Final Project

Object Oriented Programming

{Java}

Submitted by:

Ali Jamshed

Roll #:

2k23-BsCs-251

Section:

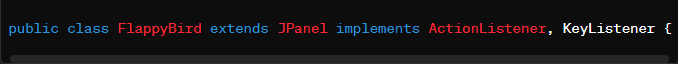
Green

Starting Explanation of Code:

Let's dive deeper into the details of each part of the code to understand how it all works together to create the Flappy Bird game.

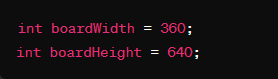
**Class and Object Definitions:**

1. **Class Declaration:**

****

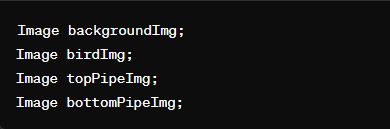
* **FlappyBird** extends **JPanel** to create a custom panel for rendering the game.
* Implements **ActionListener** to handle timer events.
* Implements **KeyListener** to handle keyboard input.

1. **Board Dimensions:**



* **boardWidth** and **boardHeight** define the dimensions of the game window.

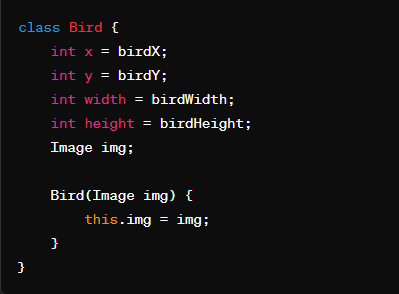
1. **Image Variables:**



* These variables will hold the images for the background, bird, and pipes.

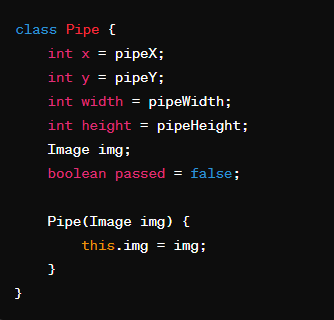
**Bird and Pipe Classes:**

1. Bird Class:



* **Bird** class defines the properties of the bird, including its position (**x**, **y**), size (**width**, **height**), and image (**img**).
* An instance of **Bird** is created with the bird image.

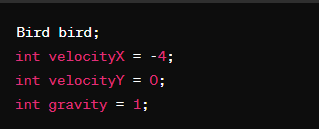
1. **Pipe Class:**



* **Pipe** class defines the properties of the pipes, including position (**x**, **y**), size (**width**, **height**), image (**img**), and a **passed** flag to track if the bird has passed the pipe.
* Pipes are created with images for the top and bottom pipes.

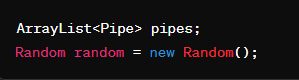
**Game Logic and Variables:**

1. **Bird Instance:**

****

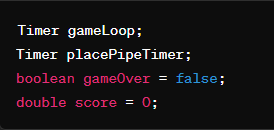
* bird is an instance of the Bird class.
* velocityX controls the speed of the pipes moving to the left.
* velocityY controls the vertical speed of the bird.
* gravity simulates gravity affecting the bird's vertical movement.

1. **Pipes:**



* **pipes** is an **ArrayList** to hold the **Pipe** objects.
* **random** is used to generate random values for pipe placement.

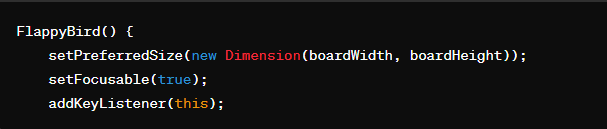
1. **Game Control Variables:**



* **gameLoop** is a timer that controls the main game loop.
* **placePipeTimer** is a timer that controls when new pipes are added.
* **gameOver** indicates if the game has ended.
* **score** keeps track of the player's score.

