CMIS 141 Hands-on Lab
Week 3

Overview

The topics covered in this lab include using the Java Math class, using the Random class, using the String class, and developing and testing repetition statements including the for, while and do-while statements. Nested repetition statements are also covered.

It is assumed the JDK 8 or higher programming environment is properly installed and the associated readings for this week have been completed.

Submission requirements

Hands-on labs are designed for you to complete each week as a form of self-assessment. You do not need to submit your lab work for grading. However; you should post questions in the weekly questions area if you have any trouble or questions related to completing the lab. These labs will help prepare you for the graded assignments, therefore; it is highly recommended you successfully complete these exercises.

Objectives

The following objectives will be covered by successfully completing each exercise:

1. Use Java's Math class to perform common math functionality
2. Use Java's Random class to generate pseudorandom numbers of different data types.
3. Construct Strings and convert strings to numeric data types
4. Design, develop and test repetition statements in Java
5. Design, develop and test nested repetition statements in Java

More about Java Classes and Methods

As discussed last week, the Java API consists of thousands of existing classes that provide a significant amount of functionality. This week we explore some additional classes including the Math, Random and String class. The Math class provides numeric operations such as exponential, logarithm, square root, and trigonometric functions. The Math class also provides the ability to generate pseudorandom numbers.

All of the methods and fields in the Math class are static. This is a very important concept in object-oriented languages. A method (or field) that is declared as static means there is only one instance available. This is ideal for fields and methods that remain constant and don't need multiple copies in other instances of classes. From an implementation standpoint, you don't construct a Math instance. You just use the methods by calling the Classname.Methodname. For example, to calculate the absolute value of an int named value, you would use this code:

```
int absValue = Math.abs(value);
```

Notice the pattern of Math.method() is used for all methods in the Math class. The same pattern works for the static fields. For example, to print the value of PI, the following code would work:

```
System.out.println("The value for PI in Java is " + Math.PI);
```

The Random class provides an alternative to creating pseudorandom numbers. The Random class produced streams of data that can be accessed individually. Seed values can be set to allow reproducing the same set of random values. The Math.random() method provides similar functionality and may be easier to use in many cases.

The String class is used to generate character strings. String is a class but can be used like a literal. For example, the following code will create an instance of a String named firstName.

```
String firstName = "John";
```

Are you go through the examples and demonstrations in this lab, be sure to reference the Java API, found currently at the following URL:

Java 8 API URL: http://docs.oracle.com/javase/8/docs/api/

You should use the API to review the Math, Random and String classes along with their fields and methods. Note the parameters, the member modifiers (e.g. static) and the return data types. For example, here are the details of the Math.sin() method:

```
static double   sin(double a)
```

In this case, this the member modifier is static, the return type is a double, the method name is sin and the input parameter is a double.

To use this method, the following code would work:

```
double angle = 45;
double angleSin = Math.sin(Math.toRadians(angle));
```

In this case, we are using both the toRadians() and sin() methods of the Math class. The toRadians() method is needed since the sin() method requires the input to be in Radians.

Exercise 1 – Use Java's Math class to perform common math functionality

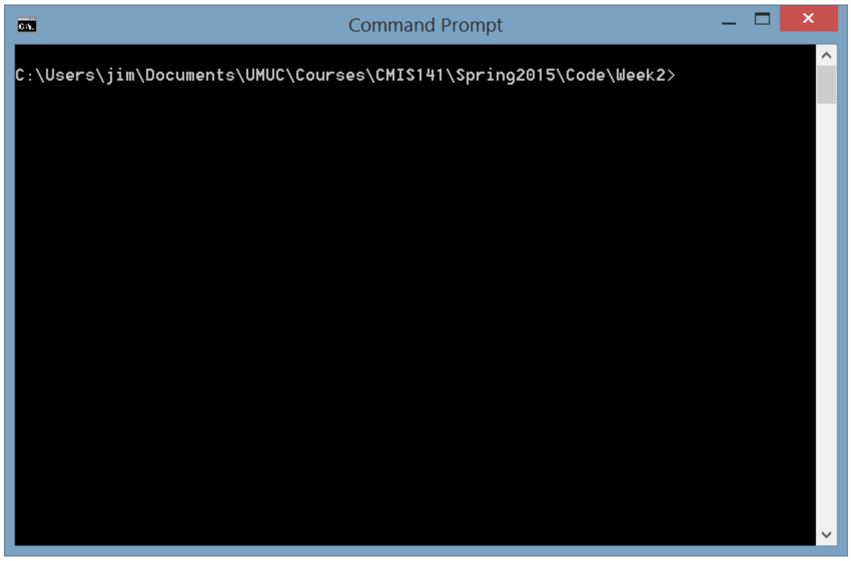
As mentioned in the More About Java Classes and Methods section above, the methods and fields of Math class are all static. When we use a static field or method we call the classname.Methodname. There is no need to construct an instance of the Math class. This makes the class and methods relatively simple to use.

