Period 6
Group Members: Jessica Yu and Ayesha Talukder
Group Name: Suika Fruits
Project Title: Suika Game

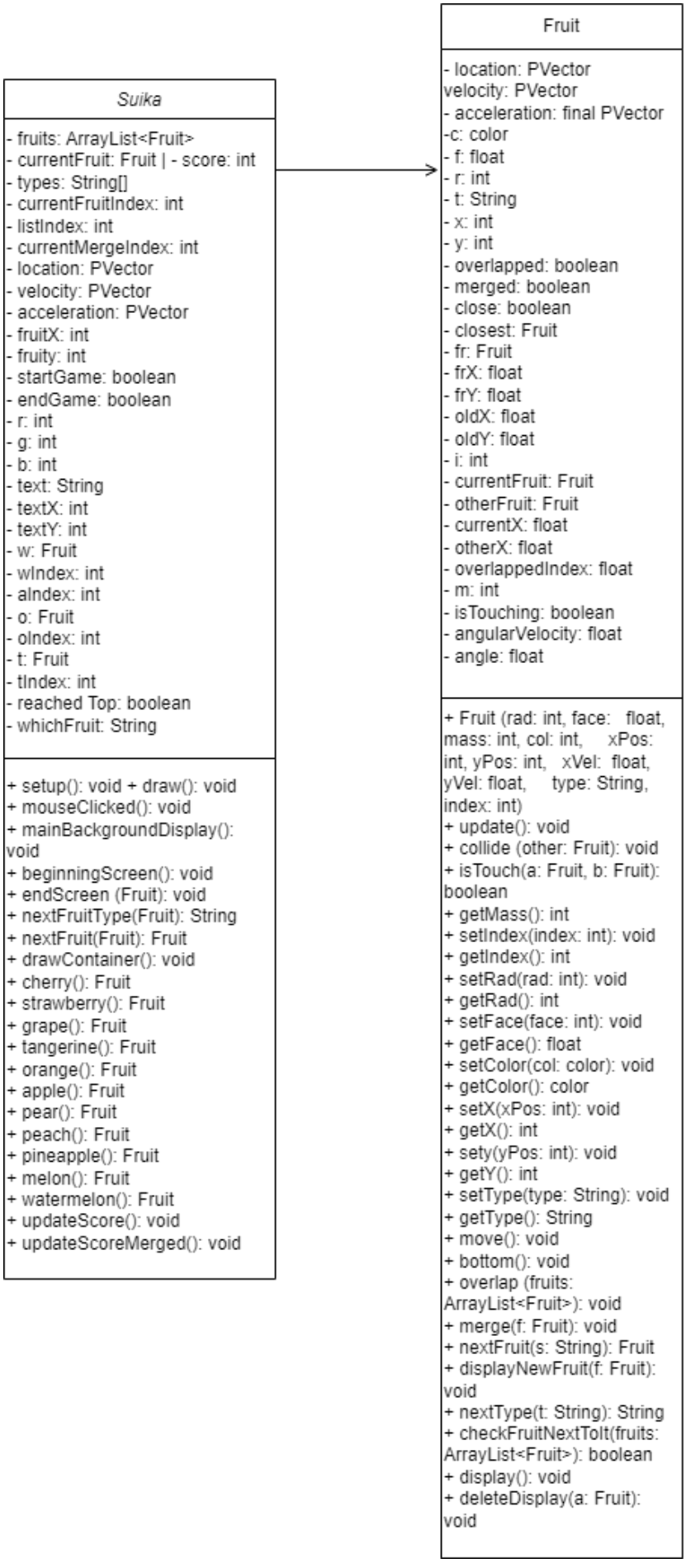
Description:

- Inspired by the actual Suika Game. You drop random fruit from the top of the screen into a container and when it touches something the same fruit as the fruit you dropped, it merges and turns into a slightly larger fruit type. Otherwise, if the fruit doesn't merge, it stacks on the fruit beneath it and depending on its placement onto the fruit, it could roll off. It adds points whenever the fruit drops and points vary depending on the size of the fruit. Whenever fruits merge, more points get added. The game ends when no more fruit can be dropped into the container (the fruits extrude off of the top of the container).
- Libraries: None needed so far
- Functionalities:
    - Setup: sets up the size of the screen for the game to (800, 800) and then sets up the Fruits ArrayList
    - Draw: displays the different backgrounds (Start Screen and Game Screen) and features of the background (container, points, fruit)
    - MouseClicked: click the right button to change the fruit that can be dropped and click the left button to drop the fruit
    - NextFruitType: chooses a random fruit from cherry to orange that will be dropped next
    - FruitType: spawns the fruit that was chosen by the NextFruitType method
    - Fruit Methods: draws the different types of fruits using parameters from the Fruit constructor
    - UpdateScore: updates the score whenever a fruit is dropped
    - UpdateScoreMerged: update score whenever a fruit gets merged
    - Set Methods: set the parameters of the Fruit object
    - Get Methods: gets the parameters set
    - Move: updates the location of the fruit using velocity and acceleration
    - Bottom: ensures that the fruits getting dropped never pass the bottom of the container
    - Overlap: prevents the fruits from overlapping when dropped. Checks the coordinates in which the current fruit got dropped and uses a for loop to find the closest fruit in the array. Depending on its x coordinate placement, the new fruit will be dropped in relative to the part of the curvature of the fruit beneath it.

- Merge: combines the two fruits together into the fruit of the next index of the Array (a slightly larger fruit) and deletes the two old fruits. The merged fruit spawns slightly above where the old fruits were, but drops and calls the overlap method again to ensure it's in an empty space.
- NextFruit: displays all the fruits in the game
- DisplayNewFruit: displays the new fruit after merging occurs
- NextType: assists the NewFruit method by obtaining the right fruit index
- Collision: Checks the coordinates of the fruit and other fruit in relation to it. They will bounce off each other if touched.
- Display: displays the fruits and gives them different facial expressions depending on which fruit is being dropped. Makes it so that the faces are constantly switching as you play the game.
- DeleteDisplay: deletes the fruits presented on the start screen

UML:

- Suika Class
- Fruit Class


Suika
- fruits: ArrayList<Fruit>
- currentFruit: Fruit | - score: int
- types: String[]
- currentFruitIndex: int
- listIndex: int
- currentMergeIndex: int
- location: PVector
- velocity: PVector
- acceleration: PVector
- fruitX: int
- fruity: int
- startGame: boolean
- endGame: boolean
- r: int
- g: int
- b: int
- text: String
- textX: int
- textY: int
- w: Fruit
- wIndex: int
- aIndex: int
- o: Fruit
- oIndex: int
- t: Fruit
- tIndex: int
- reached Top: boolean
- whichFruit: String
+ setup(): void + draw(): void
+ mouseClicked(): void
+ mainBackgroundDisplay(): void
+ beginningScreen(): void
+ endScreen (Fruit): void
+ nextFruitType(Fruit): String
+ nextFruit(Fruit): Fruit
+ drawContainer(): void
+ cherry(): Fruit
+ strawberry(): Fruit
+ grape(): Fruit
+ tangerine(): Fruit
+ orange(): Fruit
+ apple(): Fruit
+ pear(): Fruit
+ peach(): Fruit
+ pineapple(): Fruit
+ melon(): Fruit
+ watermelon(): Fruit
+ updateScore(): void
+ updateScoreMerged(): void
Fruit
- location: PVector
velocity: PVector
- acceleration: final PVector
-c: color
- f: float
- r: int
- t: String
- x: int
- y: int
- overlapped: boolean
- merged: boolean
- close: boolean
- closest: Fruit
- fr: Fruit
- frX: float
- frY: float
- oldX: float
- oldY: float
- i: int
- currentFruit: Fruit
- otherFruit: Fruit
- currentX: float
- otherX: float
- overlappedIndex: float
- m: int
- isTouching: boolean
- angularVelocity: float
- angle: float
+ Fruit (rad: int, face: float, mass: int, col: int, xPos: int, yPos: int, xVel: float, yVel: float, type: String, index: int)
+ update(): void
+ collide (other: Fruit): void
+ isTouch(a: Fruit, b: Fruit): boolean
+ getMass(): int
+ setIndex(index: int): void
+ getIndex(): int
+ setRad(rad: int): void
+ getRad(): int
+ setFace(face: int): void
+ getFace(): float
+ setColor(col: color): void
+ getColor(): color
+ setX(xPos: int): void
+ getX(): int
+ sety(yPos: int): void
+ getY(): int
+ setType(type: String): void
+ getType(): String
+ move(): void
+ bottom(): void
+ overlap (fruits: ArrayList<Fruit>): void
+ merge(f: Fruit): void
+ nextFruit(s: String): Fruit
+ displayNewFruit(f: Fruit): void
+ nextType(t: String): String
+ checkFruitNextToIt(fruits: ArrayList<Fruit>): boolean
+ display(): void
+ deleteDisplay(a: Fruit): void

4. How it works

- Step 1:
    - Click enter to get past the start screen
- Step 2:
    - Drop a fruit at the desired position using the left click button on your mouse and by hovering over the area you want it to be dropped
    - If fruit touches similar fruit, the fruits will merge, else the fruits will stay in their respective places or move

  Step 3:
    - After the fruit has dropped, click the right button on your mouse to change the fruit type.
    - Only press this button once.
    - The fruit that is able to be dropped will be displayed on the top right corner.

  Step 4:
    - Repeat! Try to merge as many fruits as possible and to keep the fruits within the container!
    - When the container can no longer hold any more fruit, a Game Over Screen will appear, also showcasing the amount of points you got in total.

Objective: get the most amount of points possible without completely filling in the container.
Keys To Use:

- Click enter to get past the start screen.
- Click the left button on the mouse to drop.
- Click the right button on the mouse to switch fruit (right click only once)
- Move mouse pointer to hover over the spot in which you want to drop the fruit

6. Log

Ayesha

- Fruit Class
    - Developed the Fruit Constructor
    - added Getters and Setters
    - Moved method which updated the positions of the balls and got them to move
    - Helped with merge logic

        - Implemented nextFruit method to get the nextFruit to be displayed once two fruits are merged together
        - Implemented displayNewFruit to display that fruit
        - Implemented nextType to get the nextType of fruit that the merged fruit will become and use that to create the nextFruit
    - Added collision logic to game to get fruits to roll off of each other
- Suika Class
    - Helped change mouseclicked method to get the fruit to drop in and show the type of nextFruit that would be dropped in the corner of the screen
    - nextFruitType method to get the nextFruit that would be dropped
    - nextFruit to return the nextFruit to be dropped
    - drawContainer method to draw a container for the fruits in the background of the game
    - Various fruit constructor methods to easily create different fruits
        - Ex: cherry(), strawberry(), etc.
    - Update Score method to update the total score when a fruit is dropped
    - EndScreen method for when the container is full

Jessica

- Fruit
    - Added parameters to Fruit method and created get/set methods
    - Fixed the Move method
    - Implemented Bottom method which makes sure fruits don't go outside the bottom of the container
    - displayMethods, displayNewFruit and display to get fruits to display correctly and create the faces on the fruits
    - DeleteFruitMethod to provide a smooth transition between the start and game screen
    - Created NextFruit method to return all the fruits when merged
    - checkFruitNexttoIt which looks at the fruit next to it and returns boolean to see if they are close enough to overlap
    - Overlap methods which helped the fruits not overlap with each other when they touch each other
    - Provided base code for merge and then debugged after Ayesha implemented
    - Implemented collide method
    - Helped aid in collision detection logic and debugged
- Suika Class
    - Implemented setUp and draw methods (location and size of text, color of background, etc)

- Implemented mouseClicked to get the nextFruit to drop in correct xPosition and set the correct boundaries for fruit dropping
- Implemented mainBackground display to display the main game
- Created the beginningScreen()
- Implemented updateScoreMerged() to update the score when two fruits merge together
- Added Various fruit constructor methods to easily create different fruits
    - Ex: cherry(), strawberry(), etc.