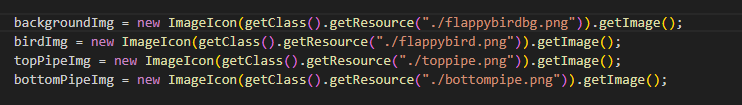
**Constructor:**

1. **Setting Up the Game:**



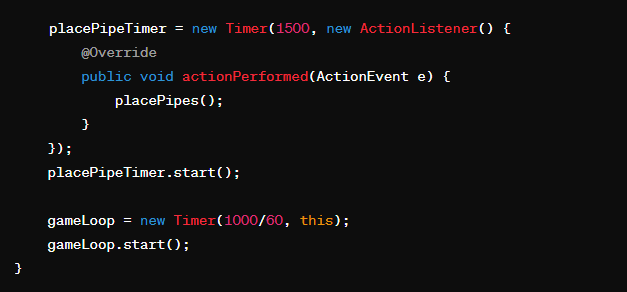
* Sets the preferred size of the panel.
* Makes the panel focusable to receive keyboard input.
* Adds a key listener to handle user input.

1. **Loading Images:**



* Loads images for the background, bird, and pipes from the resources.

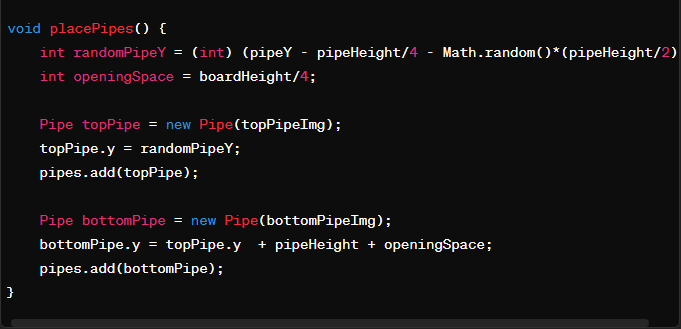
1. **Timers:**



* **placePipeTimer** triggers every 1.5 seconds to add new pipes.
* **gameLoop** triggers 60 times per second to update and repaint the game state.

**Pipe Placement Method:**

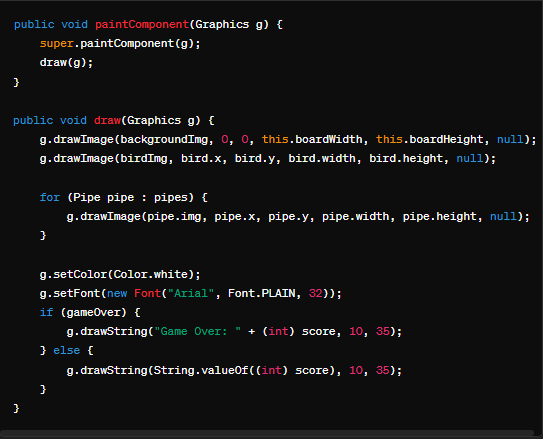
1. **Place Pipes:**



* Generates random positions for the top and bottom pipes to create a gap.
* Adds the pipes to the **pipes** list.

**Drawing Method:**

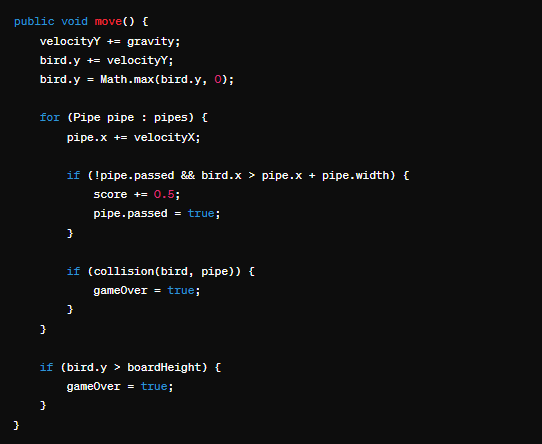
1. **Drawing Components:**

****

* paintComponent and draw methods render the game elements: background, bird, pipes, and score.
* The bird and pipes are drawn using their respective images.
* The score is displayed in the top-left corner.

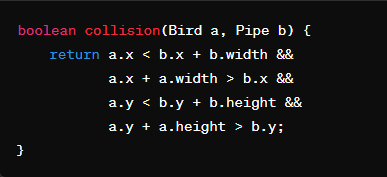
**Movement and Collision Detection:**

1. **Move Method:**

****

* Updates the bird's vertical position by applying gravity.
* Moves pipes to the left.
* Checks if the bird has passed a pipe to update the score.
* Checks for collisions between the bird and pipes.
* Ends the game if the bird hits the ground.

