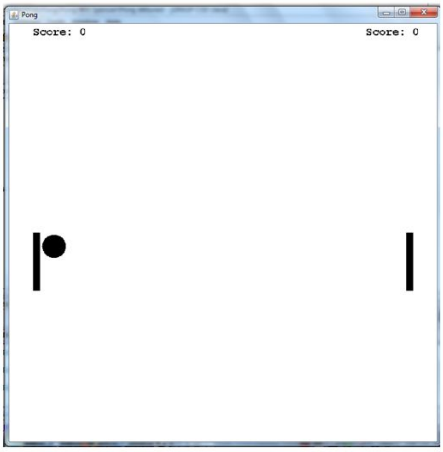
Pong Planning Document



It is time to write the classic Pong game. *Pong* is a two-dimensional sports game that simulates table tennis. The player controls an in-game paddle by moving it vertically across the left side of the screen, and can compete against either a computer-controlled opponent (AI) or another player controlling a second paddle on the opposing side. Players use the paddles to hit a ball back and forth. The aim is for each player to reach eleven points before the opponent; points are earned when one fails to return the ball to the other. You only need to do the two-player game, with colors, background lines or shapes, and a scoreboard.

The packet has you modifying Lab14. Open Lab14 and **plan** what you must do.

Write your name(s): **Anirudh Mantha, Rohan Ramprasad, Aurora Richard**

1. **Describe** (in words) how your two-person Pong will work, including keeping score. What will the user see? How will the user interact with your Pong?

USER VIEW: (Everyone)

- Two paddles on the right side and left side of the screen that move vertically but cannot move laterally.
- Ball will spawn in the middle and will propel towards one side, and the user should try to move the paddle to obstruct the ball's path and keep it from hitting the edge of the screen.
- The middle of the frame will be designated by a dotted line.
- If the ball hits the edge of one side it will respawn in the center and move towards the direction it originally came from.
- There will be a score that is displayed on the top right and top left (the top right score corresponds to the right players points and the top left score corresponds to the left players points)
- The score will increase by one point every time the ball passes the opponent's paddle and runs into the side of the frame. For example, if the ball passes the left player's paddle and hits the edge of the frame, the right player will get one point.
- After the ball bounces off a paddle, it will continue on its trajectory until it gets to the other side of the screen where the opponent should be ready to hit the ball back.

- The opponent should use their arrow keys or “W” and “S” keys to move their paddle to obstruct the ball from touching the edge of the screen on their side.
- Once one player scores 11 points an end game screen is displayed. It will say “GAME OVER” and which player has won. Either the right player has won or the left player has won. There will also be an option to play again.

INTERACTION: (Everyone)

- 2 players with paddles on left and right side respectively
- Paddles can move vertically
- The player on the left side uses the “W” and “S” keys. “W” moves the paddle up and “S” moves the paddle down.
- The player on the right side uses the arrow keys. The up arrow key moves the paddle up and the down arrow key moves the paddle down.
- The users will move their paddles to try to block the ball from hitting their edge of the screen
- Players can change the ball speed and trajectory, by clicking the right side of the mouse
- Players can reset the ball to the center position by pressing shift and left click at the same time
- To reset the game completely players can left click the mouse. This action resets both scores to 0
- If the users would like to play again they can left click on the end game screen.

2. Give an **example** of your Pong, which in this case could be a screen-shot of your game. (Anirudh/Aurora)

3. List the **classes** that you need to modify (or perhaps the entirely new class that you need to write). In each class, list the **method headers** (including arguments) that you need. **(Everyone)**

We are not making a paddle resource class and instead are going to instantiate two bumpers one being named "PaddleL" and the other being named "PaddleR."

