenderableFood):IRenderableFood
getWidth():int
getHeight():int
getX():double
getY():double
<<Java Class>> Chopper model.counter
Chopper(double,double,int,int)
chopping():void
canSettle(IRenderableFood):boolean
draw(GraphicsContext):void
<<Java Class>> Garbage model.counter
Garbage(double,double,int,int)
counterHaveFood():boolean
canSettle(IRenderableFood):boolean
setFoodOnCounter(IRenderableFood):IRenderableFood
draw(GraphicsContext):void
<<Java Class>> Shelf model.counter
side: int
Shelf(double,double,int,int,IRenderableFood,int)
canSettle(IRenderableFood):boolean
setFoodOnCounter(IRenderableFood):IRenderableFood
draw(GraphicsContext):void
<<Java Class>> Stove model.counter
FPS: int
LOOP_TIME: long
ripen: Runnable
Stove(double,double,int,int)
setMusic(boolean):void
ripening():void
setFoodOnCounter(IRenderableFood):IRenderableFood
canSettle(IRenderableFood):boolean
draw(GraphicsContext):void

1. Package main

1.1 Class Main extends Application

1.1.1 Field

+ static final widht: int	900
+ static final height	700
- SceneManager sceneManager	It will help us to change scene.
- MenuControl menuControler	It controls menu in the first page.

1.1.2 Method

+ start(Stage primaryStage): void	-Initialize SceneManager, MenuControler -tell ResourceLoader to load resource -set title to "Prog Chief...Cooking game" -set the stage to be non-resizable -set icon that use from ResourceLoader.icon -tell sceneManager to go to menu page.
+ main(String[] args): void	An entry point to the JavaFX program. This should call Application's launch method.

2. Package game.logic

2.1 Class GameLogic

2.1.1 Field

- static final FPS: int	Number of frame rates per second. set is 60

- static final LOOP_TIME: long	Time period between each update of a game animation
- model: GameModel	A game model
- isGameRunning: boolean	The flag indicate that the game is start and not end yet.
- gameObjectContainer: List<Entity>	The list of enity in the game
+ counterInGame: List<Counter>	The list that contain all of counter in the game
- player: Player	A player

2.1.2 Method

+ GameLogic(GameModel model)	- set model by the given parameters - create new field and use this field to add counter by call addCounter(field) - add player - set isGameRunning to false
- addCounter(Field field): void	Add counter to the game by data in parameter
# addNewObject(Entity entity): void	Add parameter to the game
+ startGama(): void	Aet isGameRunning is true and start the game loop
+ stopGame(): void	Set isGameRunning is false
+ logicUpdate(long elapsedTime): void	- update player in this game, - decrease remaining time of model, - If the remaining time reaches zero, stop the game.

3. Package game.ui

3.1 Class GameScreen extends Canvas

3.1.1 Field

- static final FPS: int	Number of frame rates per second. set is 60
- static final LOOP_TIME: long	Time period between each update of a game animation
- static final SCORE_TIME_FONT: Font	A score and time font
- static final DEADLINE: Font	A time font for deadline time
- static final SCORE_EFFECT: Font	A increased score font
- gc: GraphicsContext	GraphicsContext of gameScreen
- menu: Menu	A menu
- private model: GameModel	A game model
- gameAnimation: Thread	A thread for game animation
- isAnimationRunning: boolean	The flag indicate that the game is start and not end yet
- score: Pair<Integer,Integer>	The last score and amount of changed score
- scoreEffect: int	Time of effect when score change

3.1.2 Method

+ GameScreen()	- set width and height from field of main - create new menu, - set isAnimationRunning to false - add event handlers for this canvas by calling addListerner()

+ startAnimation(): void	- request focus to this screen - set isAnimationRunning to true - initialize and start game animation loop
+ stopAnimation(): void	Set isAnimationRunning to false
- animationLoop(): void	The game animation loop, the loop will stop when isGameRunning is false and for each loop, call updateAnimation()
+ addListener(): void	When player press or release any letter key use this key set InputUtility
+ updateAnimation(long now): void	- draw all of entity from game model - draw time from remainning time of model - draw present score + effect of score - update menu
+ setGameModel(GameModel model): void	Set game model to the given model

