Lab: JavaTown Objects

For this lab, you will need to have both Java and JavaTown installed on your computer. JavaTown is a visual and interactive programming environment. It provides a visual representation of the inner workings of object-oriented programs, in which on-screen characters are literally depicted as carrying messages to each other and remembering values. JavaTown was created to introduce students to a *subset* of the Java language and to illustrate some of the more elusive concepts in object-oriented programming. Note down the answers to the questions in your notebook. Make sure to read the ***BeforeYouStartOnTheJavaTownObjectsLab.pdf*** on Schoology before you get started :)

Exercise: Calculator

You're now going to create the classes we used in our role-playing exercise. You'll write the code for these classes **in a separate text file** (using a text editor like Microsoft Notepad or TextEdit, or TextWrangler for the Mac, etc.), and periodically reload these definitions using JavaTown's "load" button.

Now, go ahead and create the `Calculator` class from our role-play. You'll define all of these in the same file (something we would never do in Java). Be sure to test each of your methods in JavaTown.

- When asked to `add`, it will be given two numbers. Reply with their sum. Note that it can only add two numbers—not just one (or none) and not three or more.
- When asked to `subtract`, it will be given two numbers. Reply with their difference. Note that it can only subtract two numbers—not just one (or none) and not three or more.
- When asked to `multiply`, it will be given two numbers. Reply with their product. Note that it can only multiply two numbers—not just one (or none) and not three or more.

Exercise: Some Additions

Now add the following more advanced methods to the `Calculator` class. Be sure to test that each works correctly.

- `square`, which should return the square of the number passed to it.
- `isOdd`, which should return *`true`* if the number passed to it is odd, and *`false`* otherwise. If you're really clever, you can implement this method *without* using the `if` construct.

- `max`, which should take in two numbers and return whichever value is greater. What should this method do if both numbers are the same?
- `abs`, which should return the absolute value of the number passed to it. What is the absolute value of 0?

Exercise: Box

Define a class called `Box`, which should behave as follows:

```
box = new Box(4);
x = box.getValue(); //x is 4
y = box.getValue(); //y is 4
box.setValue(2);
x = box.getValue(); //x is 2
```

NOTE: This is a sample usage. The above code should not appear in the box definition!

Exercise: Counter

Define a class called `Counter`, which should behave as follows:

```
counter = new Counter();
counter.tick();
counter.tick();
counter.tick();
x = counter.getCount();  //x is 3
counter.tick();
counter.tick();
x = counter.getCount();  //x is 5
counter.reset();
counter.tick();
counter.tick();
x = counter.getCount();  //x is 2
```

Exercise: Flipper

Define a class called `Flipper`, which should behave as follows:

```
flip = new Flipper();
x = flip.next(); //x is true
x = flip.next(); //x is false
x = flip.next(); //x is true
x = flip.next(); //x is false
```

Exercise: Fibber

Define a class called `Fibber`, which should output the Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, ...) as follows:

```
fib = new Fibber();
x = fib.next();   //x is 1
x = fib.next();   //x is 1
x = fib.next();   //x is 2
x = fib.next();   //x is 3
```

Exercise: Swapping

Define the following `Container` class.

```
public class Container
{
     private stuff;

     public Container(s) { stuff = s; }

     public getStuff() { return stuff; }
}
```

Now define a class called `Swapper`, which behaves as follows.

```
c1 = new Container(1);
c2 = new Container(2);
swapper = new Swapper();
swapper.swap(c1, c2);
x = c1.getStuff();          // x is 2
x = c2.getStuff();          // x is 1
```

Hint: If you find it helpful to modify the `Container` class, you may do so. (Justify whether you can get `Swapper` to work correctly on the unmodified `Container` class.)

Additional Credit (required to earn above 95%):

Exercise: Enumerator

Define a class called `Enumerator`, that "lists" all integers from a low to a high value. The following shows a sample usage.

```
e = new Enumerator(3, 5);
e.hasNext()          returns true
e.next()             returns 3
e.next()             returns 4
e.hasNext()          returns true
e.next()             returns 5
e.hasNext()          returns false
```

What do you think should happen if the `next()` method is called when `hasNext()` is false. Justify your answer in your notebook and implement accordingly.

How to Get Checked Off

You must show me the file in which you have defined the classes `Calculator` (with additional methods), `Box`, `Counter`, `Flipper`, `Fibber`, `Container`, and `Swapper`. In addition, you must demonstrate that these classes work properly. If possible, you should simply show all this to me on your computer, during class. Otherwise, you may show me your work during office hours (late penalty applies). You **must** also upload your code to Schoology before the due date. You have up to 2 working days post the due date to get checked off in class.