OOP – Test – April 23, 2015 (30 minutes) Name:_____

1. Suppose you have a method **f** that throws an exception and another method **g** that doesn't. How will you write a main method that first calls **f**, and if it throws an exception, calls **g**? Use the following template to answer:

```
class myException extends Exception{
   public myException(String s){
         super(s);
         System.out.println("myException Thrown.."+this.getMessage());
   }
}

class exceptionEx{
   public static void f(int i) throws myException{
         if ( i <=0){
            throw new myException("Negative numbers not allowed");
         } else{
            System.out.println("f threw no exception");
         }
   }
   public static void g(int i) {
         System.out.println("using g - g never throws an exception");
   }
   public static void main(String[] args){
   //YOUR CODE
   /** Read in the int value of args[0], pass it to f() and if that throws exception
       pass it to g() which will not throw any exceptions.
   */


         int  i= Integer.parseInt(args[0]);
         try{
            f(i);
         }catch (myException e){
            try{
                  g(i);
            }catch (Exception e2){
                  System.out.println(e2.getMessage());
            }
         }

   }
}
```

```
linux5:~/OOP/Week13$ java exceptionEx 3
f threw no exception
linux5:~/OOP/Week13$ java exceptionEx -8
myException Thrown..Negative numbers not allowed
using g - g never throws an exception
linux5:~/OOP/Week13$
```

2. Write a method, ***public static void printInfo(Object o)*** , that uses Java's reflection to print the type of o, as well as all its super classes and the interfaces each of these classes implement. Use the following template as a guide.

```
public class ObjectProps{

    public static void printInfo(Object o){
        //YOUR CODE
        Class c=o.getClass();
        System.out.println("This object's class is "+c.getName());
        printInterface(c);
        while ((c = c.getSuperclass()) != null) {
            System.out.println("....extends " + c.getName());
            printInterface(c);
        }
    }

    public static void printInterface(Class c){
        Class[] theInterfaces = c.getInterfaces();
        for (int i = 0; i < theInterfaces.length; i++) {
            String interfaceName = theInterfaces[i].getName();
            System.out.println("........Implements Interface->"+interfaceName);
        }
    }
    public static void main(String[] args){
        A a=new A();
        B b=new B();
        C c=new C();
        Integer i= new Integer(10);
        printInfo(a);
        System.out.println("____________________
        printInfo(b);
        System.out.println("____________________
        printInfo(c);
        System.out.println("____________________
        printInfo(i);
        System.out.println("____________________
    }
}
class A implements I1{
}
class B extends A implements I2{
}
class C extends B{
}
interface I1{}
interface I2{}
```

```
linux5:~/OOP/Week13$ java ObjectProps
This object's class is A
........Implements Interface->I1
....extends java.lang.Object
____________________
This object's class is B
........Implements Interface->I2
....extends A
........Implements Interface->I1
....extends java.lang.Object
____________________
This object's class is C
....extends B
........Implements Interface->I2
....extends A
........Implements Interface->I1
....extends java.lang.Object
____________________
This object's class is java.lang.Integer
........Implements Interface->java.lang.Comparable
....extends java.lang.Number
........Implements Interface->java.io.Serializable
....extends java.lang.Object
____________________
linux5:~/OOP/Week13$
```