Summary

Make a Java program that prints 3 lines and then uses one Turtle in one World to draw two digits or letters near the center. Verify you can write, save, test and improve programs outside of lab, and upload course work to Blackboard for credit, so this is not a problem.

Deadline

This assignment is due uploaded to Blackboard by Wednesday, September 5, at 11:59PM. Late submissions are accepted by Thursday or Friday 11:59PM, with 20 of the 100 points deducted for each nightly deadline missed.

Prerequisites

You must get an iClicker to do this assignment completely. Study example programs and notes from lectures via http://www.cs.albany.edu/~sdc/CSI201, textbook pages 16-18, 20, 24, 43-50. Unlike the examples on pages 43-50, you must make a Java main method (like on page 20) that contains the program for printing and drawing, NOT hand in manual commands people might type into the interactions area.

Description

1. Code in the main method three calls to method System.out.println(...) print on 3 separate lines your full name, your UAlbany NetId and the "iClicker ID" from the back of your iClicker

   The code that prints the ID of one of my iClickers is

   System.out.println( "348F05BE" );

2. Under the lines that commanded the printing, code additional lines to make a World, and then make a Turtle in that World. Here's how:

   World w = new World( );
   Turtle tu = new Turtle( w );

3. Finish the program to make the Turtle draw, near the middle of the window, the last two digits or letters of your iClicker ID UNLESS THEY ARE B or E. If you have a B or an E or both, replace them with other digits or letters from your iClicker ID.

   Here is how to figure out how to do this:

   1. Study my sample code and it's results: Trace using a pencil what the Turtle does (it mostly moves in a straight line or turns in place a given number of degrees) in order to draw the B and the E.

   2. Write original code to draw your two digits or letters. Figure out, code, test, and IMPROVE your artistic results JUST A LITTLE AT A TIME, running the program and seeing the results again and again. It is OK to use trial, error and correction, which is what I did.
Example: (Remember, you must use your own name, netID, iClicker ID, and avoid my letters B, E).

```java
public class PrintAndTurtleDemo{
    public static void main(String[] a)
    {
        System.out.println("Prof. S. Chaiken");
        System.out.println("sdc");
        System.out.println("348F05BE");
        World w = new World();
        Turtle tu = new Turtle( w );
        tu.forward(-30);
        tu.forward(60);
        tu.turn(120);
        tu.forward(30);
        tu.turn(120);
        tu.forward(40);
        tu.turn(120);
        tu.forward(40);
        tu.turn(-150);
        tu.penUp();
        tu.forward(50);
        tu.penDown();
        tu.turn(-90);
        tu.forward(60);
        tu.turn(90);
        tu.forward(30);
        tu.turn(180);
        tu.forward(30);
        tu.turn(-90);
        tu.forward(30);
        tu.turn(-90);
        tu.forward(15);
        tu.turn(180);
        tu.forward(15);
        tu.turn(-90);
        tu.forward(30);
        tu.turn(-90);
        tu.forward(30);
        tu.turn(-90);
        tu.forward(30);
    }
}
```

1. Your figures (like my B) don't have to be perfect, just recognizable.

2. Your must **only** use **Turtle** methods `turn()`, `forward()`, `penUp()` and `penDown()`.

3. This way of programming graphics is tedious and usually requires debugging. ALL future projects require writing methods and calling them. We will teach much better ways. So **START EARLY!!!** Your benefit from this assignment is to get practice with choosing steps (and numeric parameter values) in a particular order in a situation when the choices of previous steps affect the results of the next steps. That kind of problem solving is the first to master!

4. The **Turtle** starts in the middle of the **World** window, pointing upward. `turn()` makes it turn clockwise the given number of degrees, so negative numbers make it turn counterclockwise.