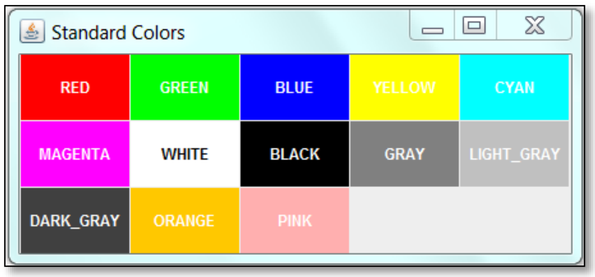
Sprite Paddles

Time required: 45 minutes

Contents

In this tutorial we add a paddle using a Sprite called paddle. The paddle will move up or down when we press the cursor keys, so our program will read from the keyboard.

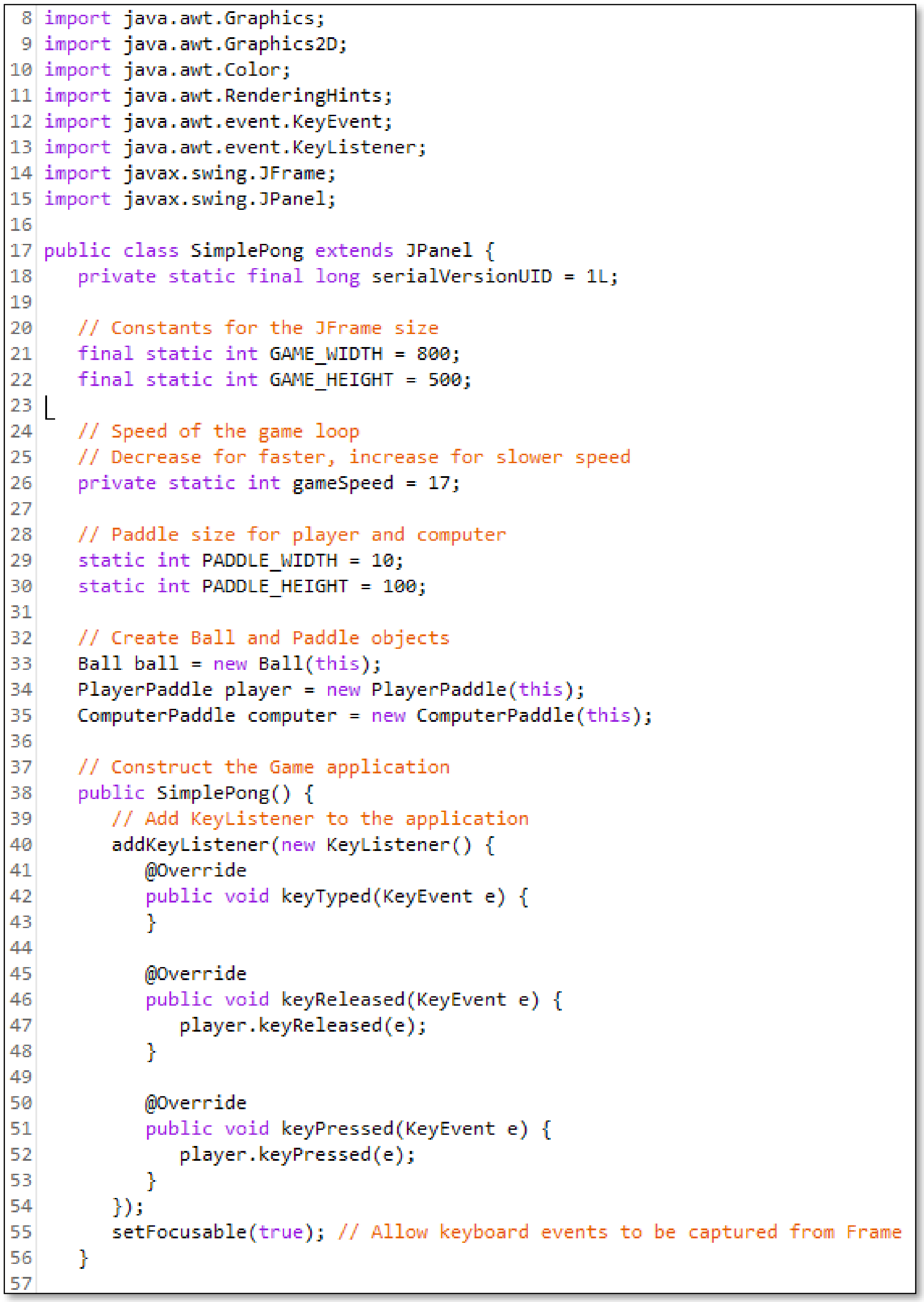
Color provides 13 standard colors as named constants. They are: Color. RED, GREEN, BLUE, MAGENTA, CYAN, YELLOW, BLACK, WHITE, GRAY, DARK_GRAY, LIGHT_GRAY, ORANGE, and PINK.