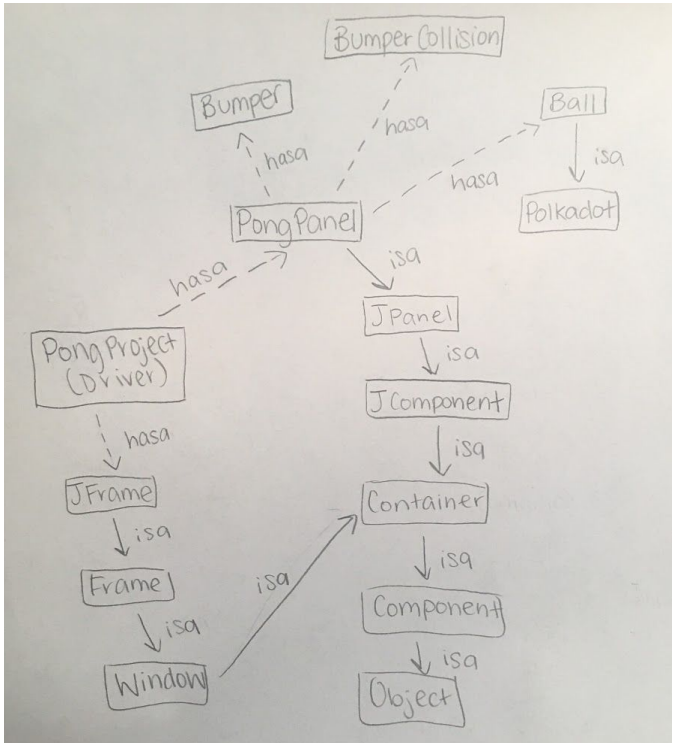
- We have to create and add a Driver Class named **PongProject**; this driver has to be linked with PongPanel (it will run PongPanel)
- BumperPanel will get modifications and be called PongPanel
    - Under public PongPanel, instantiate both of the paddles; which are bumpers with a custom constructor
    - Also, add private variables for both of the paddles and for both of the hits (scores).
    - We will use the Key class
        - keyPressed will be used ("W", "S", up, and down keys)
            - This will make the paddles move up and down and make sure they don't go out of the frame
    - We will use Listener
        - Action Performed will be the method that makes the balls move and count points for both sides
        - Action Performed will be where the dotted center line code is placed
        - Action Performed will have the end game screen code as well
            - Will have two if-statements, one for the right player winning and one for the left player winning
            - Will also include a left click to reset option, to play again
        - At the very end of action performed we also put if statements that determine how the ball resets when it passes the barrier (should start at center line)
            - This also keeps the speed of the ball slow and random
    - We will use the Mouse class
        - If there is a right click it will change the trajectory and speed of the ball
        - If there is a shift click then this will reset the ball to the center again
            - This component is not required to play pong but it certainly helps in the case of a stalemate or the ball being stuck bouncing in a certain area
        - Else if there is a left click that will reset the whole game
            - The score will go to 0 and the ball will start at the center again

Note: There will be smaller changes as well but we didn't list them because they are miniscule.

4. Write an important **algorithm** in your Pong, in pseudo-code, diagrams, or flowchart. *Do not write Java code.*

In our pong panel there are several algorithms we write along with several components - **crucial parts will be bolded**

UML DIAGRAM: (Aurora)



PSEUDOCODE: (Everyone)

1. Define private variables - **including the scores on both sides**
2. Inside Pong panel

    1. Create buffered image
    2. Define the ball
    3. Make the ball jump
    4. Define Paddle for Right and Left
    5. Make sure ball is outside of the Paddle
        - Do that by using if statements and boolean inBumper
    6. Implement Mouse and Key Listener

3. Inside paintComponent
    1. Draw the Image
4. Listener
    1. Clear the buffer and move the ball
    2. Draw the ball and paddles

3. Make sure if the ball goes on either the right or left side it counts a point for that respective side and resets the ball location

- Do this by using if statements and the modifier methods for the Ball class

- Use the increment to make score go up

- use math.Random()

4. Draw the dotted line in the center using a for loop

5. This is also making an End game screen

- Have 2 if statements one for right winning, and one for left winning

- These if statements will draw text in the condition of getting more than 10 points.

4. Update the scores on the buffer

5. Mouse

1. If there is a right click Make sure the speed and direction of the ball is changed

2. If there is a shift click make sure that the ball starts in the center again and is in a different speed

3. If there is a left click make sure that the ball resets and the scores also reset restarting the whole game

6. Key

1. If the left paddle is eligible make it move 20 pixels down when the S key is pressed and make it move up 20 pixels if the W key is pressed

2. If the right paddle is eligible make it move 20 pixels down when the down arrow is pressed and make it move up 20 pixels when the up arrow is pressed