4. Package input

4.1 Class InputUtility

4.1.1 Field

- static activeKeys: Set<KeyCode>	A set of pressing input character
- static triggerKeys: Queue<KeyCode>	A queue that holds the input characters that should be triggered in a next polling time.
- static final POLLABLE_KEYS: Set<KeyCode>	A set of input character that able to polling

4.1.2 Method

+ static getKeyPressed(KeyCode keycde): boolean	Check given keycode that was in activeKeys.
+ static setKeyPressed(KeyCode keycode, boolean pressed): void	If press is true - add keycode to triggerKey when keycode was in POLLABLE_KEYS and activeKeys not contain this keycode - add keycode to activeKeys If press is false - remove keycode from activeKeys
+ static isPollAvailable(): boolean	Return true if tiggerKeys have any member
+ static pollKey(): KeyCode	Return head of triggerKeys member
+ static getTriggerKeys(): Set<KeyCode>	Return the current activeKeys
+ static getTriggerKeys(): Queue<KeyCode>	Return the current triggerKeys

5. Package model

5.1 Class Menu

food that customer order

5.1.1 Field

+ static allMenu :CopyOnWriteArrayList<Pair<Food, Time>>	It keeps every menu that haven't finish.
+ static position: ArrayList<Pair<Double, Double>>	add position that menu will show by using Pair and sent 2 parameter position x and y.

5.1.2 Method

+ Menu()	initialize allMenu
+ generateMenu(): Pair<Food, Time>	random and if we get - 1 food will be vegetable+ bread and initialize time 30000 - 2 food will be meat+bread and initialize time 50000 - 3 food will be vegetable+meat+bread and initialize time 60000 and add food and time to allMenu
+ remove(Food food): boolean	If we finish in this dish, this will remove food from allMenu and return true but if the dish isn’t connect to menu, it will return false.
+ updateMenu(GraphicsContext gc) throws MenuException: void	draw menu and if timeup, it will remove menu and if size of allMenu equal 4 or more than it will throw MenuException and if it equal less than 3 , will generate menu.

5.2 Class GameModel

5.2.1 Field

- static final START_NANO_TIME: long	duration time per 1 game in nanoseconds
- renderableHolder: RenderableHolder	A renderableHolder that keep object that can render in the game
- menu: Menu	A menu
- static score: int	score in the game
~ remainingNanoTime: long	remaining time in nanoseconds

5.2.2 Method

+ GameModel()	- set score to 0 - set remainingNanoTime as START_NANO_TIME - add new three menu by call menu.generateMenu()
+ decreaseRemainingTime(long decreaseNonoTime): void	subtract the remaining time by decreaseNonaTime
+ serve(Food food): boolean	If given food in the menu - add score by ingredient in food (meat = 30, vegetable = 20, bread = 20) - add new menu - return true else return false
+ getRenderableHolder(): RenderableHolder	return the current renderableHolder
+ getEntities(): List<IRenderable>	return the current list of entities from renderableHolder
+ static getScore(): int	return the current score
+ getTimeNanosecond(): long	return the current remaining time in nanoseconds
+ getTimeSecond(): int	return the current remaining time in seconds

5.3 Class RenderableHolder

5.3.1 Field

# entities: List<IRenderable>	A collection of entities
- comparator: Comparator<IRenderalbe>	A comparator for compare Z value of each IRenderable

5.3.2 Method

+ RenderableHolder()	- Initialize entities and comparator
+ add(IRenderable entity): void	- add given entity to entities - sort entities by comparator
+ getEntities(): List<IRenderable>	return the current entities

5.4 Abstract Class Entity

5.4.1 Field

# x: double	A value of x position
# y: double	A value of y position
# z: int	z value (if some IRenderable z is lower, it will draw after)
# width: int	width of each IRenderable
# height: int	height of each IRenderable