In this exercise we will prompt the user to enter a couple of numbers and then take advantage of the methods in the Math class to perform several math calculations.

a. Open your favorite text editor and Type (or copy and paste) the following Java

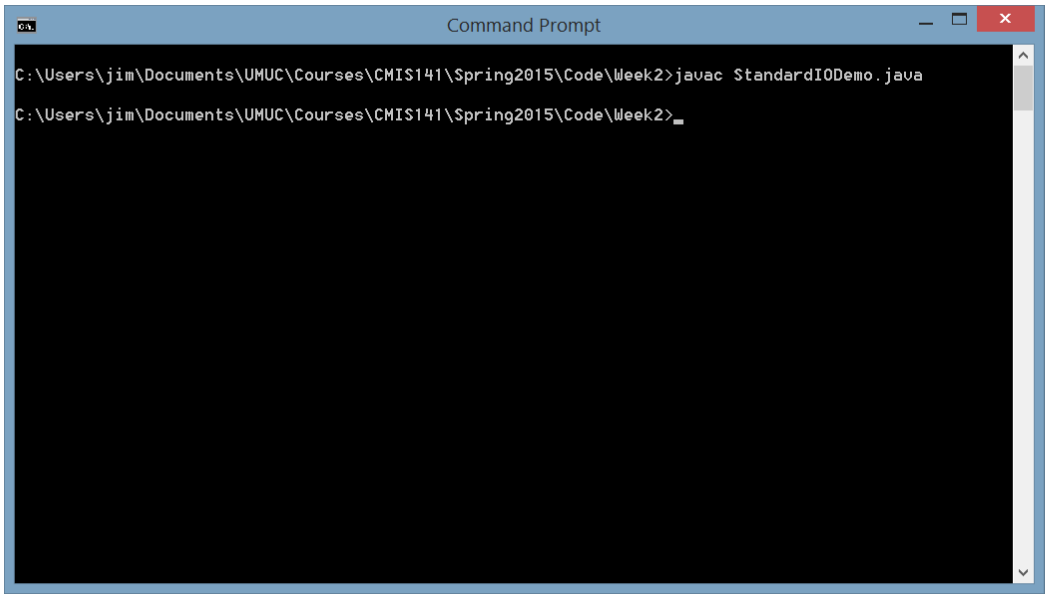
```
/*
* File: MathDemo.java
* Author: Dr. Robertson
* Date: January 1, 2015
* Purpose: This program demonstrates the use of static methods
* and fields of the Math class.
*/

// Import statements
import java.util.Scanner;

public class MathDemo {
    public static void main(String[] args)  {

      // Display a Welcome note
       System.out.println("Welcome to the Math Class Demo");

      // Variables to hold values
      double angle = 0.0;
      int negativeInt = 0;

      // Use the Scanner class to input data
        Scanner scannerIn = new Scanner(System.in);

      System.out.println("Enter the angle in degrees (0.0-360.0):");
      // the nextInt() method scans the next int value
        angle = scannerIn.nextDouble();

      // Verify the angle was entered
      System.out.println("The following angle was entered " + angle);

      System.out.println("Enter a negative int of your choice :");
      // the nextInt() method scans the next int value
        negativeInt = scannerIn.nextInt();

      // Verify the int was entered
      System.out.println("The following negative Int was entered " +
negativeInt);

    }
}
```

b. Save the file as “MathDemo.java” in a location of your choice.
c. Use the cd command to change to the folder (directory) location of your MathDemo.java file.



```
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week2>
```

d. To compile the file, type javac MathDemo.java at the command prompt.

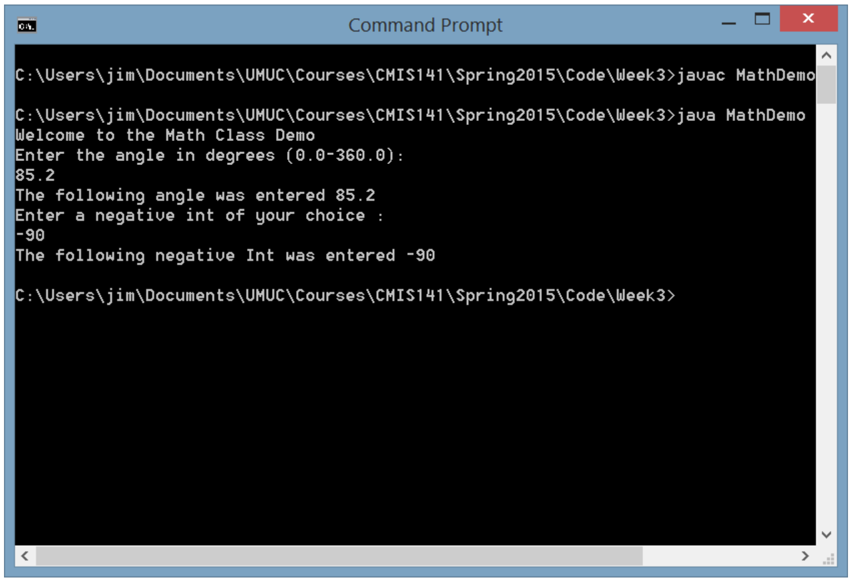


```
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week2>javac StandardIODemo.java

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week2>_
```

e. To run the file, type java MathDemo at the command prompt. When prompted, enter a double and a negative int. For example, the following values could be entered:

85.2
-90

The output will look similar to this:



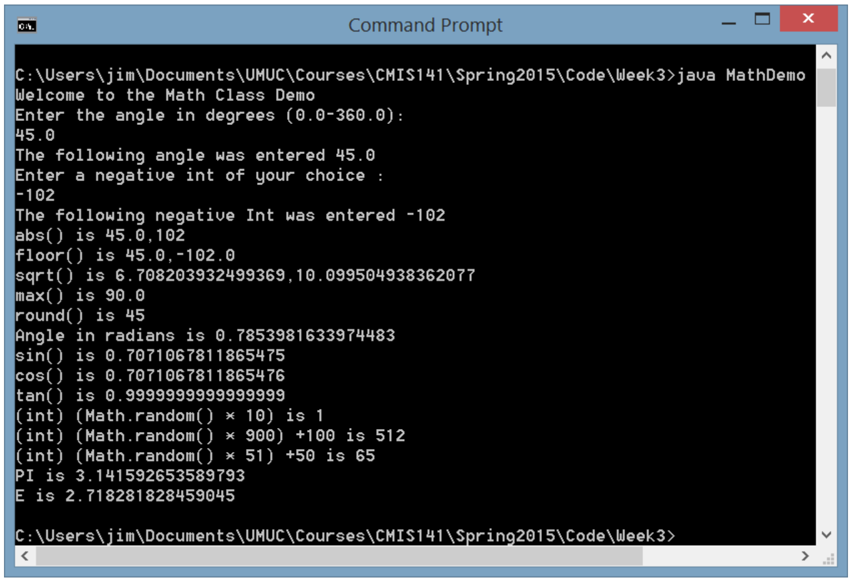
```
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>javac MathDemo

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>java MathDemo
Welcome to the Math Class Demo
Enter the angle in degrees (0.0-360.0):
85.2
The following angle was entered 85.2
Enter a negative int of your choice :
-90
The following negative Int was entered -90

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>
```

f. Now, that we know the input works as expected, we can experiment with the Math static methods. Use the Java API to reference the functionality and the specific calls for each method. We will use several methods including abs(), floor(), sqrt(), ceil(), max(), round(), sin(), cos(), tan() and random(). Since the values require the angle to be in radians, we will convert the angle to radians before using the trigonometric method calls. Update the main() method to include the following code:

```
public static void main(String[] args)  {

      // Display a Welcome note
       System.out.println("Welcome to the Math Class Demo");

      // Variables to hold values
      double angle = 0.0;
      int negativeInt = 0;

      // Use the Scanner class to input data
        Scanner scannerIn = new Scanner(System.in);

      System.out.println("Enter the angle in degrees (0.0-360.0):");
      // the nextInt() method scans the next int value
        angle = scannerIn.nextDouble();

      // Verify the angle was entered
      System.out.println("The following angle was entered " + angle);

      System.out.println("Enter a negative int of your choice :");
      // the nextInt() method scans the next int value
        negativeInt = scannerIn.nextInt();

      // Verify the int was entered
      System.out.println("The following negative Int was entered " +
negativeInt);
```

```
      // Calculate the abs(), floor(), sqrt(), max() and round() of
each entry
      System.out.println("abs() is " + Math.abs(angle) + "," +
Math.abs(negativeInt));
      System.out.println("floor() is " + Math.floor(angle) + "," +
Math.floor(negativeInt));
      System.out.println("sqrt() is " + Math.sqrt(angle) + "," +
Math.sqrt(Math.abs(negativeInt)));
      System.out.println("max() is " + Math.max(angle,90.0) );
      System.out.println("round() is " + Math.round(angle) );

      // Convert the angel to radians
      angle = Math.toRadians(angle);
      System.out.println("Angle in radians is " + angle);

      // Calculate the sin, cos and tan values for the angle entered
      System.out.println("sin() is " + Math.sin(angle));
      System.out.println("cos() is " + Math.cos(angle));
      System.out.println("tan() is " + Math.tan(angle));

      // Generate some random numbers
      // A random int between 0 and 9
      int randInt = (int) (Math.random() * 10);
      System.out.println("(int) (Math.random() * 10) is " + randInt);
      // A random int between 100 and 999
      randInt = (int) (Math.random() * 900) +100 ;
      System.out.println("(int) (Math.random() * 900) +100 is " +
randInt);
      // A random int between 100 and 999
      // Notice the pattern
      //(int) (Math.random() * MAX) +OFFSET ;
      // Yields values between OFFSET and OFFSET + (MAX-1)
      randInt = (int) (Math.random() * 51) +50 ;
      System.out.println("(int) (Math.random() * 51) +50 is " +
randInt);
      //a Random int between 199 and 1000

      // Display the PI and E values
      System.out.println("PI is " + Math.PI);
      System.out.println("E is " + Math.E);
    }
```

Compile and run the code.



```
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>java MathDemo
Welcome to the Math Class Demo
Enter the angle in degrees (0.0-360.0):
45.0
The following angle was entered 45.0
Enter a negative int of your choice :
-102
The following negative Int was entered -102
abs() is 45.0,102
floor() is 45.0,-102.0
sqrt() is 6.708203932499369,10.099504938362077
max() is 90.0
round() is 45
Angle in radians is 0.7853981633974483
sin() is 0.7071067811865475
cos() is 0.7071067811865476
tan() is 0.9999999999999999
(int) (Math.random() * 10) is 1
(int) (Math.random() * 900) +100 is 512
(int) (Math.random() * 51) +50 is 65
PI is 3.141592653589793
E is 2.718281828459045

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>
```

As you analyze and experiment with the code, note the following:

1. Methods can be daisy chained together. Notice the call to `Math.sqrt(Math.abs(negativeInt));` In this example, the absolute value of the negativeInt is calculated first and then the square root is calculated. This makes sense because negative square roots do not produce real numbers.
2. Some results may not what you expect. For example, the tan(Math.toRadians(45)) should be 1 yet the results is 0.999999999. This is a computer precision issue and occurs frequently when double precision is used.
3. The Math.random() method can be used to generate values between a specific range. The pattern for this becomes: `(int) (Math.random() * MAX) +OFFSET,` where the resulting range of values will be between OFFSET and OFFSET+MAX-1. You should experiment with this to generate different ranges of random numbers.