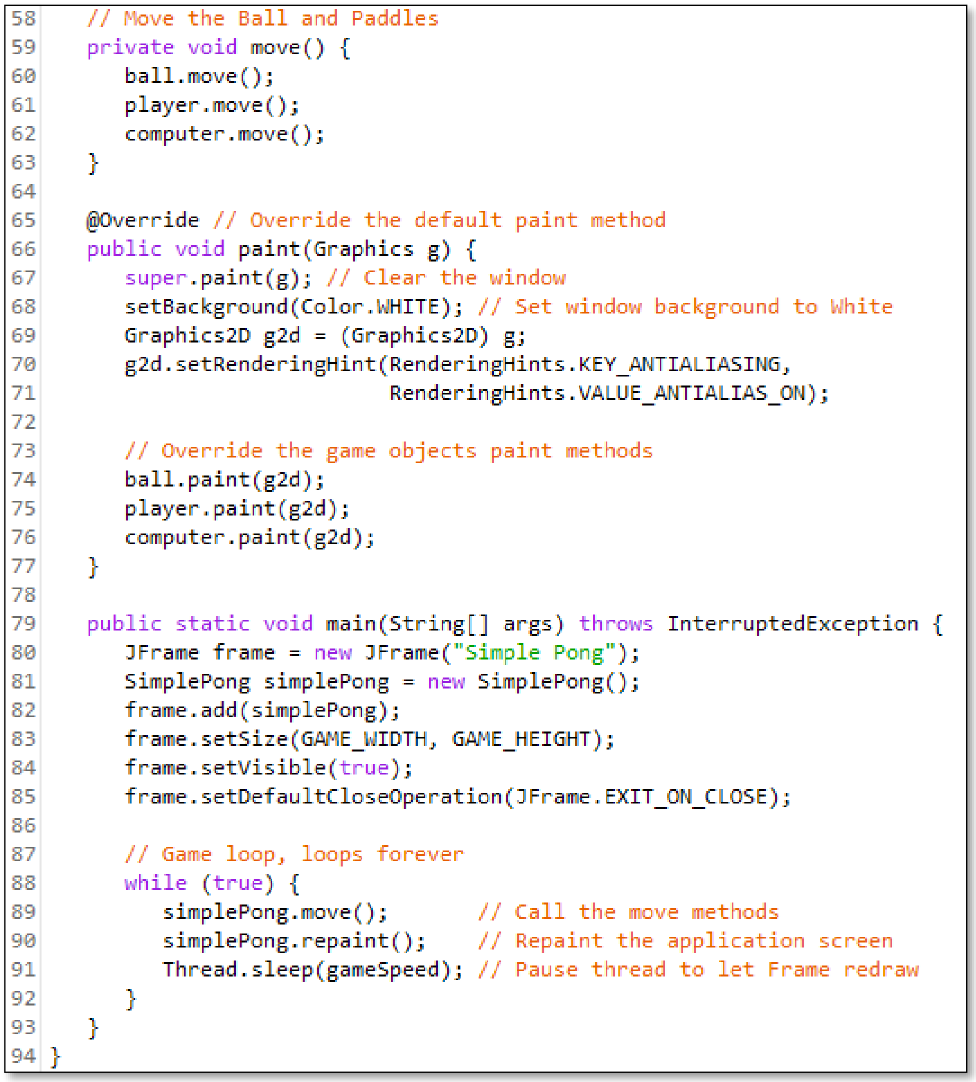
SimplePong Changes

In the **SimplePong** class we create two new objects: a **player** object from the **PlayerPaddle** class and a **computer** object from the **ComputerPaddle** class. In the **move()** method we add a call to **player.move()** and **computer.move()**. In the **paint()** method a call to **player.paint()** and **computer.paint()** is added. Until now, everything is similar to the sprite "Ball", but we have to do something else because the position of the Player Paddle responds to the keyboard.

