The goal of this assignment is to get your bouncing SmileyFace (assignment 1E) working under a double buffered solution.

# Change the GUI model

The previous assignments used a JPanel object as the game surface. It was embedded in a JFrame with panel of buttons at the bottom. Switch to use a full screen model instead. As a basis, use the example we did in class. Take the timing and position logic from your bouncing SmileyFace solution and put it in this new model.

# Revise the SmileyFace class

The SmileyFace class encapsulates a GUI Component object (examples: JPanel, JFrame, etc.) The SmileyFace draws itself on Graphics objects obtained from the encapsulated Component. This will not work under the new model. The new model has alternating Graphics objects that it uses to achieve double buffering and page flipping. Therefore the SmileyFace class will have to change as follows:

1. Add a new draw() method that receives a Graphics object as a parameter. This will allow your client to pass in an alternating Graphics object. Copy the logic for the existing draw() method, and remove the code that retrieves a Graphics object from the encapsulated Component. Draw on the Graphics object that’s received as a parameter instead.
2. Add another moveTo() method to complement the first moveTo() method you wrote. The one you wrote before receives x and y. The new one will need to receive x, y, and a Graphics object. The method can be thought of as “moving the SmileyFace to a new location on another Graphics object.”
   1. There’s no need to call clear() in this one because we will presume that we are receiving a *fresh* Graphics object from the client each time. The SmileyFace ‘s image will not be on this fresh Graphics object.
   2. Change the instance variables for x and y.
   3. Call the new draw() method, the one that receives a Graphics object.