Elizabeth Chen, Linda Zheng
Period 1
Group Name: Pobrecitas
Game Name: Cooking Baba

2. Description:

Our game will be a replication of the popular Nintendo Game, Cooking Mama. There will be two main levels where users can make either a cake or curry. Each level or dish requires several different tasks that generally fall under the same categories, such as obtaining the ingredients, preparing them, stirring, and serving/decorating. Despite falling under the same categories, the tasks are still different depending on the level the player chooses to play.

Functionalities:

Cake Level:

- Step 1: Catch the Chickens
  - The objective is to catch **Chicken objects** that will be bouncing across the screen. The purpose is to "get" eggs to bake the cake. The player must click on the chicken to obtain the eggs for the cake. The game will utilize vectors in order to move the chicken across the screen.

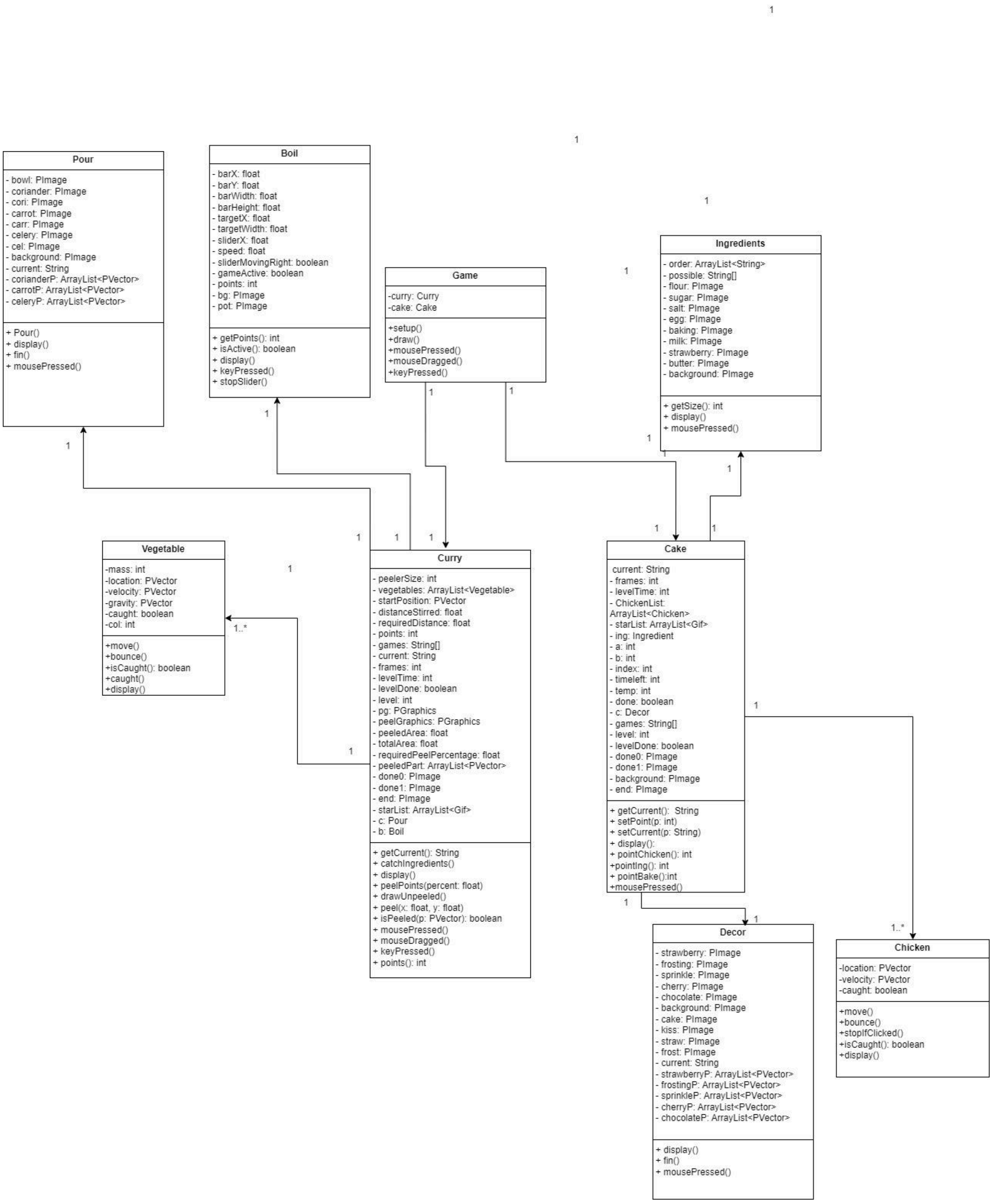
- Step 2: Choose Ingredients
  - The objective is to pick all the right ingredients for the cake. There will be an ingredient and players need to choose the right one. After the players choose the right ingredient, the next one appears. Each time the game is run, the ingredients are in a different order. The players must correctly choose all the ingredients before the timer runs out.
- Step 3: Bake
  - The objective is to bake the cake at the right temperature. There are two buttons: one that increases the temperature and the other one that decreases the temperature. The game will display a math problem to find the right temperature, so the players have to adjust the temperature until it is the answer to the math problem.
- Step 5: Decorate
  - Objective to decorate the cake. Players will be able to place toppings on top of the cake. The game will utilize vectors to keep track of the x and y coordinates of the toppings. The toppings will be represented by PImages.