In the constructor of the class **SimplePong** we can see how we register a listener to capture the events of the keyboard. In the **keyPressed()** method of the listener, we inform the paddle that a key has been pressed by calling **player.keyPressed(e)**. We do the same for **player.keyReleased()**. With this the sprite **PlayerPaddle** will know when a key has been pressed.



```java
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Color;
import java.awt.RenderingHints;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class SimplePong extends JPanel {
   private static final long serialVersionUID = 1L;

   // Constants for the JFrame size
   final static int GAME_WIDTH = 800;
   final static int GAME_HEIGHT = 500;

   // Speed of the game loop
   // Decrease for faster, increase for slower speed
   private static int gameSpeed = 17;

   // Paddle size for player and computer
   static int PADDLE_WIDTH = 10;
   static int PADDLE_HEIGHT = 100;

   // Create Ball and Paddle objects
   Ball ball = new Ball(this);
   PlayerPaddle player = new PlayerPaddle(this);
   ComputerPaddle computer = new ComputerPaddle(this);

   // Construct the Game application
   public SimplePong() {
      // Add KeyListener to the application
      addKeyListener(new KeyListener() {
         @Override
         public void keyTyped(KeyEvent e) {
         }

         @Override
         public void keyReleased(KeyEvent e) {
            player.keyReleased(e);
         }

         @Override
         public void keyPressed(KeyEvent e) {
            player.keyPressed(e);
         }
      });
      setFocusable(true); // Allow keyboard events to be captured from Frame
   }

```



```java
    // Move the Ball and Paddles
    private void move() {
       ball.move();
       player.move();
       computer.move();
    }

    @Override // Override the default paint method
    public void paint(Graphics g) {
       super.paint(g); // Clear the window
       setBackground(Color.WHITE); // Set window background to White
       Graphics2D g2d = (Graphics2D) g;
       g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                            RenderingHints.VALUE_ANTIALIAS_ON);

       // Override the game objects paint methods
       ball.paint(g2d);
       player.paint(g2d);
       computer.paint(g2d);
    }

    public static void main(String[] args) throws InterruptedException {
       JFrame frame = new JFrame("Simple Pong");
       SimplePong simplePong = new SimplePong();
       frame.add(simplePong);
       frame.setSize(GAME_WIDTH, GAME_HEIGHT);
       frame.setVisible(true);
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

       // Game loop, loops forever
       while (true) {
          simplePong.move();       // Call the move methods
          simplePong.repaint();    // Repaint the application screen
          Thread.sleep(gameSpeed); // Pause thread to let Frame redraw
       }
    }
 }
```