5.4.2 Method

# Entity()	call super()
+ getZ(): int	return the z value

5.5 Interface IRenderable

5.5.1 Method

+ getZ(): int	do nothing
+ draw(GraphicsContext gc): void	do nothing

6. Package model.counter

6.1 Abstract Class Counter extends Entity

6.1.1 Field

# foodOnCounter: IRenderableFood	A food on the counter

6.1.2 Method

+ Counter(double x, double y, int w, int h)	- set x, y, widht and height by given parameter - set foodOnCounter to null
+ abstract canSettle(IRenderableFood foodOnPlayer): boolean	do nothing
+ callIngredient(Player player): IRenderableFood	return food on counter and set foodOnCounter to null
+ counterHaveFood(): boolean	return true if food on counter isn't null
+ setFoodOnCounter (IRnedeableFood food): IRenderableFood	set food on counter as given food
+ getWidth(): int	return the width
+ getHeight(): int	return the height
+ getX(): double	return the current X position
+ getY(): double	return the current Y position

6.2 Class Chest extends Counter

6.2.1 Field

- ingredient: int	use any number instead of ingredient type

6.2.2 Method

+ Chest(double x, double y, int w, int h, int ingredient)	- set x, y, widht, height and Ingredient by given parameter - set foodOnCounter to null
+ counterHaveFood()	always return true
+ canSettle(IRenderableFood foodOnPlayer): boolean	return true when foodOnCounter isn't null and foodOnplayer is Ingredient
+ callIngredient(Player player): IRenderableFood: IRenderable	If it has food on counter - return food on counter and set to null else - return ingredient that match with type of ingredient
+ draw(GraphicsContext gc): void	- draw this chest with image chest that match type - draw food on chest if it isn't null

6.3 Class Chopper extends Counter

6.3.1 Method

+ Chopper(double x, double y, int w, int h)	- set x, y, widht, height and by given parameter - set foodOnCounter to null - set check to false
+ chopping(): void	If food on counter can chop - add time to chop of foodOnCounter - if time to chop equal or more than time for complete chopped, set food on counter to state complete chop
+ canSettle(IRenderableFood foodOnPlayer): boolean	return trun when foodOnCounter isn't null and foodOnplayer is chopable and can chop

+ draw(GraphicsContext gc)	- draw with image for chopper - if have food on counter draw food on counter

6.4 Class Cashier extends Counter

6.4.1 Field

- model: GameModel	the game model

6.4.2 Method

+ Cashier(dooubel x, double y, int w, int h, GameModel model)	- set x, y, widht, height and model by given parameter - set foodOnCounter to null
+ canSettle(IRenderableFood foodOnPlayer): boolean	return ture if food on player is plate and have food on plate
+setFoodOnCounter(IRenderablefood food): IRenderableFood	if food is plate - serve food on plate to model - dump food on plate - return plate
+ draw(GraphicsContext gc): void	- draw with image for cashier - if have food on counter draw food on counter

6.5 Class Garbage extends Counter

6.5.1 Method

+ Garbage((double x, double y, int w, int h)	- set x, y, widht, height by given parameter - set foodOnCounter to null
+ counterHaveFood(): boolean	alway return false
+ canSettle(IRenderableFood foodOnPlayer): boolean	return fasle when food is on plate and don’t have any ingredient otherwise return true

+setFoodOnCounter(IRenderable Food food): IRenderableFood	If food is plate - dump food on plate - return plate
+ draw(GraphicsContext gc)	- draw with image for garbage - If have food on counter draw food on counter

6.6 Class Shelf extends Counter

6.6.1 Field

- side: int	use number instead of turn side of shelf

6.6.2 Method

+ Shelf(double x, double y, int w, int h, IRenderableFood food, int side)	- set x, y, widht, height and side by given parameter - set foodOnCounter to null
+ canSettle(IRenderableFood foodOnPlayer): boolean	- If food on counter is null return ture - If food on counter is plate and plate can settle food return ture - If food on player is plate and it can settle food return ture
+ setFoodOnCounter(IRenderable Food foodOnPlyer): IRenderableFood	if food on player is null - set food on counter as food on player else - if food on player or counter is plate settle another food to the plate and set food on counter to it
+ draw(GraphicsContext gc): void	- draw with image for shelf which match the side - if have food on counter draw food on counter

6.7 Class Stove extends Counter

6.7.1 Field

- static final FPS: int	Number of frame rates per second. set is 60
- static final LOOP_TIME: long	Time period between each update of a game animation