Now it is your turn. Try the following exercise:

Create a Java class named MyMathDemo using your favorite text editor. Be sure you name the file “MyMathDemo.java”. Add code to the file in the main() method that will provide functionality to prompt the user to enter the length of two sides of a right triangle. Then use the Math.hypot() method to calculate the length of the hypotenuse. Next, demonstrate the use of the Math.pow() method by printing the results for 10 raised to the 2nd, 3rd and 4th powers. Finally, simulate the Maryland lottery pick 5 by generating five “pseudorandom” numbers between the values of 1 and 35.

Exercise 2 – Use Java's Random class to generate pseudorandom numbers of different data types

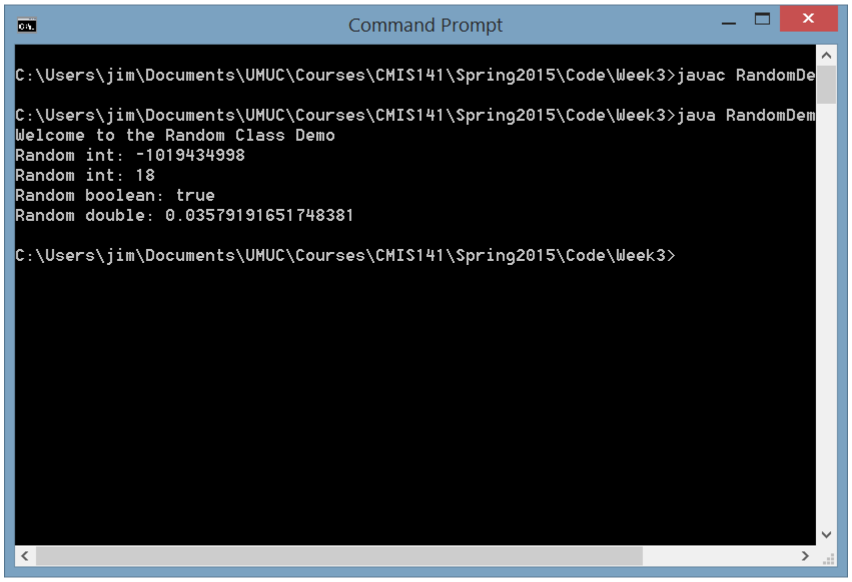
Similar to the Math.random() method in the Math class, Java's Random class has the ability to generate pseudorandom numbers. The Random class has many methods to use but let us concentrate on generating int, boolean, and double data types.

a. Open your favorite text editor and type (or cut and paste) the following code:

```
/*
* File: RandomDemo.java
* Author: Dr. Robertson
* Date: January 1, 2015
* Purpose: This program demonstrates the use of Java's
* Random class
*/

// Import statements
import java.util.Random;

public class RandomDemo {
    public static void main(String[] args)  {

      // Display a Welcome note
       System.out.println("Welcome to the Random Class Demo");

      // Variables to hold values
      int randInt = 0;
      boolean randBoolean = false;
      double randDouble = 0.0;

      // Construct a Random class instance
        Random randomGen = new Random();

      // Generate a random Int
      // range will be from MIN to MAX Java int
      randInt = randomGen.nextInt();
      System.out.println("Random int: " + randInt);

      // Generate a random Int
      // but limit it between 0 and 35
      randInt = randomGen.nextInt(36);
        System.out.println("Random int: " + randInt);

      // Generate a random boolean
      randBoolean = randomGen.nextBoolean();
        System.out.println("Random boolean: " + randBoolean);

      // Generate a random Double
      randDouble = randomGen.nextDouble();
        System.out.println("Random double: " + randDouble);

    }
}
```

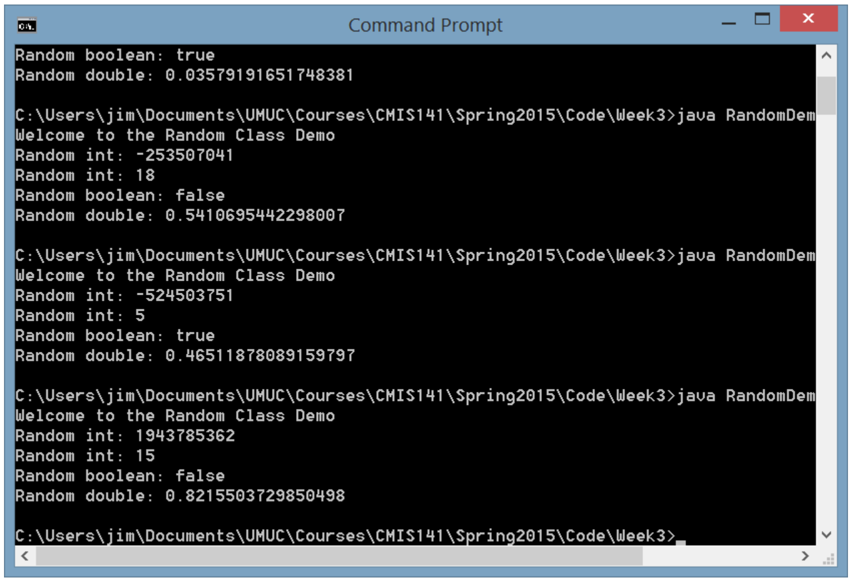
b. Save the file as “RandomDemo.java” in a location of your choice.