Ball Class

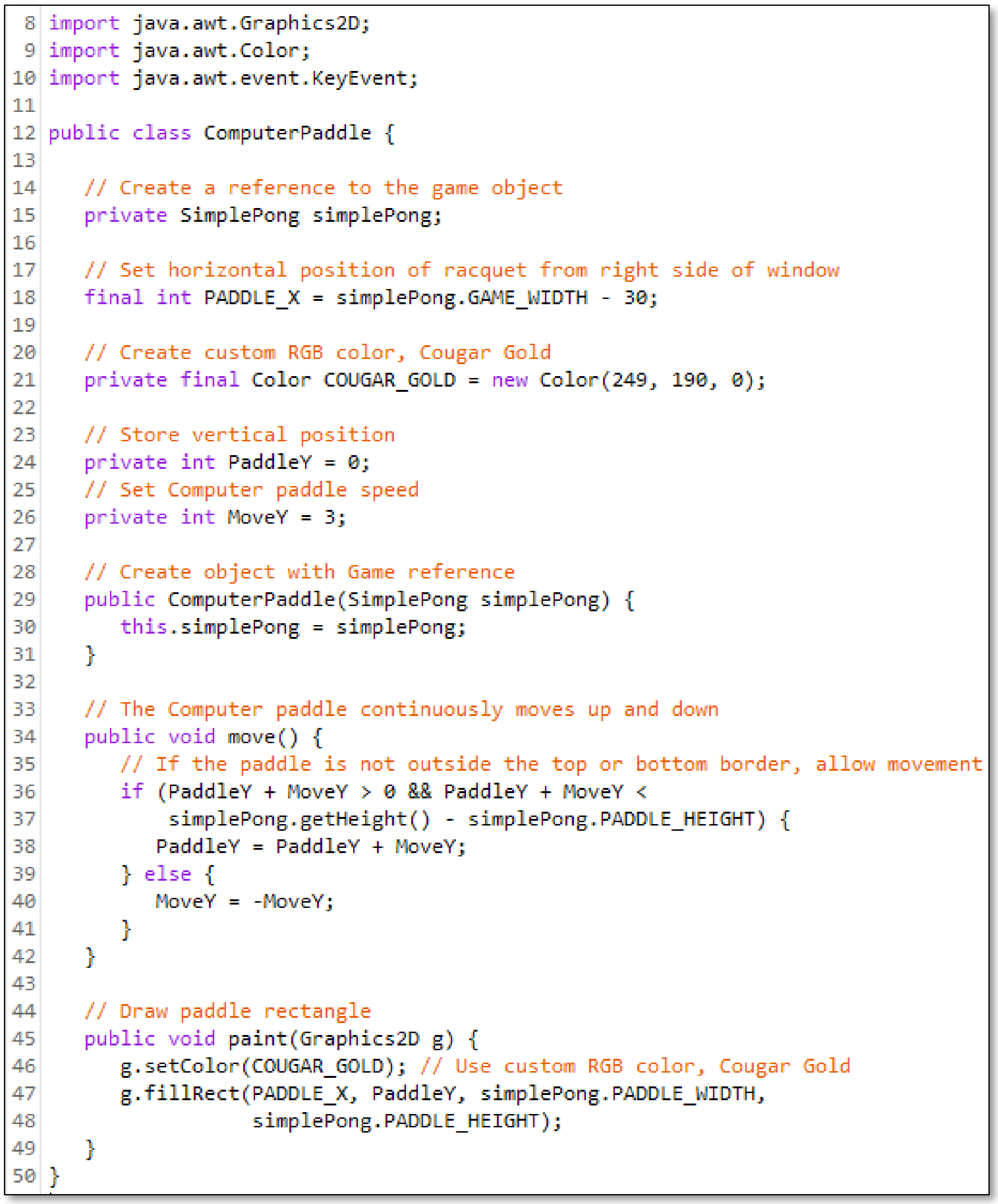
The Ball class doesn’t have any changes. It is shown here for reference.

```java
import java.awt.Graphics2D;
import java.awt.Color;

public class Ball {
   private final int BALL_DIAMETER = 30;

   // Store the ball's x, y location
   private int BallX = 200;
   private int BallY = 200;

   // Store the ball's x, y movement
   private int MoveX = -3;
   private int MoveY = 3;

   // Create Game variable
   private SimplePong simplePong;

   // Create a ball object with a reference to the game board
   public Ball(SimplePong simplePong) {
      this.simplePong = simplePong;
   }

   void move() {
      // Move the ball by adding x, y integers to current location
      BallX = BallX + MoveX;
      BallY = BallY + MoveY;

      // If the ball runs into the left border, reverse direction
      if (BallX + MoveX < 0)
         MoveX = -MoveX;

      // If the ball runs into the right border, reverse direction
      if (BallX + MoveX > simplePong.getWidth() - BALL_DIAMETER)
         MoveX = -MoveX;

      // If the ball runs into the top border, reverse direction
      if (BallY + MoveY < 0)
         MoveY = -MoveY;

      // If the ball runs into the bottom border, reverse direction
      if (BallY + MoveY > simplePong.getHeight() - BALL_DIAMETER)
         MoveY = -MoveY;
   }

   // Paint the ball/circle
   public void paint(Graphics2D g) {
      g.setColor(Color.DARK_GRAY); // Change the paint color to DARK GRAY
      g.fillOval(BallX, BallY, BALL_DIAMETER, BALL_DIAMETER);
   }
}
```

ComputerPaddleClass

Unlike **Ball**, the Computer paddle doesn't have any properties for the speed **MoveX.** This is because the Computer paddle doesn't change its horizontal position; it will only move up or down, never left or right.