6.7.2 Method

+ Stove(double x, double y, int w, int h)	- set x, y, widht, height by given parameter - set foodOnCounter to null
+ static setSound(boolean play): void	play sound when ripening
+ ripening(): void	If food on counter and can ripen - ripening setSound to true - If time to ripened equal or more than time for complete ripened, set food on counter to state complete ripened
+setFoodOnCounter(IRenderable Food): IRenderable	set food on counter and ask ripening to do
+ canSettle(IRenderableFood foodOnPlayer): boolean	return trun when foodOnCounter isn't null and foodOnplayer is ripenable and can ripen
+ draw(GraphicsContext gc): void	- draw with image for stove - If have food on stove draw food on stove

7. Package model.field

7.1 Class Field Implements IRenderable

7.1.1 Field

- filed: int[][]	Collection of position of counter(use int to symbol of each counter) in field
- row: int	Amount row of the field
- col: int	Amount column of the field

7.1.2 Method

+ Field()	- set position of each counter by initialize array - set row as length of field - set col as length of field
+ getZ(): int	return z value
+ draw(GraphicsContext gc): void	draw background with canvas size
+ getter of all field	

8. Package model.food

8.1 Intreface IRenderableFood

8.1.1 Method

+ getZ(): int	return position
+ draw(GraphicsContext gc, double x, double y): void	draw image

8.2 Interface Chopable extends IRenderableFood

interface for ingredient that can chop

8.2.1 Field

+ static final TIMEFORCOMPLETECHOPPED: int	Time in chopping that will set to 70

8.2.2 Method

+ addTimeToChopped(): void	add time in chopping
+ setStateWhenCompleteChop(): void	set state when finish chopping
+ getTimeToChopped(): int	return timeToChopped

8.3 Interface Ripenable

interface for ingredient that can ripen

8.3.1 Field

+ static final TIMEFORCOMPLETECHOPPED: double	Time in riping that will set to 720

8.3.2 Method

+ setStateWhenCompleteRipen(): void	set state when finish ripening
+ addTimeToRipened(): void	add time in riping
+ getTimeToRippened(): double	return timeToRipened

8.4 Abstract Class Ingredient

abstract class of all ingredient

8.4.1 Field

+ static final COOKED: int	ready to serve will set it to 3.
+ static final CAN_RIPEN: int	need to ripe will set it to 2.
+ static final CAN_CHOP: int	need to chop will set it to 1.
# state: int	state of ingredient

8.4.2 Method

+ getState(): int	return stage

8.5 Class Bread extends Ingredient

bread in game

8.5.1 Method

+ Bread()	set state = COOKED
+ getZ(): int	reuturn 0
+ draw(GraphicsContext gc, double x, double y): void	draw image from ResourceLoader.bread and x = x-25 , y = y-60 , width = 50, height = 50

8.6 Class Vegetable extends Ingredient implements Chopable

vegetable in game

8.6.1 Field

- timeToChopped: int	Time that vegetable use to chop

8.6.2 Method

+ Vegetable()	At the first time, we need to set state to CAN_CHOP and timeToChopped to 0.
+ getZ(): int	return 0
+ draw(GraphicsContext gc, double x, double y): void	-state = CAN_CHOP we need to split time to each chopping and draw ResourceLoader.veg that is a array and set x = x-30, y = y-60, width = 55, height = 50 and it has a green bar that is the time of chopping.When it finishes, it’s gone and set state to COOKED

+ addTimeToChopped(): void	add time in chopping
+ setStateWhenCompleteChop(): void	set state when finish chopping
+ getTimeToChopped(): int	return timeToChopped

8.7 Class Meat extends Ingredient implements Chopable, Ripenable

Meat in game

8.7.1 Field

- timeToChopped: int	It is the time that vegetable neet to use to chop.
- timeToRipened: double	It is the time that meat need to ripe.