c. Compile and run the code. The results should look similar to this. However; keep in mind your pseudorandom values will be different.



```
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>javac RandomDe

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>java RandomDem
Welcome to the Random Class Demo
Random int: -1019434998
Random int: 18
Random boolean: true
Random double: 0.03579191651748381

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>
```

d. You should run the code several times and look at the numbers generated to make sure they seem to be in the range you expect.



```
Random boolean: true
Random double: 0.03579191651748381

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>java RandomDem
Welcome to the Random Class Demo
Random int: -253507041
Random int: 18
Random boolean: false
Random double: 0.5410695442298007

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>java RandomDem
Welcome to the Random Class Demo
Random int: -524503751
Random int: 5
Random boolean: true
Random double: 0.46511878089159797

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>java RandomDem
Welcome to the Random Class Demo
Random int: 1943785362
Random int: 15
Random boolean: false
Random double: 0.8215503729850498

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>_
```

As you analyze and experiment with the code, note the following:

1. The nextInt() method generates values in the range between -2^{31} and $2^{31} - 1$. Although applications exist, where you might want this range of values, often, you will need to use the nextInt(n) method allowing you to generate positive int values between 0 and n-1.

2. Be sure to view the Java 8 API to reveal additional methods in the Random class such as nextFloat() and nextLong(), nextGaussian(), and setSeed().

Now it is your turn. Try the following exercise:

Create a Java class named MyRandomDemo using your favorite text editor. Be sure you name the file “MyRandomDemo.java”. Add code to the file in the main() method that will provide functionality simulate the Maryland lottery Mega-Millions game by generating five “pseudorandom” numbers between the values of 1 and 75 and one “pseudorandom” number between the values of 1 and 15. Be sure to use Java’s Random class for this exercise.

Exercise 3 – Construct Strings and convert strings to numeric data types

Strings are groups of characters. They are used extensively in most programming languages. In Java, String are classes. However; we can use them similar to the way we use primitives. This exercise will demonstrate how to construct strings and use wrapper classes to convert strings into numeric data types.

Wrapper classes are convenience classes providing methods for parsing the numeric values from a string. There are some assumptions when using these conversion methods in that the string is a valid number. For example 12 is a valid number and can be parsed using the Integer.parseInt() method. However “Hello, 12” is not a valid number and would cause a runtime error if we applied the Integer.parseInt() method. A wrapper class exists for most Java primitives. Many methods are available for the wrapper classes. For this exercise we will focus on the parse methods including Byte.parseByte(), Integer.parseInt(), Float.parseFloat(), Double.parseDouble(), Long.parseLong() and Short.parseShort(). We will also convert numbers back to strings using the toString() methods of the wrapper classes.