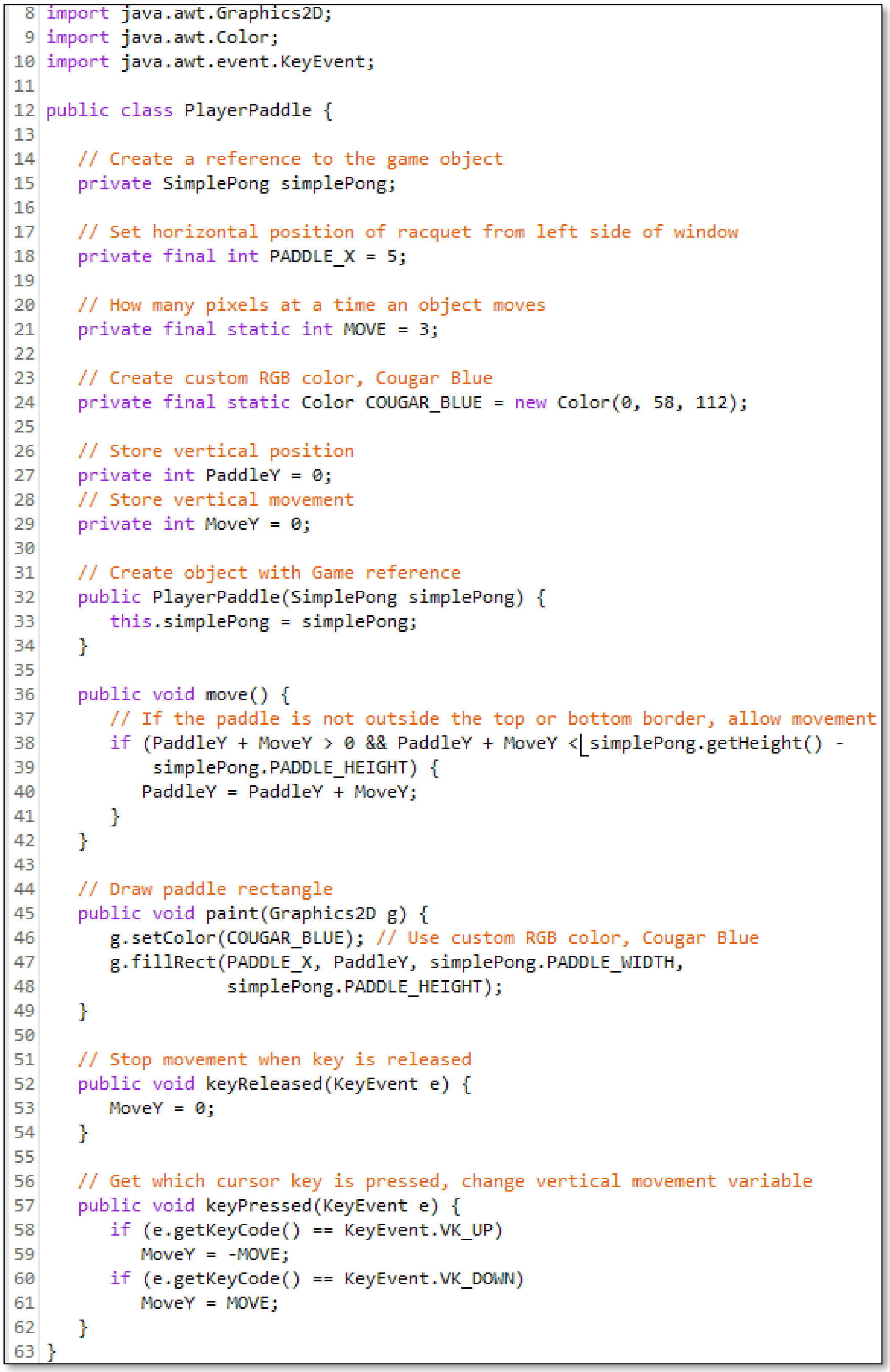
The **move()** method increases **MoveY** the position **PaddleY** and keeps the sprite within the borders of the Frame.



```java
import java.awt.Graphics2D;
import java.awt.Color;
import java.awt.event.KeyEvent;

public class ComputerPaddle {

   // Create a reference to the game object
   private SimplePong simplePong;

   // Set horizontal position of racquet from right side of window
   final int PADDLE_X = simplePong.GAME_WIDTH - 30;

   // Create custom RGB color, Cougar Gold
   private final Color COUGAR_GOLD = new Color(249, 190, 0);

   // Store vertical position
   private int PaddleY = 0;
   // Set Computer paddle speed
   private int MoveY = 3;

   // Create object with Game reference
   public ComputerPaddle(SimplePong simplePong) {
      this.simplePong = simplePong;
   }

   // The Computer paddle continuously moves up and down
   public void move() {
      // If the paddle is not outside the top or bottom border, allow movement
      if (PaddleY + MoveY > 0 && PaddleY + MoveY <
          simplePong.getHeight() - simplePong.PADDLE_HEIGHT) {
         PaddleY = PaddleY + MoveY;
      } else {
         MoveY = -MoveY;
      }
   }

   // Draw paddle rectangle
   public void paint(Graphics2D g) {
      g.setColor(COUGAR_GOLD); // Use custom RGB color, Cougar Gold
      g.fillRect(PADDLE_X, PaddleY, simplePong.PADDLE_WIDTH,
                 simplePong.PADDLE_HEIGHT);
   }
}
```

