Lab 4: if Statements & Simple Debugging

Goal

- This lab will introduce you to some fundamental techniques concerning if statements, Strings and simple debugging

---

Debugging and Testing

Debugging code usually means tracking down and fixing problems with code that cause run-time errors or incorrect results (logic errors). Debugging can also mean removing syntax errors reported by the compiler.

The best strategy in debugging code, regardless of the category of problem, is to work on small portions of the code, check that code for correctness then proceed. For example, you may need to write a program that reads three integers from the user, determines which of the three is the smallest, and then displays the value of the smallest. From the description in the previous sentence, it should be obvious that there are three distinct tasks to perform:

1. Read 3 integers from the user
2. Determine which of the three is the smallest
3. Display the value of the smallest

So the first thing you do is set up your class and the variables you will need.

```
import java.util.Scanner;

public class FindSmallest
{
    public static void main(String [] args)
    {
        int num1, num2, num3, smallest;

    } // end main method
} // end class FindSmallest
```

From this point you can actually compile your Java code. Granted, the .class file created won't do anything, but you can at least confirm you have no syntax errors. Once this is done, you can proceed with the first real task: reading 3 integers from the user. Since you'll essentially be doing the same thing three times (reading in integers), it suffices to write the code to read the first integer, then compile and run. Your program won't do anything other than read an integer entered by the user, but again you can confirm the program is free of errors, and it compiles properly.

```
public class FindSmallest
{
    public static void main(String [] args)
    {
        int num1, num2, num3, smallest;
```

```
        Scanner keyboard = new Scanner(System.in);

        // read input from user
       System.out.print("Enter first integer: ");
      num1 = keyboard.nextInt();

    } // end main method
} // end class FindSmallest
```

OUTPUT

```
Enter first integer: 3
Press any key to continue . . .
```

Now that you know this much works, you can write the code to read the other two integers from the user. If you want to confirm the input was properly assigned to the variables you used, you can add a System.out.println() that echoes the values of the integers the user typed. This is a very handy debugging tool, especially when you perform an arithmetic operation and want to confirm the result before proceeding with subsequent operations. Many programmers will place such statements all the way on the left margin of their code to remind them that the statement is only there for debugging purposes.

```
import java.util.Scanner;

public class FindSmallest
{
   public static void main(String [] args)
   {
      int num1, num2, num3, smallest;
      Scanner keyboard = new Scanner(System.in);

      System.out.print("Enter first integer: ");
      num1 = keyboard.nextInt();
      System.out.print("Enter second integer: ");
      num2 = keyboard.nextInt();
      System.out.print("Enter third integer: ");
      num3 = keyboard.nextInt();

//DEBUGGING STATEMENT: REMOVE WHEN NO LONGER NEEDED
System.out.println("DEBUG: The integers entered were: " + num1
                   + " " + num2 + " " + num3);

    } // end main method
} // end class FindSmallest
```

OUTPUT

```
Enter first integer: 5
Enter second integer: 6
Enter third integer: 7
DEBUG: The integers entered were: 5 6 7
Press any key to continue . . .
```

Now that the input gathering has been confirmed correct both syntactically and logically, you can proceed with the next major task, which is to find the smallest of the three integers. First concern yourself with the first two integers and see if you can determine which of those is the smallest. Use a print statement to show the result.

```
import java.util.Scanner;

public class FindSmallest
{
   public static void main(String [] args)
   {
      int num1, num2, num3, smallest;
      Scanner keyboard = new Scanner(System.in);

      System.out.print("Enter first integer: ");
      num1 = keyboard.nextInt();
      System.out.print("Enter second integer: ");
      num2 = keyboard.nextInt();
      System.out.print("Enter third integer: ");
      num3 = keyboard.nextInt();

       //find smallest integer entered
      //assume its num1 until another value proves us wrong
      smallest = num1;
      if (num2 < smallest)
         smallest = num2;

//FOR DEBUGGING PURPOSES
System.out.println("DEBUG: The smallest of " + num1 + " and " + num2
   + " is: " + smallest);

    } // end main method
} // end class FindSmallest
```

OUTPUT

```
Enter first integer: 4
Enter second integer: 5
Enter third integer: 6
DEBUG: The smallest of 4 and 5 is: 4
Press any key to continue . . .
```

At this point, you know the program correctly handles comparison of the first two integers provided that the first is less than the second. But what other possibilities are there? Well, the second integer could be less than the first, or they both could have the same value. These different combinations are the type of things you want to do to thoroughly test your program to ensure it handles the different instances. So run the program again with the second integer less than the first.

OUTPUT

```
Enter first integer: 10
Enter second integer: 5
Enter third integer: 1
DEBUG: The smallest of 10 and 5 is: 5
Press any key to continue . . .
```

Finally, run the program with the same value for the first and second integer.

