```
(Pseudocode is not final, and most of it will be modified)
class Characters:
    integer x, y, width, height
    boolean isDead
    string imageName
    double yVelocity, gravity
    integer jumpDelay
    double rotation
    Image image
    Keyboard keyboard

    Characters():
        Initialize properties from configuration file
        Initialize keyboard

    update():
        Update yVelocity and jumpDelay

        if not isDead and space key is pressed and jumpDelay is 0:
            Set yVelocity and jumpDelay

        Update y position based on yVelocity

    getRender():
        Create a new Render object r
        Set r.x and r.y to current x and y values

        Load image if not already loaded

        Calculate rotation based on yVelocity
        Limit rotation to a maximum of 90 degrees

        Apply transformations to r

        Return r
class GameRunner:
    constant PIPE_DELAY = 100
    Properties prop
    Boolean paused, gameover, started
    integer pauseDelay, restartDelay, pipeDelay
    Characters character
    ArrayList<Obstacles> pipes
    Keyboard keyboard
    integer score

    GameRunner():
        Initialize properties and objects
        restart()
```

```
    restart():
        Reset game state and score
        Create new character and pipes

    update():
        Handle game controls
        Update character movement
        Move pipes
        Check for collisions

    getRenders():
        Create an array of renders
        Add background, pipes, foreground, and character renders
        Return the array of renders

    handleStart():
        Start the game if space key is pressed

    handlePause():
        Toggle pause state if 'P' key is pressed

    handleReset():
        Restart the game if 'R' key is pressed

    movePipes():
        Decrement pipe delay counter
        If it's time to create new pipes:
            Create north and south pipes
            Position them vertically
            Add them to the pipe array
        Update pipe positions

    checkForCollisions():
        Check for character and pipe collisions
        Check for ground collision

class Obstacles:
    integer x, y, width, height, speed
    string orientation
    Image image

    Obstacles(orientation):
        Set orientation
        Call reset()

    reset():
        Set width, height, and x position
        Set y position based on orientation
```

```
    update():
        Decrement x by speed

    collides(_x, _y, _width, _height):
        Check for collision between obstacle and given coordinates and
dimensions
        Return true if collision occurs, false otherwise

    getRender():
        Create a new Render object r
        Set r.x and r.y to current x and y values
        Load or retrieve image based on orientation
        Set r.image to the image
        Return r

class GamePanel extends JPanel, implements Runnable:
    GameRunner game

    GamePanel():
        Create a new instance of GameRunner
        Start a new thread

    update():
        Update the game state
        Repaint the panel

    paintComponent(g):
        Call parent's paintComponent method
        Create a Graphics2D object g2D from g
        Iterate over the renders obtained from the game
            Draw the image with the associated transformation if available
            Otherwise, draw the image at the specified x and y coordinates
        Set color to black
        If the game is not started:
            Set font and draw "Press SPACE to start" message
        Else:
            Set font and draw the current score
        If the game is over:
            Set font and draw "Press R to restart" message

    run():
        Loop indefinitely
            Call update method
            Pause the thread for a short duration
        If an exception occurs, print the stack trace

class Render:
    integer x, y
    Image image
```

```
    AffineTransform transform

    Render():
        Empty constructor

    Render(x, y, imagePath):
        Synchronize with default toolkit
        Set x and y coordinates
        Load the image from the given imagePath using Util.loadImage()

class Util:
    static HashMap<String, Image> cache

    Util():
        Create a new empty cache HashMap

    static Image loadImage(path):
        Image image = null

        if cache contains path:
            Return the cached image associated with the path

        try:
            Read the image from the file at the given path
            If the cache does not contain the path:
                Add the image to the cache with the path as the key
        catch IOException:
            Print the stack trace of the exception

        Return the loaded image

class Window:
    static integer WIDTH
    static integer HEIGHT

    main(args):
        Load configuration properties from "config.properties"
        Set WIDTH to the parsed integer value of "window.WIDTH" property