Curry Level:

- Step 1: Catch Vegetables

    - The objective is to catch falling vegetables into baskets. Players need to avoid non-ingredients. The game will utilize vectors and physics to simulate the gravity of the falling vegetables.
- Step 2: Peel Vegetables
    - The objective is to peel vegetables with a peeler. Players need to shave 90% or more of the vegetable's surface in order to earn a point. There will be a vegetable and players have to color or "peel" it. The area of the colored-in or "peeled" part of the vegetable will be compared to the area of the vegetable. The game will use PGraphics to draw the vegetable as a circle and the "peeled" part as a smaller circle.
- Step 3: Boil
    - The objective is to cook or boil the ingredients properly. There will be a slider that moves along a bar with a red portion and a green portion. The placement of the green portion will be randomly selected. Players need to hit the spacebar when the slider is within the green portion in order to boil the ingredients.
- Step 4: Serve and Decor
    - The objective is to serve the rice and decorate the curry dish. Players can select different toppings-such as coriander, carrots, and celery. The game will use vectors to keep track of the x and y coordinates. This portion will use PImage to place the images of the toppings.

3.



Pour

- bowl: PImage
- coriander: PImage
- cori: PImage
- carrot: PImage
- carr: PImage
- celery: PImage
- cel: PImage
- background: PImage
- current: String
- corianderP: ArrayList<PVector>
- carrotP: ArrayList<PVector>
- celeryP: ArrayList<PVector>

+ Pour()
+ display()
+ fin()
+ mousePressed()

Boil

- barX: float
- barY: float
- barWidth: float
- barHeight: float
- targetX: float
- targetWidth: float
- sliderX: float
- speed: float
- sliderMovingRight: boolean
- gameActive: boolean
- points: int
- bg: PImage
- pot: PImage

+ getPoints(): int
+ isActive(): boolean
+ display()
+ keyPressed()
+ stopSlider()

Game

-curry: Curry
-cake: Cake

+setup()
+draw()
+mousePressed()
+mouseDragged()
+keyPressed()

Ingredients

- order: ArrayList<String>
- possible: String[]
- flour: PImage
- sugar: PImage
- salt: PImage
- egg: PImage
- baking: PImage
- milk: PImage
- strawberry: PImage
- butter: PImage
- background: PImage

+ getSize(): int
+ display()
+ mousePressed()

Vegetable

-mass: int
-location: PVector
-velocity: PVector
-gravity: PVector
-caught: boolean
-col: int

+move()
+bounce()
+isCaught(): boolean
+caught()
+display()

Curry

- peelerSize: int
- vegetables: ArrayList<Vegetable>
- startPosition: PVector
- distanceStirred: float
- requiredDistance: float
- points: int
- games: String[]
- current: String
- frames: int
- levelTime: int
- levelDone: boolean
- level: int
- pg: PGraphics
- peelGraphics: PGraphics
- peeledArea: float
- totalArea: float
- requiredPeelPercentage: float
- peeledPart: ArrayList<PVector>
- done0: PImage
- done1: PImage
- end: PImage
- starList: ArrayList<Gif>
- c: Pour
- b: Boil