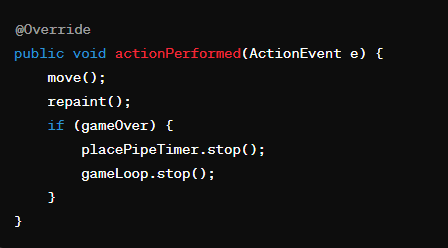
1. **Collision Detection:**



* Checks if the bird intersects with a pipe by comparing their positions and sizes.

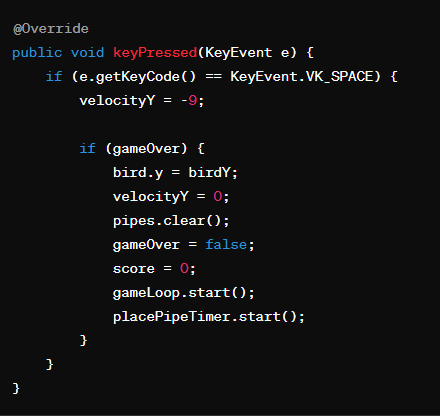
**Event Handling:**

1. **Action Listener:**

****

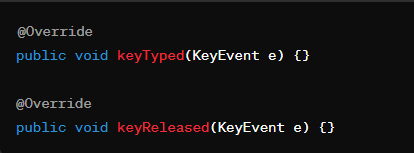
* actionPerformed method updates the game state and repaints the components.
* Stops the timers if the game is over.

1. **Key Listener:**



* Checks for space bar presses to make the bird jump.
* Restarts the game if the game is over by resetting conditions.

1. **Unused Key Listener Methods:**

****

* keyTyped and keyReleased are required by the KeyListener interface but are not used in this game.

Now the Main Class is Starting:

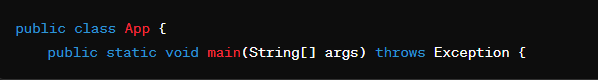
This code creates the main application for the Flappy Bird game using Java Swing. Let's break it down step by step.

**Import Statement:**



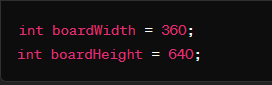
* Imports the **javax.swing** package, which contains classes for building graphical user interfaces (GUIs) in Java.

**Main Class and Method:**

****

* Defines the App class, which contains the main method.
* The main method is the entry point of the application.

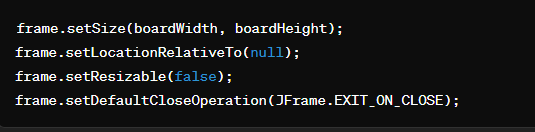
**Frame Setup:**

****

* Defines the width and height of the game window.

****

* Creates a new JFrame instance with the title "Flappy Bird".
* JFrame is a top-level container used to create a window.

****

* frame.setSize(boardWidth, boardHeight): Sets the size of the frame to the specified width and height.
* frame.setLocationRelativeTo(null): Centers the frame on the screen.
* frame.setResizable(false): Disables window resizing to keep the game dimensions fixed.
* frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE): Ensures the application exits when the frame is closed.

**Adding the Game Panel:**

****

* Creates an instance of the FlappyBird class, which extends JPanel and contains the game logic and rendering.

****

* Adds the FlappyBird panel to the frame. This panel contains the game elements and will be rendered within the frame.

****

* Packs the frame to fit the preferred size and layout of its components. This ensures that the frame dimensions are set correctly based on the panel's preferred size.

****

* Requests focus for the FlappyBird panel so that it can receive keyboard input. This is important for the game controls to work.

**Making the Frame Visible:**

****

* Makes the frame visible on the screen, displaying the game window to the user.

**Summary:**

This code sets up and displays the main window for the Flappy Bird game. It creates a JFrame, configures its properties (size, location, non-resizable, close operation), adds the FlappyBird game panel to the frame, packs the frame to ensure correct sizing, requests focus for the game panel to handle keyboard input, and finally makes the frame visible. This structure initializes and launches the game interface.

……………..Finish……………