8.7.2 Method

+ Meat()	At the first time, we need to set state to CAN_CHOP , timeToChopped to 0 and timeToRipened to 0.
+ getZ(): int	return 0
+ draw(GraphicsContext gc, double x, double y): void	-state = CAN_CHOP we need to split time to each chopping and draw ResourceLoader.meat that is a array and set x = x-30, y = y-60, width = 55, height = 50 and it has a green bar that is the time of chopping.When it finishes, it’s gone and set state to CAN_RIPEN. -state = CAN_RIPEN we need to split time to each ripening and draw

	ResourceLoader.meat that is a array and set x = x-20, y = y-60, width = 40, height = 45 and it has a red bar that is the time of chopping.When it finishes, it’s gone and set state to COOKED.
+ setStateWhenCompleteRipen(): void	set state when finish ripening
+ addTimeToRipened(): void	add time in riping
+ getTimeToRippened(): double	return timeToRipened
+ addTimeToChopped(): void	add time in chopping
+ setStateWhenCompleteChop(): void	set state when finish chopping
+ getTimeToChopped(): int	return timeToChopped

8.8 Class Food

type of food in the game

8.8.1 Field

+ static final MEAT: int	set to 0
+ static final VEGETABLE: int	set to 1
+ static final BREAD: int	set to 2
- static final AMOUNT_INGREDIENT : int	set to 3
- ingredients: Boolean[]	keep the ingredients

8.8.2 Method

+ Food(boolean haveMeat, boolean haveVegetable, boolean haveBread)	set ingredients[MEAT] = haveMeat , ingredients[VEGETABLE] =

	haveVegetable, ingredients[BREAD] = haveBread
+ equals(Object o): boolean	check ingredient in array ingredients that equal to Food o
+ IsfoodEmpty(): boolean	check that we have ingredient in array ingredients
+ getIngredient(int i): boolean	return ingredients[i]
+ setIngredient(int i, boolean have): void	ingredients[i] = have

8.9 Class Plate implements IRenderableFood

plate in game

8.9.1 Field

- static image: Image	new Image with the resource
- foodOnPlate: Food	Food that on the plate

8.9.2 Method

+ Plate()	foodOnPlate = new Food(false, false, false)
+ getFoodOnPlate(): Food	return foodOnPlate
+ dumpFood(): void	drum the Food to set everything to false in plate
+ getZ(): int	return 0
+ canSettleOnPlate(IRenderableFood food): boolean	return that the ingredient can settle on the plate
+ settleMeatOnPlate(): IRenderableFood	setIngredient meat = true
+ settleVegtableOnPlate(): IRenderableFood	setIngredient vegetable = true

+ settleBreadOnPlate(): IRenderableFood	setIngredient bread = true
+ plateHaveFood(): boolean	return not of asking foodOnPlate is foodEmpty
+ draw(GraphicsContext gc, double x, double y): void	draw plate that from image set position x = x-35, y = y-70,width = 70,height = 70 and draw food and set x ,y ,width and height that appropriate.

9. Package model.exception

9.1 Class MenuException extends ArrayIndexOutOfBoundsException

menu exception will catch when menu is equal or more than 4

9.1.1 Field

- static final serialVersionUID: long	1L

9.1.2 Method

+ MenuException()	We have menu but the area to show menu is limit so we need to check if it is more than and prints "Menu out of bounds".

10. Package model.player

10.1 Class Player extends Entity

player in game

10.1.1 Field

- static final RIGHT: int	set to 1
- static final LEFT: int	set to 2
- static final UP: int	set to 3
- static final DOWN: int	set to 4

- static final speed: int	set to 4
- scope: int	It is the scope that player can do.
- static name: String	It is player’s name.
- logic: GameLogic	It is gamelogic.
- direction: int	It is the direction that player need to go.
# foodOnPlayer: IRenderableFood	Things that will on player.

10.1.2 Method

+ Player(double x, double y, GameLogic logic)	set x, y ,logic like input that we get set width and height to 40 , set scope to ⅓ of width, set direction to DOWN and foodOnPlayer to null
+ frontHaveObject(double otherX, double otherY, int otherW, int otherH, int changeX, int changeY): boolean	tell that we have thing in front of the player.
- checkFrontObject(): Counter	check in front of player and return a counter if it is counter. We have frontX and frontY that : -direction =LEFT : frontX = -scope -direction =RIGHT:frontX = scope -direction = UP : frontY = -scope -direction =DOWN: frontY = scope and use frontHaveObject that it is the counter that in front of player if it isn’t Counter return null
- settleFood(Counter counter): void	settle ingredient on counter. Player will ask counter that canSettle and settle it on counter.
- callIngredient(Counter counter): void	check when we run process that on the counter have ingredient: If we have ingredient, prints

