

Due:

Activity (in-lab): Monday, September 26, 2011 by the end of lab

Homework: Tuesday, September 27, 2011 by 11:59 PM on Web-CAT

Goals:

By the end of this activity you should be able to do the following:

- Gain a further understanding of if-else statements
- Understand the basics of loops

Description:

In this activity you will create two classes. NumberOperations will hold an integer value and perform various operations on that value.

Directions:**Part 1: NumberOperations: Method Stubs (40%)**

- Create a class called NumberOperations.
- Add method stubs for the following methods. **The first two are given; do the rest on your own.**
 - The constructor takes an int parameter called numberIn

```
public NumberOperations(int numberIn) {  
}
```

- getValue: takes no parameters; returns an int value

```
public int getValue() {  
    return 0; // placeholder return  
}
```

- oddsUnder: takes no parameters; returns a String
- powersTwoUnder: no parameters; returns a String
- isGreater: takes an int parameter called compareNumber; returns an int
- toString; no parameters; returns a String

Create method stubs with placeholder returns for each method on your own.

Compile NumberOperations and run the following in interactions. **Do not continue until your program compiles and the following code runs without error in interactions.**

```
NumberOperations numOps = new NumberOperations(5);  
String s1 = numOps.oddsUnder();  
String s2 = numOps.powersTwoUnder();  
int n1 = numOps.isGreater(2);  
String s3 = numOps.toString();
```

Part 2: NumberOperations: instance variable, Constructor, getValue, and toString (20%)

- Add an instance variable with the name *number* to your class that is of type int.
- In your constructor, add code that will set the value of *number* to *numberIn*.
- In your getValue method, delete the placeholder return and return the value of *number*.
- Replace the placeholder return in the toString method with the following code:

```
return number + "";
```

Compile NumberOperations and run the following code in the interactions pane. **Do not continue until the following code runs without error in interactions.**

```
NumberOperations numOps = new NumberOperations(5);
numOps.getValue()
5
numOps // displays the toString return
5
```

Part 3: NumberOperations: oddsUnder Method (10%)

- Create a local variable in oddsUnder called output and initialize it to an empty string literal.

```
public String oddsUnder() {
    String output = "";
}
```

- Add a local int variable i and a while loop that will iterate through each value of i until the value of *number*.

```
int i = 0;
while (i < number) {

}
```

- Inside of the above loop, add code that will concatenate the value of i if it is an odd number. Also increment the value of i during each iteration of the loop.

```
if(i % 2 != 0) {
    output += i + "\t";
}
i++;
```

- After the loop, add code to **return the value of output**.

Compile NumberOperations and run the following code in the interactions pane. **Do not continue until the following code runs without error in interactions.**

```
NumberOperations numOps = new NumberOperations(9);
numOps.oddsUnder()
1 3 5 7
```

Part 4: NumberOperations: powersTwoUnder Method (15%)

- Create a local String variable in powersTwoUnder called *output* and initialize it to an empty string literal as you did in oddsUnder.
- Create another local variable of type int called *powers* and initialize its value to 1.
- Add a while loop that will iterate through each number up until the value of *number*.

```
while (powers < number) {

}
```

- Inside of the while loop, add code that will concatenate the value of powers to output if it is a power of 2 and then calculate the next power of two (the comments below are optional).

```
|   output += powers + "\t"; // concatenate to output
|   powers = powers * 2; // get next power of 2
```

- Add code to **return the value of output**.

Compile NumberOperations and run the following code in the interactions pane. **Do not continue until the following code runs without error in interactions.**

```
NumberOperations numOps = new NumberOperations(20);
numOps.powersTwoUnder()
1    2    4    8    16
```

Part 4: NumberOperations: isGreater Method (15%)

- Delete the placeholder return from isGreater and add code that will return 1 if number is greater than compareNumber, -1 if number is less than compareNumber, or 0 if the numbers are equal.

```
if (number ____ compareNumber) {
    return 1;
}
else if (number ____ compareNumber) {
    return -1;
}
else {
    return 0;
}
```

Compile NumberOperations and run the following code in the interactions pane. **Do not continue until the following code runs without error in interactions.**

```
NumberOperations numOps = new NumberOperations(10);
numOps.isGreater(2)
1
numOps.isGreater(15)
-1
numOps.isGreater(10)
0
```

Homework Directions:

- Add appropriate Javadoc comments to your program. Modify your toString method so that it will print the value of the number and then ": Number is negative even", ": Number is negative odd.", ": Number is positive even", or ": Number is positive odd" (whichever is applicable) on a new line. Only the value should be printed if the number is equal to 0. Example output:

```
NumberOperations n1 = new NumberOperations(0);
n1
0
NumberOperations n2 = new NumberOperations(7);
n2
7: Number is positive odd.
NumberOperations n3 = new NumberOperations(8);
n3
8: Number is positive even.
NumberOperations n4 = new NumberOperations(-8);
n4
-8: Number is negative even.
NumberOperations n5 = new NumberOperations(-9);
n5
-9: Number is negative odd.
```

- Download the driver program called NumberList and add code so that it does the following: prompts the users for a number, creates a NumberOperations object for the number, and then adds the NumberOperations object to the ArrayList numberList. Once the user enters a value of 0, print out each NumberOperations object in the ArrayList. Example output:

```
----jGRASP exec: java NumberList

Enter a set of non-zero numbers (0 to end):
5
2
-5
-1
8
-4
0

5: Number is positive odd.
2: Number is positive even.
-5: Number is negative odd.
-1: Number is negative odd.
8: Number is positive even.
-4: Number is negative even.

----jGRASP: operation complete.
```