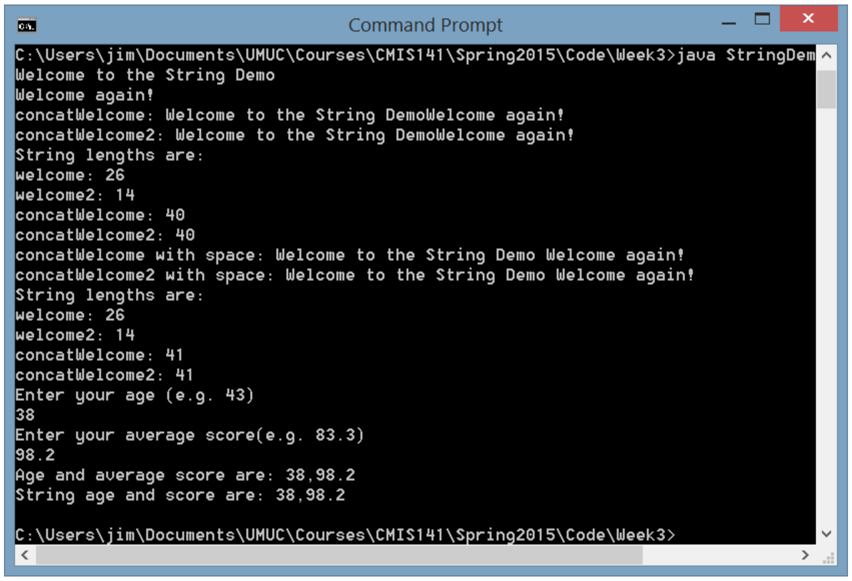
a. Open your favorite text editor and type (or cut and paste) the following code:

```
/*
* File: StringDemo.java
* Author: Dr. Robertson
* Date: January 1, 2015
* Purpose: This program demonstrates the use of Java's
* String and wrapper classes for number conversions
*/

// Import statements
import java.util.Scanner;

public class StringDemo {
    public static void main(String[] args)  {

      // Construct strings using literal assignment
```

```
String welcome = "Welcome to the String Demo";

// Display the Welcome note
  System.out.println(welcome);

// Use the typical method to construct a class:
String welcome2 = new String("Welcome again!");

// Display the second Welcome note
  System.out.println(welcome2);

// Use the concatenation method
String concatWelcome = welcome.concat(welcome2);
// Achieve the same results with the + concatenation
String concatWelcome2 = welcome + welcome2;

// Display Concatenated Strings.
System.out.println("concatWelcome: " + concatWelcome);
System.out.println("concatWelcome2: " + concatWelcome2);

// Display the length of each of the Strings
String lengthString = "String lengths are:\n" +
            "welcome: "+ welcome.length() +
            "\nwelcome2: "+ welcome2.length() +
              "\nconcatWelcome: "+ concatWelcome.length() +
              "\nconcatWelcome2: "+ concatWelcome2.length();
System.out.println(lengthString);

// To add a space between the messages
concatWelcome = welcome.concat(" ").concat(welcome2);
// Achieve the same results with the + concatenation
concatWelcome2 = welcome + " " + welcome2;

// Redisplay Concatenated Strings.
System.out.println("concatWelcome with space: " + concatWelcome);
System.out.println("concatWelcome2 with space: " + concatWelcome2);

// ReDisplay the length of each of the Strings
// One additional character is added for the space
lengthString = "String lengths are:\n" +
            "welcome: "+ welcome.length() +
            "\nwelcome2: "+ welcome2.length() +
              "\nconcatWelcome: "+ concatWelcome.length() +
              "\nconcatWelcome2: "+ concatWelcome2.length();
System.out.println(lengthString);

// Variables to hold values
int age = 0;
double avgScore = 0.0;

// Construct a Scanner class
  Scanner scannerIn = new Scanner(System.in);

// Prompt user for age and avgScore
System.out.println("Enter your age (e.g. 43)");
// Scan the next String and parse the Integer value
age = Integer.parseInt(scannerIn.next());
```

```
        System.out.println("Enter your average score(e.g. 83.3)");
        // Scan the next String and parse the Integer value
        avgScore = Double.parseDouble(scannerIn.next());

        // Display the results of the scans and parsing
        System.out.println("Age and average score are: " +
          age + "," + avgScore);

        // We can also convert numbers to Strings
        String stringAge = Integer.toString(age);
        String stringScore = Double.toString(avgScore);
        // DiIsplay results
        System.out.println("String age and score are: " +
            stringAge + "," + stringScore);

    }
}
```

b. Save the file as “StringDemo.java” in a location of your choice.
c. Compile and run the code. When prompted enter an int value for age and a double value for average exam score. The results should look similar to this:



```
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>java StringDem
Welcome to the String Demo
Welcome again!
concatWelcome: Welcome to the String DemoWelcome again!
concatWelcome2: Welcome to the String DemoWelcome again!
String lengths are:
welcome: 26
welcome2: 14
concatWelcome: 40
concatWelcome2: 40
concatWelcome with space: Welcome to the String Demo Welcome again!
concatWelcome2 with space: Welcome to the String Demo Welcome again!
String lengths are:
welcome: 26
welcome2: 14
concatWelcome: 41
concatWelcome2: 41
Enter your age (e.g. 43)
38
Enter your average score(e.g. 83.3)
98.2
Age and average score are: 38,98.2
String age and score are: 38,98.2

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>
```

As you analyze and experiment with the code, note the following:

1. You can use the String constructor (String myString = new String(“Hello”) and a literal assignment (String myString = “Hello”) to construct a String in Java
2. You can use the concat() method or the + operator to concatenate Strings in Java.
3. Daisy chaining methods are often possible in Java to use the method over and over again. For example, string1.concat(string2).concat(string3) will allow you to concatenate string1, string2 and string3.

4. Wrapper classes include parsing methods (e.g. parseInt()) and methods to convert numbers to Strings (e.g. toString()).

Now it is your turn. Try the following exercise:

Create a Java class named MyStringDemo using your favorite text editor. Be sure you name the file "MyStringDemo.java". Add code to the file in the main() method that will provide functionality to prompt the user for two strings and two integer values. Use the String concat() method and the + operator to concatenate the two Strings. Use the Integer.parseInt() to scan in two int values. Convert the int values to Strings and display the length of each of the Strings.

Exercise 4 – Design, develop and test repetition statements in Java

In this exercise, we will work through an example using repetition statements including while, do-while and for statements. We will discuss and demonstrate the use of break and continue statements within loops. In most cases, you can use either of the repetition statements for your functionality. In this exercise we will use each of the repetition statements to perform the same functionality that includes generating a user-defined number of random integers.

