# Midterm Review

## 1 True/False

Indicate whether each of the following statements is true or false. For false statements explain WHY they are false.

1. Java class names must begin with an uppercase letter, while variable names must begin with a lower case letter.

   **Solution**: False that is just a convention, not a requirement

2. The following Java statement is legal:

   ```
   float X = 3.5;
   ```

   **Solution**: False the literal is a double and must be explicitly cast to a float for the assignment.

3. The scope of Java method variables is the point in the method that they are declared up to the end of the block in which they are declared.

   **Solution**: True

4. In order to effectively use some new class, `Foo`, I need to know what instance variables `Foo` has and how they are manipulated.

   **Solution**: False the notion of data abstraction enables us to use a class without knowing its implementation details

5. The declaration

   ```
   int [] A;
   ```

   creates an array of integers.

   **Solution**: False it creates an array variable but the array object does not yet exist.

6. The declaration

   ```
   String [] S = new String[5];
   ```

   creates 5 String objects.

   **Solution**: False it creates an array of five locations which are initialized to null

## 2 Short Answers

1. We discussed compiler/syntax errors and logic errors. Explain what each is and how they differ. Also give an example of each.

   **Solution**: Compiler errors are problems with the syntax or structure of a program. These prevent the program from being compiled properly. Some examples are forgetting a semicolon after a statement or not closing a brace at the end of a block. Logic errors are problems with what the code actually does. In other words, the code compiles and runs, but it doesn't do what is supposed to do. An example is a dangling else, such as:

   ```
   if (score >= 90)
       if (extra == true)
           System.out.println("A+");
   else
       System.out.println("Less than 90");
   ```

   The code suggests that the else is associated with the outer if, but it is actually associated with the inner if.

2. One of the debugging issues that we discussed is the infinite loop. Explain what this is and explain how a while loop can be an infinite loop. Be specific.

   **Solution**: An infinite loop is a loop that never terminates. A while loop becomes an infinite loop if the entry test condition is always true. This can happen if nothing is done to change the condition within the loop body. For example:

   ```
   int ctr = 1;

   while (ctr <= 10)
   {
       System.out.println("ctr = " + ctr);
   }
   ```

3. We said in lecture that object-oriented programming consists of 3 primary ideas. List these ideas and briefly define each.

   **Solution**:

   (a) Encapsulation and Data Abstraction

   (b) Inheritance

   (c) Polymorphism

   See slides for more details

# 3 Tracing Code

1. Give all output produced by the execution of the Java program below. Clearly mark your output by drawing a box around it. Show your work for partial credit.

```
public class pracTrace1
{
    public static void loopy(int n)
    {
        int i = 1;
        while (i < n)
        {
            for (int j = 1; j <= i; j++)
            {
                System.out.print(i + "," + j + " : ");
            }
            i += 2;
            System.out.println();
        }
        System.out.println();
    }

    public static void doopy(int n)
    {
        int i = n;
        do
        {
            System.out.println(i + " ");
            i += 1;
            i = i / 2;
        } while (i > 1);
        System.out.println();
   }

   public static void main(String [] args)
   {
       loopy(8);
       doopy(27);
   }
}
```

**Solution**:

```
1,1 :
3,1 : 3,2 : 3,3 :
5,1 : 5,2 : 5,3 : 5,4 : 5,5 :
7,1 : 7,2 : 7,3 : 7,4 : 7,5 : 7,6 : 7,7 :

27
14
7
4
2
```

# 4 Coding

1. Write a single complete Java program that does the following:

   (a) Prompts the user to enter a positive integer and reads it in. Any numbers 0 or less should be rejected and the user must re-enter.

   (b) Prompts the user to enter a positive integer larger than the first and reads it in. Any numbers less than or equal to the first integer should be rejected and the user must re-enter

   (c) Prints out the two numbers that were input by the user.

   A static method with the header below must be called to implement a) and b) above. You must write the body of this method so that it works as required.

```
public static int getInteger(Scanner s, int lowerBound)
```

   For example, see the run below:

```
Enter an integer > 0
-3
Enter an integer > 0
0
Enter an integer > 0
4
Enter an integer > 4
4
Enter an integer > 4
8
Your numbers are 4 and 8
```

   **Solution**: Answers will vary. One possibility is shown below.

```
import java.util.*;

public class pracCode1
{
    public static int getInteger(Scanner s, int lowerBound)
    {
        int val;

        do
        {
            System.out.println("Enter an integer > " + lowerBound);
            val = s.nextInt();
        } while (val <= lowerBound);

        return val;
    }
```

```
    public static void main(String [] args)
    {
        int num1, num2;

        Scanner inScan = new Scanner(System.in);
        num1 = getInteger(inScan, 0);
        num2 = getInteger(inScan, num1);
        System.out.println("Your numbers are " + num1 + " and " + num2);
    }
}
```

2. Consider the main program below. Complete the class TVClass so that the program will run as shown with the output as shown. Be careful!

```
public class pracCode2
{
    public static void main(String [] args)
    {
        TVClass myTV = new TVClass("Sony");
        myTV.setResolution(TVClass.HD);
        myTV.setSize(40);
        myTV.type = "LCD";     private type
        System.out.println(myTV);
    }
}
```

Output:

```
Brand: Sony
Size: 40
Type: LCD
Resolution: HDTV
```

**Solution**:

```
public class TVClass
{
    public static String STAN = "Standard"; // Don't worry about this decl.
    public static String ED = "EDTV"; // Dont worry about this decl.
    public static String HD = "HDTV"; // Don't worry about this decl.
    private String brand;
    private String resolution;
    private int size;
    public String type;

    public TVClass(String br)
    {
```

```
        brand = new String(br);
    }

    public void setResolution(String res)
    {
        resolution = res;
    }

    public void setSize(int sz)
    {
        size = sz;
    }

    public String toString()
    {
        StringBuilder b = new StringBuilder();
        b.append("Brand: " + brand + "\n");
        b.append("Size: " + size + "\n");
        b.append("Type: " + type + "\n");
        b.append("Resolution: " + resolution + "\n");
        return b.toString();
    }
}
// Note that instance variable type is public while the others are private.
// The convention is to make instance variables private. However, in the
// main program the "type" instance variable is accessed directly. In order
// to allow this, the variable must be declared to be public.
```