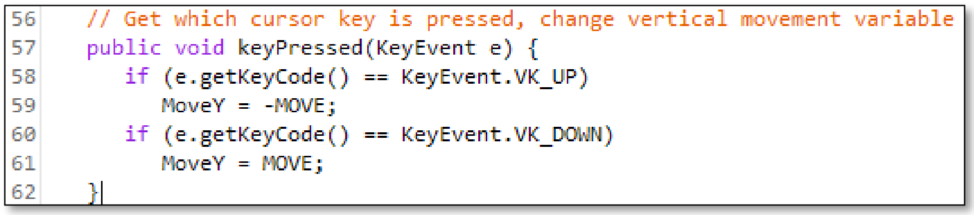
In the beginning the value of **PaddleY** is zero, which indicates that the paddle will be in the top border of the canvas. "MoveY" is also initialized to zero, which makes the paddle look static in the beginning, because **PaddleY = PaddleY + MoveY** won't change **PaddleY** while **MoveY** is zero.

PlayerPaddle Class



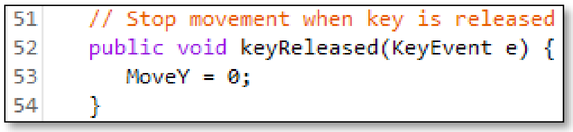
```java
import java.awt.Graphics2D;
import java.awt.Color;
import java.awt.event.KeyEvent;

public class PlayerPaddle {

   // Create a reference to the game object
   private SimplePong simplePong;

   // Set horizontal position of racquet from left side of window
   private final int PADDLE_X = 5;

   // How many pixels at a time an object moves
   private final static int MOVE = 3;

   // Create custom RGB color, Cougar Blue
   private final static Color COUGAR_BLUE = new Color(0, 58, 112);

   // Store vertical position
   private int PaddleY = 0;
   // Store vertical movement
   private int MoveY = 0;

   // Create object with Game reference
   public PlayerPaddle(SimplePong simplePong) {
      this.simplePong = simplePong;
   }

   public void move() {
      // If the paddle is not outside the top or bottom border, allow movement
      if (PaddleY + MoveY > 0 && PaddleY + MoveY < simplePong.getHeight() -
          simplePong.PADDLE_HEIGHT) {
         PaddleY = PaddleY + MoveY;
      }
   }

   // Draw paddle rectangle
   public void paint(Graphics2D g) {
      g.setColor(COUGAR_BLUE); // Use custom RGB color, Cougar Blue
      g.fillRect(PADDLE_X, PaddleY, simplePong.PADDLE_WIDTH,
                 simplePong.PADDLE_HEIGHT);
   }

   // Stop movement when key is released
   public void keyReleased(KeyEvent e) {
      MoveY = 0;
   }

   // Get which cursor key is pressed, change vertical movement variable
   public void keyPressed(KeyEvent e) {
      if (e.getKeyCode() == KeyEvent.VK_UP)
         MoveY = -MOVE;
      if (e.getKeyCode() == KeyEvent.VK_DOWN)
         MoveY = MOVE;
   }
}
```

When the up cursor key is pressed (KeyEvent.VK_UP), the **keyPressed** method of **PlayerPaddle** will be called and this will set **MoveY** to **-MOVE**. This will move the paddle up. In the same way if we press the key KeyEvent.VK_DOWN it will move to the bottom.



```java
   // Get which cursor key is pressed, change vertical movement variable
   public void keyPressed(KeyEvent e) {
      if (e.getKeyCode() == KeyEvent.VK_UP)
         MoveY = -MOVE;
      if (e.getKeyCode() == KeyEvent.VK_DOWN)
         MoveY = MOVE;
   }
```

When a key is released, the method **keyReleased** is called. **MoveY** changes its value to zero, which makes the racquet stop.



```java
   // Stop movement when key is released
   public void keyReleased(KeyEvent e) {
      MoveY = 0;
   }
```

If we run the example, we can see how the ball moves bouncing against the borders, the computer paddle moving up and down. The player paddle moves when we press the direction keys. When the ball collides with the paddle, it goes through as if it didn't exist. In the next tutorial we will see how to make the ball bounce on the paddle.

Assignment Submission

Attach the .java files to the assignment in Blackboard.