        Set HEIGHT to the parsed integer value of "window.HEIGHT" property

        Create a JFrame
        Configure the JFrame: make it visible, set close operation, center it

        Add a keyboard listener to the JFrame

        Create a GamePanel
        Add the GamePanel to the JFrame
        Configure the JFrame: make it non-resizable, set its size
```

```
Old Pseudocode:
Runner Class:
      Import every library needed
      Create public class GameRunner which implements interface KeyListener{
            Construct a global DrawingPanel with size 1920x1080 named panel
            Construct a global Graphics object named g
            Construct a global Bird object that is yellow with size 100x100
            Main Method{
                  Set panel background to black
                  Construct a new test obstacle with attributes
(100,200,100,600)
                  Set obstacle color to blue
                  Generate obstacle
                  Set bird color to red
                  int temp = 0
                  (loop below is temporary and is for testing movement, will be
                  replaced by methods in the future using KeyListener)
                  for(int i = 0; i<10; i++) {
                        WHILE LOOP BELOW MOVES BIRD DOWN
                        while(temp<300) {
                              Spawn a bird on screen using spawn method
                              Panel sleeps for 5 ms
                              Clear bird using clearBird method
                              Add 5 to temp
                              Set yShift of bird to temp
                        }
                        WHILE LOOP BELOW MOVES BIRD UP
                        while(temp>-200) {
                              Spawn a bird on screen using spawn method;
                              Panel sleeps for 5 ms
                              Clear bird using clearBird method
                              Subtract 5 from temp
                              Set yShift of bird to temp
                        }

                  }

      }
Bird/Character Class:
      Import every library needed
      private int width
      private int height
      private int yShift = 0
      private Color color
      Constructor: public Bird(Color c, int w, int h) {
            width = w
            height = h
            color = c
      }
```

```
	public void spawn(Graphics g) {
		(Code here is temporary, graphics will be improved later)
		Sets color of g to bird color
		g.fillRect(300, 300+yShift, width, height) (draws bird body)
		Set g color to white
		g.fillRect(380, 325+yShift, width/5, height/5)
		Set g color to black
		g.fillRect(385, 330+yShift, 10, 10)
	}
	public void clearBird(Graphics g) {
		g.clearRect(300, 300+yShift, width, height)
	}
	public void moveDown(int pixels, Graphics g, Bird b, DrawingPanel panel)
{
		(Code here is temporary)
		int temp = 0
		while(temp<pixels) {
			Spawn bird
			Panel sleeps 5 ms
			Clears bird
			Add 2 to temp
			Set yShift to temp
		}
	}
	public void moveUp(int pixels, Graphics g, Bird b, DrawingPanel panel) {
		(Code here is temporary)
		int temp = 0
		while(temp>-200) {
			Spawn bird
			Panel sleeps 5 ms
			Clears bird
			Subtract 5 from temp
			Set yShift to temp
		}
	}
	public int getWidth() {
		return width
	}
	public void setWidth(int width) {
		this.width = width
	}
	public int getHeight() {
		return height
	}
	public void setHeight(int height) {
		this.height = height
	}
	public Color getColor() {
		return color
```

```
	}
	public void setColor(Color color) {
		this.color = color
	}
	public void setYShift(int yShift) {
		this.yShift = yShift
	}
}
Obstacle Class:
	private int width
	private int height
	private int x
	private int y
	private Color mainColor
	private Color secondaryColor
	private Color tertiaryColor
	public Obstacle(int w, int h, int xLoc, int yLoc) {
		width = w
		height = h
		x = xLoc
		y = yLoc
	}
	public void generate(Graphics g) {
		g.setColor(mainColor)
		g.fillRect(x, y, width, height)
		g.drawRect(x, y, width, height)
	}
	public int getWidth() {
		return width
	}
	public void setWidth(int width) {
		this.width = width
	}
	public int getHeight() {
		return height
	}
	public void setHeight(int height) {
		this.height = height
	}
	public int getX() {
		return x
	}
	public void setX(int x) {
		this.x = x
	}
	public int getY() {
		return y
	}
	public void setY(int y) {
```

```
		this.y = y
	}
	public Color getMainColor() {
		return mainColor
	}
	public void setMainColor(Color mainColor) {
		this.mainColor = mainColor
	}
	public Color getSecondaryColor() {
		return secondaryColor
	}
	public void setSecondaryColor(Color secondaryColor) {
		this.secondaryColor = secondaryColor
	}
	public Color getTertiaryColor() {
		return tertiaryColor
	}
	public void setTertiaryColor(Color tertiaryColor) {
		this.tertiaryColor = tertiaryColor
	}
}
```