Lab 4: Static Stuff

Due: 11:59PM 9/19/12

For this lab, we're going to experiment with `static` a bit. The point of these exercises is to understand this important keyword, **not** necessarily to demonstrate the best way to implement things. Write programs as needed for the following items, and answer the questions:

1. Since `static` methods can operate on `static` class variables, we may be tempted to treat the class object like a normal instance object. This is, in general, a bad idea, but that doesn't mean it's a bad exercise.

   Write a `StaticCounter` class with a `static int count()` method that starts at zero, and returns consecutive integers with each call (starting at zero). Demonstrate this in action with the following driver class:

```
public class Driver1
{
    public static void main(String[] args)
    {
        for (int i=0; i<10; i++)
        {
          System.out.println("Counter returned: " +
                                  StaticCounter.count());
        }

    }
}
```

   Question: Why isn't this a good way to implement a Counter class?

2. During lecture this week, we developed a `CountedObject` class that updated a `static` variable every time an instance was created (using code in the instance's constructor). We want to do something similar, but instead of merely counting the number of instances, we want the class to actually keep a reference to every instance.

   Call your class `SavedObject` and give it the following static variable:

```
private static ArrayList<SavedObject> instances
```

   In the constructor, save a reference to the new instance (under construction) using the following code:

```
instances.add(this);
```

   How this works: `this` is a reference to the instance currently being constructed, and we're merely adding it to the static `ArrayList`. **Make sure this is the last line in your constructor, so the instance variable mentioned below will be correct.**

Like our `CountedObject` from lecture, make sure the `SavedObject` also has an `int id` instance variable that is unique for each instance. You should initialize this value in the constructor to the current size of the `instances` list. For the first instance, the list will be empty so the `id` will be 0. For the second instance, the list will have one element in it, so the `id` will be 1...

Provide a

```
public static SavedObject getInstance(int id)
```

method that returns the instance with `id`. If you do it right, it should be the `id`th entry in `instances`.

Your `SavedObject` class should work with the following driver code:

```
public class Driver2
{
    public static void main(String[] args)
    {
        for (int i=0; i<100; i++)
        {
            SavedObject tmp = new SavedObject();
        }
        SavedObject instance = SavedObject.getInstance(5);
        System.out.println("got instance: " + instance.getId());
    }
}
```

Bonus Question: How does the fact that your class saves a reference to every instance affect the garbage collector?

3. If you try to compile the following code, it will fail. Fix it, and describe (in properly formatted English) the problem with you solved.

```
public class BadProgram
{
  public String name="Carl";
  public static void main(String[] args)
  {
    System.out.println("The name is " + name);
  }
}
```

Turn In

All code and discussiouns should exist in a folder named `YourName_1110_Lab4`. Zip up this folder and submit via blackboard by the due date. Failure to follow this naming convention will result in a 0.

Make sure all files have your name at the top.