	"success for call ingredient" and set food on counter to foodOnPlayer or prints "counter not have food" if it isn’t Food. If counter doesn’t have any ingredient, prints "In front of not have counter for use"
- chopping(Counter counter): void	ask that counter is Chopper and than if on counter have ingredient tell counter to chop and set isChop to true and if it doesn’t have ingredient prints "no food can chopping" and set isChop to false.If this counter isn’t Chopper, prints "It's not Chopper" and set isChop to false.
- canWalk(int changeX, int changeY): boolean	use frontHaveObject to decide to return value
- right(): void	set direction and plus position in x by using speed
- left(): void	set directiom and minus position in x by using speed
- up(): void	set direction and plus position in y by using speed
- down(): void	set directiom and minus position in y by using speed
- update(): void	check in front of object and use InputUtility in direction and press ‘s’ to chop, press ‘a’ to settle and call ingredients on counter.
+ draw(GraphicsContext gc): void	draw player by using in ResourceLoader.player1 that is arraylist to get position from direction and draw food on player.

+ getZ(): int	return 9
+ static getName(): String	return name
+ static setName(String name): void	set player's name

11. Package Utility

11.1 Class Pair<x,y>

generic class that we want to make Pair like c++ programing

11.1.1 Field

+ first: x	first value that don't know type
+ second: y	second value that don't know type

11.1.2 Method

+ Pair(x first, y second)	set first and second
+ static <x, y> make_pair(x first, y second): Pair<x, y>	making pair by using make_pair like c++ programing

11.2 Class Score implements Comparable<Score>

use to open score file

11.2.1 Field

- name: String	name of player
- score: int	amount of score

11.2.2 Method

+ Score(String name, int score)	set initialization of name and score
+ static add(String name, int score): void	add score in file score.dat by using BufferedWriter and write name and score and flush file *should have try-catch exception

+ static read(): void	read file score.dat and bring it in queue to check top 10 that we will annouce
+ compareTo(Score o): int	compare o’s score and this score and return
+ getter of all field	

11.3 Class Time

use in menu in the game

11.3.1 Field

- start: long	tell start time when we init the thing
- t: int	Amount of time of the thing that we want in initialization

11.3.2 Method

+ Time(int tm)	set t from input and set start from this time
+ getReduce(): long	difference of t ,time at the beginning and this time that multiply 130 and divide t
+ isTimeup(): boolean	check time at the beginning and this time that more than t
+ getters and setters of all field	

11.4 Class ResourceLoader

use to load picture and audio in game.

11.4.1 Field

+ static final bg0: Image	Background of menu page
+ static final bg1: Image	Background of game page

+ static final icon: Image	Icon of game
+ static final logo: Image	Logo of our game that will draw in menu page
+ static player1: HashMap<Integer, Image>	Map that will choose the picture 1 is right, 2 is left, 3 is back and 4 is front
+ static counter: Image[]	Array of picture of counter that 0 is shelf type 1, 1 is shelf type 2, 2 is meat chest,3 is vegetable chest, 4 is bread chest,5 is chopper, 6 is chopper when chopping,7 is stove,8 is cashier and 9 is garbage.
+ static bread: Image	Picture of bread
+ static veg: Image[]	Array of picture of vegetable that 0 is getting from chest, 1-6 is when chopping and 7 is final
+ static meat: Image[]	Array of picture of meat that 0 is getting from chest, 1-7 is when chopping and 8-9 is when ripening
+ static final brVeg: Image	Picture of bread and vegetable
+ static final brMeat: Image	Picture of bread and meat
+ static final brVM: Image	Picture of bread, vegetable and meat
+ static final meatV: Image	Picture of meat and vegetable
+ static click_sound: AudioClip	Audio of clicking at menu page
+ static gameOver_sound: AudioClip	Audio when game is time up
+ static rip_sound: AudioClip	Audio when ripening
+ static chop_sound: AudioClip	Audio when chopping

11.4.2 Static Block

Set player1, meat, veg, and counter to array with ordered.

12. Package menuController

12.1 Class MenuControl

keep key to make a process in menu pane

12.1.1 Field

# menu: MenuTeb	control logic of menu.
# menuCanvas: Canvas	Canvas that will draw menu.

12.1.2 Method

+ MenuControl(Canvas mCanvas, MenuTeb menu)	set menu and menuCanvas
+ keyEvent(): void	set event by using mouse if it is in the menu area, it will print “Menu”+order of number and set menu select in that number-1 and if it ot of menu area menu set select to -1 and if we click ,we will tell menu to click.

12.2 Class MenuTeb

logic of menu pane

12.2.1 Field

- select: int	set to -1
- state: int	set to 0
- mainMenu: List<String>	initialize CopyOnWriteArrayList and add “”,"Start Game","Scoreboard"
- gamePlayMenu: List<String>	initialize CopyOnWriteArrayList and add "Play again","Scoreboard","Change Player"

12.2.2 Method

- gotoMainmenu(): void	set select to -1 and state to 0
- gotoGameplaymenu(): void	set select to -1 and state to 1
+ getMenu(): List<String>	return menu that we want to use
+ click(): void	play ResourceLoader.click_sound ,check state - state = 0 select = 1 It mean that you want to play game.If you don't write your name, it will alert then you need to write your name and go to game select = 2 score will be read. - state = 1 select = 0 It mean you want to play again so it will go to game page select = 1 score will be read. select = 2 gotoMainmenu
+ getSelect(): int	return select
+ getState(): int	return state
+ setSelect(int i): void	select = i

13. Package view

13.1 Class ScenceManager

control scene in Prog Chef

13.1.1 Field

- stage: Stage	this stage
- static crrS: SceneManager	set to null
- mScene: HashMap<String, Scene>	Mapping of scene

- mPane: HashMap<String,Pane>	Mapping of pane

13.1.2 Method

+ SceneManager(Stage stage)	set state ,currentStage ,new GamePane , MenuPane put to mPane and new Scene of each of mPane
+ goTo(String key): void	setScene of key in mScene and requestFocus
+ getters of all field	

13.2 Class MenuPane extends AnchorPane

pane of menu

13.2.1 Field

- menu: MenuTeb	Menu in menu page
- static mCanvas: Canvas	Canvas that use in manu page
- kFrame: KeyFrame	Keyframe of to handle
- menuLoop: Timeline	Timeline of menu
- static final font: Font	Font in menu page in menuteb
- static final fontInTextField: Font	Font in textfield
- name: TextField	Text in textfield
- mMusic: Audio	Audio in menu page

13.2.2 Method

+ MenuPane()	initialize all field except menuLoop, set font and text in textfield to "Hello,Chef ..TYPE YOUR NAME.." and set keyframe to Duration.millis(23) -state of menu = 0 draw background ,logo , menu

	and set name to visible -state of menu = 1 draw background, logo,menu and set name to invisible
+ start(): void	play mMusic ,set cycle of audioclip to INDEFINITE ,initialize menuLoop ,set cycle of timeline to INDEFINITE,add kFrame and play
+ stop(): void	stop mMusic and menuLoop
+ drawMenu(GraphicsContext gc): void	get menu and draw roundrects that archWidth and arcHeight = 10 color when don't click : roundrect is brown and font is burlywood color when click: roundrect is chocolate and font is beige
+ draw(GraphicsContext gc): void	draw background and logo
+ getName(): String	return name from textfield
+ getMenu(): MenuTeb	return menu
+ getMenuCanvas(): Canvas	return mCanvas

13.3 Class GamePane extends Pane

pane of game

13.3.1 Field

- static model: GameModel	game model that connect to model of player, ingredient, food and menu
- static canvas: GameScreen	game screen is a canvas to draw game model
- static logic: GameLogic	game logic is a logic of game to make game have a process

- static gc: GraphicsContext	using to draw a picture in game pane
- static font: Font	set font
- static gMusic: AudioClip	set music in game

13.3.2 Method

+ GamePane()	initialize GameScreen and gc
+ start(): void	clear all key in inputUtility, play gMusic ,set cycle of audioclip to INDEFINITE, initialize GameModel, setGameModel and GameLogic and start game
+ stop(): void	stop gMusic, stop game, send score and announce to player ang go to menu page
+ getCanvas(): GameScreen	return canvas
+ getGameModel(): GameModel	return model