OUTPUT

```
Enter first integer: 5
Enter second integer: 5
Enter third integer: 1
DEBUG: The smallest of 5 and 5 is: 5
Press any key to continue . . .
```

At this point you may ask yourself, is the smallest value between 5 and 5, 5? Technically, it is. It's also the largest value. Situations such as these are when you may want to consult the customer (your instructor ;-) for clarification. Once you're satisfied with the results you're receiving, you can move on and write code that considers the third integer.

```
import java.util.Scanner;

public class FindSmallest
{
   public static void main(String [] args)
   {
      int num1, num2, num3, smallest;
      Scanner keyboard = new Scanner(System.in);

      System.out.print("Enter first integer: ");
      num1 = keyboard.nextInt();
     System.out.print("Enter second integer: ");
      num2 = keyboard.nextInt();
      System.out.print("Enter third integer: ");
      num3 = keyboard.nextInt();

      //find smallest integer entered
      //assume its num1 until another value proves us wrong
      smallest = num1;
      if (num2 < smallest)
         smallest = num2;
      if (num3 < smallest)
         smallest = num3;

//FOR DEBUGGING PURPOSES
System.out.println("The smallest of " + num1 + " and " + num2 +
                   " and " + num3 + " is: " + smallest);

   } // end main method
} // end class FindSmallest
```

OUTPUT

```
Enter first integer: 10
Enter second integer: 5
Enter third integer: 1
DEBUG: The smallest of 10 and 5 and 1 is: 1
Press any key to continue . . .
```

From, here you could try different combinations of values until you are satisfied the program performs as expected. Now all that remains is to print an official message to the user that shows the smallest value. We're in luck in that the last debug statement we used essentially does this for us. All we need to do is indent it and remove the DEBUG part of the message and we're good to go!

```
import java.util.Scanner;

public class FindSmallest
{
   public static void main(String [] args)
   {
      int num1, num2, num3, smallest;
      Scanner keyboard = new Scanner(System.in);

      System.out.print("Enter first integer: ");
      num1 = keyboard.nextInt();
     System.out.print("Enter second integer: ");
      num2 = keyboard.nextInt();
      System.out.print("Enter third integer: ");
      num3 = keyboard.nextInt();

      //find smallest integer entered
      //assume its num1 until another value proves us wrong
      smallest = num1;
      if (num2 < smallest)
         smallest = num2;
      if (num3 < smallest)
         smallest = num3;

      //display information
      System.out.println("The smallest of " + num1 + " and "
                         + num2 + " and " + num3 + " is: "
                         + smallest);

   } // end main method
} // end class FindSmallest
```

The previous example incorporated a number of good techniques.

1. An if statement work just like they do in other languages, you just need to understand the syntax.
2. Build your solution incrementally (set up class and variables, read user input, perform necessary calculations, display results)
3. Use print statements to help track values produced during program execution, remove these statements when they're no longer needed
4. Test smaller portions of a larger task before the task as a whole (compare two numbers before you compare three)
5. Consider the ranges of values your program needs to handle and test it with those (first less than second, second less than first, both have same value, then similar for all three)

It is also useful to try your program with simple values that you can calculate before hand on paper so that you can compare results of your program with what you expected. Certainly, with more complex programs this is not as feasible, but in most cases, this approach works well.

Another great testing and debugging strategy is to choose values that test the boundaries of what your program is supposed to handle. As a simple example, if you were required to read an integer from the user between 1 and 10 (1 and 10 inclusive) and you wanted to ensure only valid entries were accepted, you would want to test the program with values of 0, 1, 10, and 11. The values 0 and 11 lie just outside the specifications, while 1 and 10 are just within the specifications (or boundaries).

Exercises

1. Type in (copy and paste) FindSmallest.java under the package cscd210Lab4. Execute the program and capture the output. Use the numbers from the last example. Save the output in a file named findsmallestout.txt

2. Copy the code from FindSmallest.java into CSCD210Lab4a.java under the package cscd210Lab4a. Now modify CSCD210Lab4a.java so it prints out the numbers in ascending order. Using the input from the last example the output will be similar to:

   OUTPUT

   ```
   Enter first integer: 10
   Enter second integer: 5
   Enter third integer: 1
   The numbers in ascending order 1, 5, and 10
   Press any key to continue . . .
   ```

   Save the output in a file named cscd210Lab4aout.txt

3. Complete the code for CSCD210Lab4b.java in package cscd210Lab4b, that prompts the user to enter three types of fruit as Strings. The program will print the three entered fruits in alphabetical order. Save the output in a file named cscd210Lab4bout.txt

   No arrays are allowed for this assignment. You will need to read the String Java API

TO TURN IN: (via TurnIn)

- A single zip file that contains the Lab4 folder
- Lab4 will contain packages, your Java files and findsmallestout.txt, cscd210Lab4aout.txt, and cscd210Lab4bout.txt.

  Note: the naming scheme is the same. Last name first letter of first name lab4.zip (Example: steinerslab4.zip)