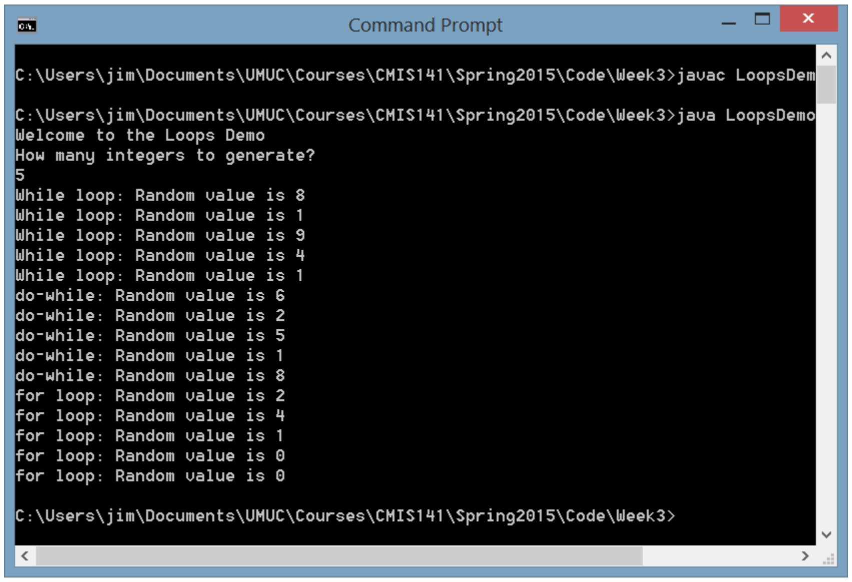
a. Open your favorite text editor and type (or cut and paste) the following code:

```
/*
* File: LoopsDemo.java
* Author: Dr. Robertson
* Date: January 1, 2015
* Purpose: This program demonstrates the use of Java's
* repetition statements (while, do-while and for)
*/

// Import statements
import java.util.Scanner;

public class LoopsDemo {
    public static void main(String[] args)  {
      int maxLoop = 0;

      System.out.println("Welcome to the Loops Demo");

      // Scanner class
      Scanner scannerIn = new Scanner (System.in);

      // Prompt use for how many random integers to generator
      System.out.println("How many integers to generate?");
      maxLoop = scannerIn.nextInt();

      // While Loop
      int cnt=0;
```

```
        int randInt = 0;
        while (cnt < maxLoop) {
            randInt = (int) (Math.random() * 10);
            System.out.println("While loop: Random value is " + randInt);
            // Increment counter
              cnt++;
          }

        // Do while loop
        // Same logic but use the do while loop
        // Reset the counter
        cnt = 0;
         do {
              randInt = (int) (Math.random() * 10);
            System.out.println("do-while: Random value is " + randInt);
             // Increment counter
               cnt++;
          } while (cnt < maxLoop);

        // For loop
        for (int i=0; i<maxLoop; i++) {
            randInt = (int) (Math.random() * 10);
             System.out.println("for loop: Random value is " + randInt);
        }

      }
}
```

b. Save the file as "LoopsDemo.java" in a location of your choice.

c. Compile and run the code. When prompted enter an int value for the number of integers to generate. The results should look similar to this:



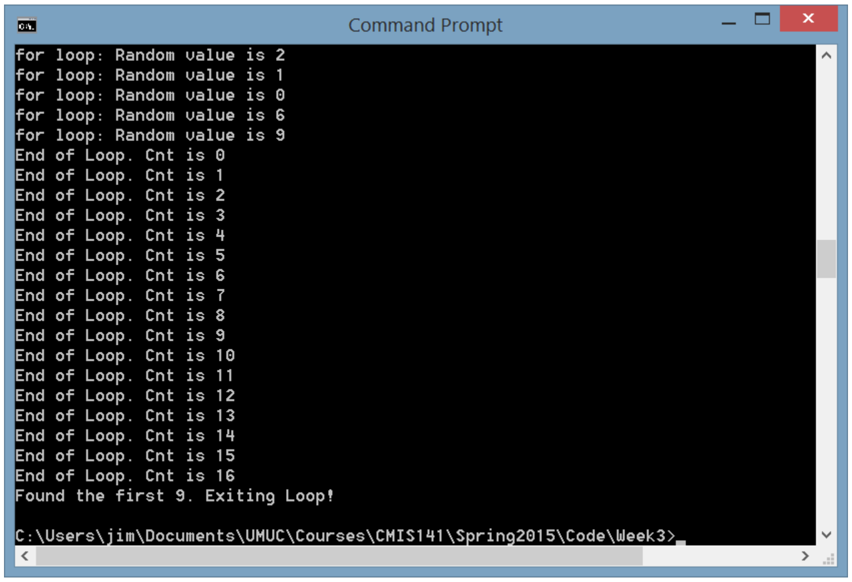
```
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>javac LoopsDem

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>java LoopsDemo
Welcome to the Loops Demo
How many integers to generate?
5
While loop: Random value is 8
While loop: Random value is 1
While loop: Random value is 9
While loop: Random value is 4
While loop: Random value is 1
do-while: Random value is 6
do-while: Random value is 2
do-while: Random value is 5
do-while: Random value is 1
do-while: Random value is 8
for loop: Random value is 2
for loop: Random value is 4
for loop: Random value is 1
for loop: Random value is 0
for loop: Random value is 0

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>
```

As you work through this code, notice the functionality of generating random integers is accomplished through the while, do-while and for structures. Notice in all cases, a counter is incremented each iteration. Without the counter incrementing, the results of evaluation the boolean expression would never change causing an infinite loop. For example, if the cnt variable was not changed, the expression of cnt < maxLoop would always remain true and the loop would continue forever.

