Part 1: Either start with your own solution to homework 6 or ask your professor for a working version:

The BouncingBall class extends the JPanel class and implements the ActionListener interface. It uses a Timer object that initially does nothing. ActionEvents are fired whenever the timer changes value. The actionPerformed method has a single line that calls repaint(). The effect is to redraw the scene every time the timer generates an ActionEvent. The paintComponent method is called when the program starts running and on every call to repaint() in actionPerformed.

For homework 6, you should have the ball start moving across the scene, toward the top left, bouncing off all 4 walls of the scene. You should have a rectangle wider than it is tall about 50 pixels above the bottom of the scene. Your first task in this lab is to make the paddle move when the mouse moves in the scene.

Part 2: Implementing a “paddle” that moves with the mouse

As mentioned in lecture today, there are several ways to listen for MouseEvents. There is an interface that specifically listens for mouse movements: MouseMotionListener. Change the file for homework 6 such that it implements both ActionListener and MouseMotionListener. The MouseMotionListener interface requires the creation of two methods: mouseMoved and mouseDragged, both of which listen for MouseEvents. For this lab, have your paddle move when the mouse moves on the scene. The x coordinate of the paddle is all that should change. The y coordinate of the paddle should be fixed so the paddle can only move horizontally, not vertically. When you move the mouse, you should have the paddle move to match the x coordinate of the mouse pointer on the scene.

Note that you can make the method bodies for unused interface methods blank by just writing, for example

```
public void mouseDragged(MouseEvent evt){}
```

You should provide an implementation for the mouseMoved method, in which you can set the new values of the paddle x coordinate to the x position of the mouse. The x position of the mouse can be accessed through the MouseEvent that the method takes as an input parameter. This MouseEvent is generated by the system.

Part 3: Detecting when the ball is touching the paddle

The only side of the paddle that you need to consider coming into contact with the ball is the top, when the ball is moving in the positive y direction. You need to detect when the ball is touching the paddle top, as shown in the figure below:
Part 4: Dropping ball off bottom of scene and writing text on drawing surface

After you successfully get the ball bouncing off the paddle, you should change the implementation so that if the ball misses the paddle, it keeps going off the bottom of the scene. When this occurs, see if you can figure out how to draw the String “Beware of falling ball” in red letters on the center of the scene.

Ball is close enough in terms of x coordinates when

\[
\text{Math.abs (paddleCenterX – ballCenterX)} \leq (\text{PADDLE\_WIDTH}/2 + \text{DIAMETER}/2)
\]

but this is not enough, as shown below.

It must also be the case that the y coordinates compare as follows

\[
\text{Math.abs(paddleCenterY – ballCenterY)} \leq (\text{PADDLE\_HEIGHT}/2 + \text{DIAMETER}/2)
\]

AND the ball is moving downward,

\(y\text{Change} > 0\)