Object Parameters Example (Version 2.1)
BE-104, Fall '05, Dr. C. S. Tritt

```
/**
 * This class demonstrates <i>primitive</i> and object parameters and
 * return values.<br> Here's more text after a html "break" tag.
 *
 * <p><b>To do:</b> Nothing. This just demonstrates embedded
 * html.</p>
 *
 * @author Dr. C. S. Tritt
 * @version 2.1
 * (last revised 5/17/05)
 */
public class ObjParms {
   /**
    * The value stored in the ObjParms object. Arbitrary units.
    */
   private int value;
   /**
    * Creates a ObjParms object containing the specified value.
    *
    * @param v The value to be stored.
    */
   public ObjParms(int v){
      value = v;
   }
   /**
    * Changes the value stored in the ObjParms object.
    *
    * @param v The new value to be stored.
    */
   public void setValue(int v) {
      value = v;
   }
   /**
    * Returns the value stored in the object.
    *
    * @return The value stored in the object.
    */
   public int getPrimitive() {
      return value;
   }
   /**
    * Returns a new object containing the same value as the existing
    * object.
    *
    * @return The new object.
    */
   public ObjParms getObject() {
      return new ObjParms(value);
   }
   /**
    * Changes the values of the passed parameters and returns the sum
    * of the new values. Changes to primitive parameters "go away"
    * while changes to the state of object parameters "stick."
    *
    * @param prim A primitive parameter.
    * @param obj An object parameter.
    *
    * @return The sum of the changed parameters.
    */
   private int doSomething(int prim, ObjParms obj) {
      prim = 11; // Will this stick?
      obj.setValue(12); // How 'bout this?
      System.out.println("At end of doSomething...");
      System.out.println("prim & obj are now: " + prim + ", " +
         obj.getPrimitive());
      return prim + obj.getPrimitive();
   }
   /**
    * Tries to change the object referenced by a passed object
    * parameter. This has no effect in calling program.
    *
    * @param obj An object parameter.
    */
   private void doSomethingElse(ObjParms obj) {
      obj = new ObjParms(13); // What will this do?
      System.out.println("At end of doSomethingElse...");
      System.out.println("obj is now: " + obj.getPrimitive());
   }
   /**
    * Used to demonstrate this class and parameter passing.
    *
    * @param args Command line arguments.
    */
   public static void main(String[] args) {

//    Primitive & object parameter & primitive return demonstration.
      int x = 1;
      ObjParms op = new ObjParms(2);
      System.out.println("Before any calls...");
      System.out.println("x & y are now: " + x + ", " +
         op.getPrimitive());
      int z = op.doSomething(x, op);
      System.out.println("After doSomething...");
      System.out.println("x, y & z are now: " + x + ", " +
         op.getPrimitive()+ ", " + z );

      //    Object return demonstration.
      ObjParms newOp = new ObjParms(3);
      ObjParms nextOp = newOp.getObject();
      System.out.println("After getObject...");
      System.out.println("newOp & nextOp are now: " +
            newOp.getPrimitive() + ", " + nextOp.getPrimitive());

      // Object parameter demonstration.
      nextOp.doSomethingElse(newOp);
      System.out.println("After doSomethingElse...");
      System.out.println("newOp & nextOp are now: " +
            newOp.getPrimitive() + ", " + nextOp.getPrimitive());
   }
}
```

Output produced:

```
Before any calls...
x & y are now: 1, 2
At end of doSomething...
prim & obj are now: 11, 12
After doSomething...
x, y & z are now: 1, 12, 23
After getObject...
newOp & nextOp are now: 3, 3
At end of doSomethingElse...
obj is now: 13
After doSomethingElse...
newOp & nextOp are now: 3, 3
```