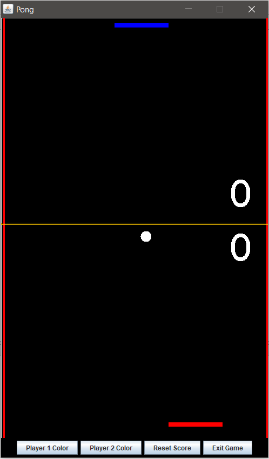
Pong

By Jonpaul Addorisio



### Introduction:

Pong is an age old classic arcade game, developed by Allan Alcorn, for the Atari gaming console in where a game of table tennis is virtually simulated. This java based program is an imitation of that very same game. In Pong, two players are placed at opposing ends vertically of the virtually created tennis board. The goal of this game is to get the pong ball past the opposing players paddle. Each player can move his or her paddle left or right, so long as it stays in bounds of the board. When the game first starts off the ball is to be placed in the middle heading towards the top player direction. After this initial setup, the ball will be placed on the paddle of the scoring opponent to represent a serve. Normally, in the original pong one player would have to score to 11, however in this version the player can score an infinite amount of time. Ending the game is completely at the user’s discretion. Pong provides the players with 2 options, Changing the paddle colors and resetting the score. Overall, pong is a fun game to play if a person is feeling nostalgic about the 70’s.

### Development Process:

When designing this mock java version of the game of pong, I decided to keep the number of files to a relatively small amount of 3: The main(Pong.java), game layout(game.java), and in game options(Options.java). I broke the development process down into 3 stages. In the first stage, the in-game mechanics were designed. The first step of this stage beside declaring potential variable that would be needed was to create a paint method for creating the graphics of the game. In this method, the ball, paddle, background, net, and score, were created using the various draw methods from Java 2D API. After creating the graphical parts of the game, event interface such as key listener and action listener were implemented into the java class. In the key listener event, the mechanics behind moving the paddles were created, so that when a user hit the corresponding key his or her paddle would move within the game boundaries. Next step was to use the action event to make the game “live”. In this event rectangle objects were created around the paddles and ball so that the intersection between the two could be detected. Whenever an intersection point between a ball and paddle occurred the ball direction would inverse itself. Also in this event code was written so that the ball would bounce off the sides of the screen. When the ball was to pass an opposing paddle, it would reset the ball position and tally a score for the other player.

The next stage of the development process was to create some added options for the pong players. In this stage, a separate class file was created to hold buttons and various return methods for players to choose their paddle color, reset the score, and exit the game. First step to this stage was declaring 4 buttons and setting their focus ability to false so that clicking on them would not interrupt the game panel. Before adding separate events to each button, a method was written that opened JColorChooser and saved the color chosen into an instance variable. After creating this method action events were added to each color button so that the new color would be saved to separate instance variables. The Next step was to create the code for the reset button and exit button. For the exit button, it was simple as adding a “system exit” method to the action event. As for the reset button things became a little complicated so I decided to user an integer variable to store a reset value for when the action was to be fired. This value was key to a code that would be written in the game file for determine when to reset the scores to zero.

After completing this stage, the stage I like to call “Tying In stage” had commenced. In this stage, the main file pong.java was created so that all the code would “run”. In this stage, the first step was declaring the frame in the main file and then adding each of the prior files to this frame. After adding the panel file to the frame, new parameters had to be created in the constructer of the game file to access options method that were created in the options file. By doing this it allowed the game file to access return color methods to repaint the paddles to a color he or she desires and reset the player scores. After all this was said and done some post creation tinkering began. At this step, I reviewed my code to see what I could make better. I created a new parameter in the constructer of the options file so that the frame component could be imported. The reason for doing this was so that the JColorChooser frame would open in the center of the game instead of the bottom left. The reason the frame opened in the bottom left was due to the position panel component that JColorChooser initially used as its open point. After solving this issue, I did some tinkering around with the ball in the game file. At this step, I added a way for the ball speed to increase with each “hit” to keep the game competitive. Once tinkering had been completed the game was ready to play free of errors.