+ getCurrent(): String
+ catchIngredients()
+ display()
+ peelPoints(percent: float)
+ drawUnpeeled()
+ peel(x: float, y: float)
+ isPeeled(p: PVector): boolean
+ mousePressed()
+ mouseDragged()
+ keyPressed()
+ points(): int

Cake

current: String
- frames: int
- levelTime: int
- ChickenList:
ArrayList<Chicken>
- starList: ArrayList<Gif>
- ing: Ingredient
- a: int
- b: int
- index: int
- timeleft: int
- temp: int
- done: boolean
- c: Decor
- games: String[]
- level: int
- levelDone: boolean
- done0: PImage
- done1: PImage
- background: PImage
- end: PImage

+ getCurrent(): String
+ setPoint(p: int)
+ setCurrent(p: String)
+ display():
+ pointChicken(): int
+pointIng(): int
+ pointBake():int
+mousePressed()

Decor

- strawberry: PImage
- frosting: PImage
- sprinkle: PImage
- cherry: PImage
- chocolate: PImage
- background: PImage
- cake: PImage
- kiss: PImage
- straw: PImage
- frost: PImage
- current: String
- strawberryP: ArrayList<PVector>
- frostingP: ArrayList<PVector>
- sprinkleP: ArrayList<PVector>
- cherryP: ArrayList<PVector>
- chocolateP: ArrayList<PVector>

+ display()
+ fin()
+ mousePressed()

Chicken

-location: PVector
-velocity: PVector
-caught: boolean

+move()
+bounce()
+stopIfClicked()
+isCaught(): boolean
+display()

4. How does it work?

First, the player would click the start button, which would generate the background of the game. The player would then press 1 or 2 to select either the curry or cake level. Afterward, the player will mainly use their mouse to control and select different aspects of the game. The player also will be able to input their selections, such as where to click or when to press a button. Ultimately, the objective of the game is to perfectly make a dish by collecting points in each step, or "mini-game."

5.

Current Functionalities:

- Catch the chicken/chicken class
- Catch vegetables
- Peel vegetables
- Select Ingredients
- Boil (slider)
- Bake (addition)
- Decorating

Meeting #1

We had issues deciding on the methods we used to calculate certain parts of the game. We also had issues finding ways to accomplish certain parts of the game with Processing, such as user input, while still making it feel like a game. There were also some issues with classes, such as classes not being found, and inheritance as it seems different from normal Java code. However, we found that the solution to the file not found issue was that of our main "Game" file. Instead of the Game file having a Game class, it should just be used to implement setup and draw methods without a class. We also ran into the issue of how to run the different steps, but we solved this by calling the steps in the constructor, so when a food object is instantiated, the steps are run.

Meeting #2

We also had some issues with the egg class because there were conflicts with how some methods were called in the draw() method. It made it hard to progress through the eggs one at a time. However, we got the ingredients class done, where players need to click through the correct order of ingredients based on the randomly generated string of 20 ingredients.

We also had some problems with the curry with the peel because the player was able to "peel" past the vegetable. However, we solved this problem by making sure the point was the same distance from the point as the radius. We also had issues with the player being able to peel the same part again, so we tried to use an array list, but it was not right. We then used another method that used a tolerance to compare instead.

Final

We had some problems with the transition screen. It would only play the transition screen for a second and move on without pressing “n”. To fix this, instead of using a boolean, we used a current string and compared it to know when to display the transition screen. We also had issues transitioning between games so we put all the games into an array and also compared it to know which game we should play. Another issue we had was trying to display the decorated food on the done screen at the end. We fixed this by creating the fin() method which just redrew all of the images. Another problem we had was that our user input was not working. Our solution to this was to have mousePressed() or keyPressed() in the level classes, such as ingredient, decor, boil, and pour. We then called them within the curry or cake class by creating an object of one of the level classes.

6. Log:

Linda implemented each level of the cake game. Elizabeth implemented each level of the curry class. Linda implemented the following functionalities: catching chickens, choosing ingredients, baking the cake, and decorating the cake. Elizabeth implemented the following functionalities: catching ingredients, peeling vegetables, boiling, and topping the curry. Linda also drew the cover and transition art.