Breakout Phase 2

In which we restructure our code to make an application class and add an invisible obstacle that causes the ball to bounce around.

1. Start a new file.

   - Load your `breakout1.py` file into IDLE
   - Do a file/save as to save it again as `breakout2.py`

2. Recreate the bouncing ball application as an object. Instead of having a `main()` function, we are going to structure the entire application as a `BreakoutGame` class. Initially, this class will have just two methods: `__init__()` and `run()`.

   - Add this class to the file (you need to fill-in the methods):

     ```
     class BreakoutGame:

         def __init__(self):
             # Create the GraphWin and Ball objects and save them as
             # instance variables

         def run(self):
             # The main loop of the breakout1 program goes here. It will
             # use the ball and window instance variable.
     ```

   - At the bottom of the file add code to create a `BreakoutGame` object and call its run method:

     ```
     game = BreakoutGame()
     game.run()
     ```

   - Test and debug this version of the bouncing ball program.

3. Add a `BoundingBox` class to your `breakout2.py`. In the final program, we will have numerous objects that interact with the ball (walls, bricks, and a paddle). The task of determining when an object is hit and causing the ball to bounce appropriately will be handled by its `BoundingBox`.

   - Copy the `BoundingBox` class from our objects lab into your `breakout2.py` file.
   - Add a `bounce()` method to the BoundingBox class. This method takes one non-self parameter, a `Ball`. It checks to see whether the ball is striking the bounding box. If it is, it reflects the ball appropriately. The method also returns a Boolean value that is `True` if the ball struck the `BoundingBox` and `False` otherwise.

Hint: For a quick and easy test of whether the ball struck a vertical side of the box and should be reflected horizontally (in the x direction), check to see if the `hTip` point is inside the bounding box. Bouncing off of the horizontal sides will be analogous (but using `vTip`).

- To test your `BoundingBox`, add a single new instance variable, obstacle, to the `BreakoutGame` class. In the __init__() method set obstacle to be a `BoundingBox` from -40,-40 to 40, 40. Then augment the animation loop in the `run()` method so that ball bounces off this invisible box. To test the return value from `bounce()`, have the program print `"BOUNCE"` each time the ball bounces off the bounding box.
- Test and debug this version of the bouncing ball program. Save it as `breakout2.py` for handing in.