Lab16: MathClass

To learn how the Math class works, let's create our own! Math class already exists, containing functions such as

```
Math.abs(-2)        => absolute value of -2
Math.min(1,2)       => the smaller value
Math.sqrt(4)        => square root of 4
Math.pow(2,3)       => 2 to the power 3
Math.random()       => random double between [0,1)
```

and constants (the naming convention for constants is all uppercase)

```
Math.E              => e = 2.72...
Math.PI             => pi = 3.14...
```

Paste your class code here:

Part 1: Create a MathClass class.

We are going to be creating our own version of the Math class called MathClass.

Create comments in this order:

```
// Constants (Final Static Variables)
// Static/Class Variables
// Nonstatic/Instance Variables
// Constructors
// Getters/Setters
// Static Methods
// Nonstatic Methods
```

Create a private static int called count initialized to 0. This is not an object count. We'll use this later.

**Put your code under the appropriate comment!**

Part 2: Final variables means that a variable can only be assigned once. It DOES NOT mean static or nonstatic. Both static and nonstatic variables can change, but final variables are constant. They store values like variables do, but do not change value.

Create final static variables for E and PI. Here is an example:

```
public static final double E = 2.72;
```

Part 3: We will be investigating static/nonstatic, so we will need constructors.

Create an empty constructor.

Lab16: MathClass

Part 4: Test your code! Create a Main class. Run this code and make sure it makes sense:

```
System.out.println("Math.E = " + Math.E);
System.out.println("MathClass.E = " + MathClass.E);
System.out.println("Math.PI = " + Math.PI);
System.out.println("MathClass.PI = " + MathClass.PI);
MathClass m = new MathClass();
// You can access static variables through objects and class.
// But you cannot access nonstatic variables through the class.
System.out.println("m.E = " + m.E);
System.out.println("m.PI = " + m.PI);
```

Paste your output here:

| |
|---|
| |

Part 5: Create the following 2 functions. The functions have the same name, but different argument types. This is called function overloading.

```
// Return the absolute value of value
// Do not use Math.abs()
public static double abs(double value)

// This works because the argument is an int, not a double.
// This function is not in the real Math class,
// and is just an example to illustrate function overloading.
public static String abs(int value) // Return a String such as "|-2|"

// This does not work, even though the argument name and return type are
different, because the computer can't tell the difference between this one and the
first.
// public static String abs(double val)
```

| |
|---|
| |

Part 6: In order to investigate static/nonstatic, create nonstatic versions of these 2 absolute value functions.

```
public double absNS(double value)
public String absNS(int value)
```

| |
|---|
| |

Part 7: Test your code! Adding below the test code from Part 4, run this code in Main and make sure it makes sense:

Lab16: MathClass

```
System.out.println("Math.abs(4.0) = " + Math.abs(4.0));
System.out.println("MathClass.abs(4.0) = " + MathClass.abs(4.0));
System.out.println("Math.abs(4) = " + Math.abs(4));
System.out.println("MathClass.abs(4) = " + MathClass.abs(4));

MathClass m2 = new MathClass();
System.out.println("m2.absNS(4.0) = " + m2.absNS(4.0));
System.out.println("m2.abs(4.0) = " + m2.abs(4.0));
System.out.println("m2.absNS(4) = " + m2.absNS(4));
System.out.println("m2.abs(4) = " + m2.abs(4));

// Calling nonstatic methods from the class is illegal.
// What are the 2 illegal method calls to absNS()?
```

Part 8: Create a nonstatic sqrt function for ints.

You may assume the argument is zero or positive by first taking the absolute value of the argument. You should use your own absolute value function. Because you are using MathClass.abs() inside of the MathClass class, you do not need to use "MathClass dot operator". This is a nontrivial task that may require some thought. This function is for ints, use doubles as little as possible.

Because our function takes and returns ints, it should return the lower value.

Ex, sqrtNS(5) => 2, sqrtNS(8) => 2, sqrtNS(9) => 3.

Algorithm- One way to do this is to create a loop that starts at 0 (where it ends is on you), and compares 0*0 to the argument. Then, compare 1*1 to the argument, and so on. If the square exceeds the argument, you've gone too far.

```
public int sqrtNS(int value)
```

Part 9: Test your code! Create your own test code in Main and include commented code for the illegal ways to call sqrtNS().

Part 10: We want to know how many times our MathClass methods have been run. In each function, increment our static int count. What code should go at the top of each function?

Part 11: Create a getter for count.

Lab16: MathClass

Part 12: Test your code! Run this code in Main and make sure it makes sense:

```
MathClass m3 = new MathClass();
for(int i=0; i<10; i++)
    System.out.println("m3.sqrtNS(" + i + ") = " + m3.sqrtNS(i));
// Guess what this will print out before running:
System.out.println("MathClass.getCount() = " + MathClass.getCount());
System.out.println("m3.getCount() = " + m3.getCount());
```

Part 13: Paste your entire MathClass code:

Paste your entire Main code:

Paste your entire output: