# THE *PADDLE* CLASS:
## ADDING A USER-CONTROLLED PADDLE

Create a **Paddle** class according the specifications outlined in the following UML diagram:

| **Paddle** |
| --- |
| - x, y, padWidth, padHeight: int<br>- c: Color |
| Paddle(x: int, y: int, padWidth: int, padHeight: int, c: Color)<br><br>+ draw(g2: Graphics2D): void<br>+ getHeight(): int<br>+ getWidth(): int<br>+ getX(): int<br>+ getY(): int<br>+ moveHorizontal(pixels: int): void<br>+ moveVertical(pixels: int): void<br>+ setHeight(height: int): void<br>+ setWidth(width: int): void<br>+ setX(xPos: int): void<br>+ setY(yPos: int): void |

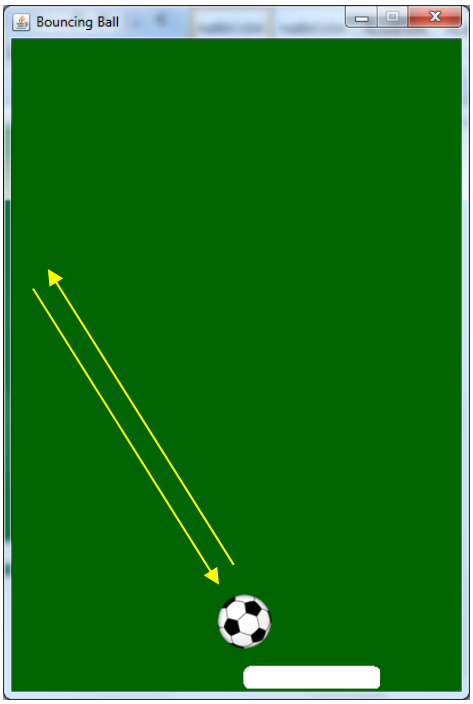
Once you have your paddle and have added it to your **BouncingBall** application, you will need to use the **KeyEvent** class in your program and write the code to increase the x-position of the paddle when the user clicks the RIGHT key and decreases the x-position of the paddle when the user clicks the LEFT key, while ensuring that the paddle does not go off the screen. Your code should look something like this:

```
public class BouncingBall extends JPanel implements
   ActionListener, KeyListener {

private Paddle p1;
...

public BouncingBall() {

   p1 = new Paddle(0, 550, 120, 20, Color.WHITE);
   ...

   setLayout(null);
   setBackground(new Color(0, 100, 0));
   addKeyListener(this);
   setFocusable(true);
   ...
}
```

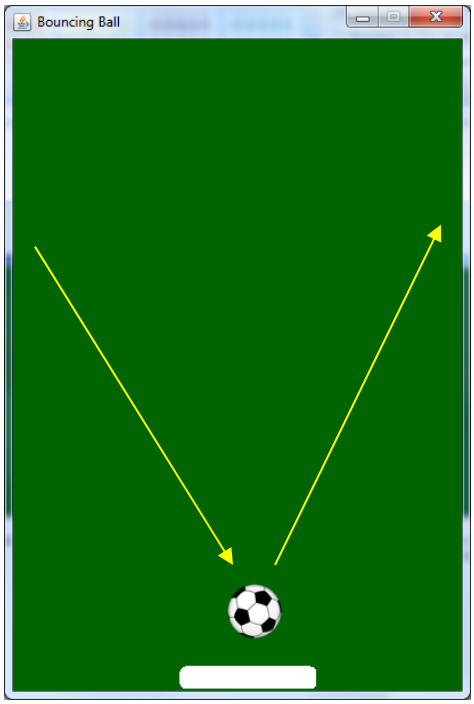
```
public void keyPressed(KeyEvent e) {

   if (e.getKeyCode() == KeyEvent.VK_RIGHT)
   {
      p1.moveHorizontal(10);
   }

   else if (e.getKeyCode() == KeyEvent.VK_LEFT)
   {
      p1.moveHorizontal(-10);
   }
}

public void keyTyped(KeyEvent e) {}
public void keyReleased(KeyEvent e) {}
```

Your program output should now look something like this where the paddle should be able to respond to the right key (by moving to the right) and the left key (by moving to the left):

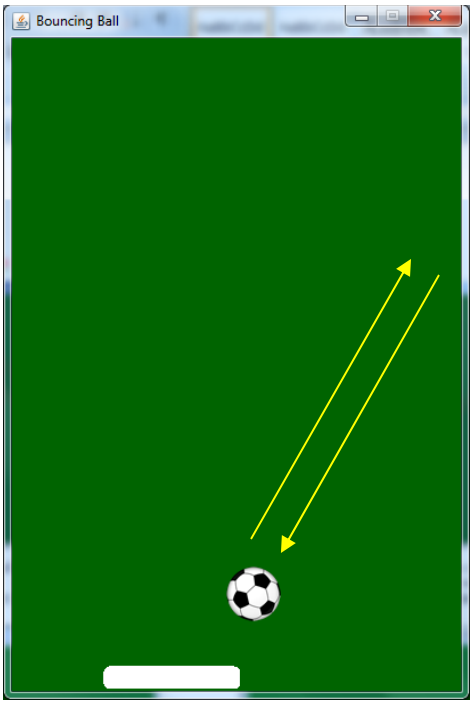
Once you have your keys working, see if you can get the paddle to detect a collision between it and the ball. If the ball hits the paddle, it should naturally ricochet off the paddle by traveling back up at the correct angle. In other words, if the ball hits the left side of the paddle while traveling south-east, it should travel north-west back up the screen. If the ball hits the middle or right side of the paddle while traveling south-east, it should continue traveling north-east up the screen. If the ball hits the right side of the paddle while traveling south-west, it should travel north-east back up the screen. If the ball hits the middle or right side of the paddle while traveling south-west, it should continue traveling north-west.



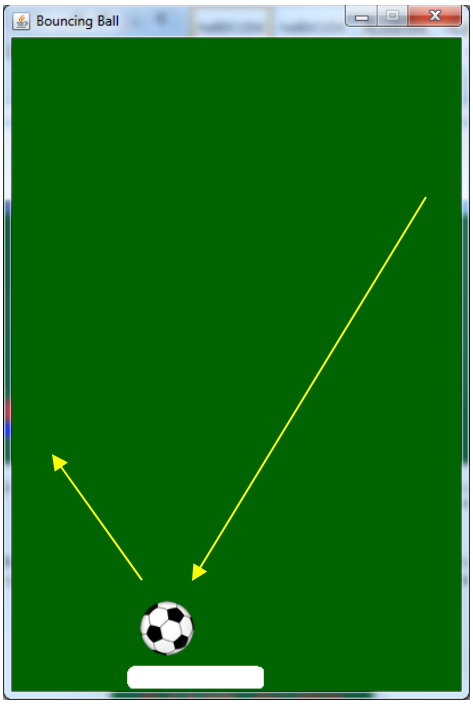

Ball hits the left side of the paddle while traveling south-east




Ball hits the middle or right side of the paddle while traveling south-east




Ball hits the right side of the paddle while traveling south-west




Ball hits the middle or right side of the paddle while traveling south-west