For our final project, we created a desktop version of the popular mobile game Flappy Bird. This game required us to implement concepts like variable image display, designing our own images, collision detection, scorekeeping, and UP/DOWN motion controls to move the bird, AKA the player.

For our project, we decided to implement class hierarchy by creating a base class entitled GameObject. Extending from GameObject, we have classes for bird, topPipe, bottomPipe, and scoreBox. The GameObject class creates a default image JLabel, initializes object velocity, and creates two constant ints for screen height and screen width. It also returns a variable topPipeHeight that is randomly reassigned every time a new top pipe is created, changing the height of the pipes, thus increasing the difficulty. Inside each of the classes that extend from GameObjects, the images were set to their respective pictures (using try/catch and exception handling) and the location on the screen of the images was set dependent on the locations of the other objects on the screen. For example, the top pipe Y-coordinate is set to the top of the screen, while the scorebox Y-coordinate is set to the height of the top pipe. We also added borders around all of our images, partially for bound-checking during tests, but also to make it a little easier for the user to get through the game and have more of an understanding of where the image actually ends on the screen.

The bird class takes most of its properties from GameObjects, but the one function we had to create was the move function. The move function takes the velocity (to increase/decrease difficulty) and continuously sets the location of the image by checking how the velocity and player movement have impacted the bird’s location. The faster the velocity, the faster the bird goes, making it harder for the player to get through the game. To make the bird move up and down, we had to implement KeyListener. Based on the ASCII codes of the UP/DOWN arrows, the KeyPressed function takes the user input and changes the velocity accordingly. If the user hits UP, the velocity becomes positive, and if the user hits DOWN, the velocity becomes negative, making the brid travel downwards.

By implementing the collide function, we dealt with score keeping and ending the game all in one. The collision function continuously checks if any items in the GameObjects arraylist are interacting with one another. This helped us fix bugs during testing that had to do with panel coordinates being different, but also works to check constantly if the bird hits a pipe or a scorebox. If the bird goes through the scorebox, the scoreTally will go up until the bounds of the two GameObjects do not interact anymore; however, if the bird bounds interact with the pipe bounds, the gameover function is called. Gameover pauses the game and outputs a new window that states “you lost” and then displays the score. If the user can get to the end of the screen, which is around 80 points, the game is paused and the “you win” JFrame pops up. Exiting out of any of the Jframes ends the game.