d. The `break` statement allows you to break out from the loop under for specific conditions. For example, you may want to exit the loop as soon as the first random integer that was equal to 9 was generated. To demonstrate this functionality, add the following code the main() method. You can add this after the `for` statement.

```
// Break out of loop
// if value == 9
cnt=0;
while (cnt < maxLoop) {
    randInt = (int) (Math.random() * 10);
    // print results if the random value == 9
    if (randInt == 9) {
        System.out.println("Found the first 9. Exiting Loop!");
        break;
    }
   System.out.println("End of Loop. Cnt is " + cnt);
      cnt++;
  }
```

Compiling and executing the code will result in the following output when 30 integers are generated. Note, your output will be different depending upon the specific numbers generated. The first 9 is found approximately 16 iterations into the loop. Once the 9 was generated, the loop exits.



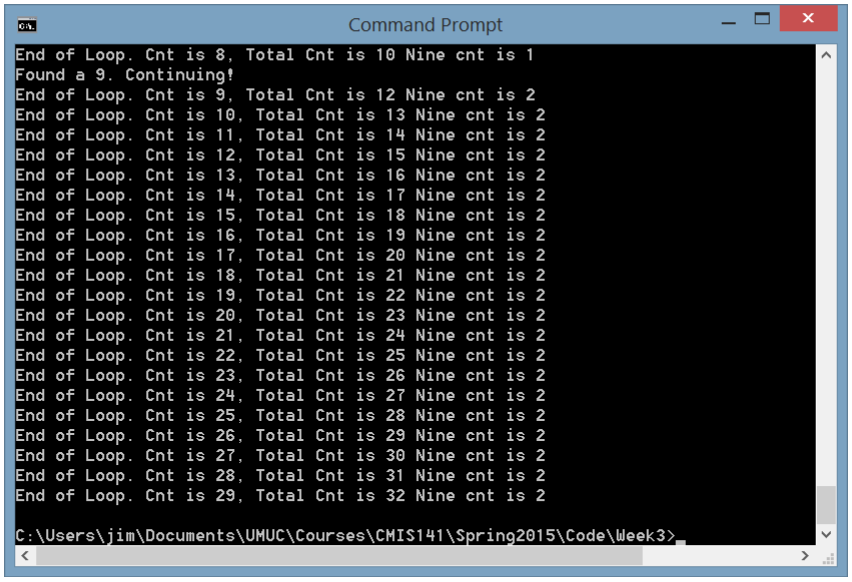
```
for loop: Random value is 2
for loop: Random value is 1
for loop: Random value is 0
for loop: Random value is 6
for loop: Random value is 9
End of Loop. Cnt is 0
End of Loop. Cnt is 1
End of Loop. Cnt is 2
End of Loop. Cnt is 3
End of Loop. Cnt is 4
End of Loop. Cnt is 5
End of Loop. Cnt is 6
End of Loop. Cnt is 7
End of Loop. Cnt is 8
End of Loop. Cnt is 9
End of Loop. Cnt is 10
End of Loop. Cnt is 11
End of Loop. Cnt is 12
End of Loop. Cnt is 13
End of Loop. Cnt is 14
End of Loop. Cnt is 15
End of Loop. Cnt is 16
Found the first 9. Exiting Loop!

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>
```

You should experiment with the code by changing the number of integers generated or the break condition.

e. The `continue` statement allows you to continue by skipping to the end of the loop's body and evaluates the boolean expression that controls the loop. The key is skipping to the end of body and evaluating the boolean expression. The last statement is not evaluated. So in our case, the cnt++ would not be evaluated. This results in possibly incrementing through the loop longer. This is useful when we are searching through a set of data looking for specific elements. To demonstrate this functionality, add the following code the main() method. You can add this after the `break` code statements.

```
// Continue
// Continue if value ==9
int totalCnt = 0;
int nineCnt = 0;
 cnt = 0;
while (cnt < maxLoop) {
    totalCnt++;
    randInt = (int) (Math.random() * 10);
    // print results if the random value ==9
    if (randInt == 9) {
      System.out.println("Found a 9. Continuing!");
      nineCnt++;
      continue;
    }
      System.out.println("End of Loop. Cnt is " +
         cnt + "," + " Total Cnt is " + totalCnt +
         " Nine cnt is " + nineCnt);
      cnt++;
  }
```

Compiling and executing the code will result in the following output when 30 integers are generated. Note, your output will be different depending upon the specific numbers generated. Notice each time a 9 is found the loop continues without incrementing the cnt variable. In this example, 2 nines were generated and making the total number of loop counts, as shown in the totalCnt variable, increase by 2. You should take your time with this example. Experiment by running the code multiple times to view and analyze the results.



```
End of Loop. Cnt is 8, Total Cnt is 10 Nine cnt is 1
Found a 9. Continuing!
End of Loop. Cnt is 9, Total Cnt is 12 Nine cnt is 2
End of Loop. Cnt is 10, Total Cnt is 13 Nine cnt is 2
End of Loop. Cnt is 11, Total Cnt is 14 Nine cnt is 2
End of Loop. Cnt is 12, Total Cnt is 15 Nine cnt is 2
End of Loop. Cnt is 13, Total Cnt is 16 Nine cnt is 2
End of Loop. Cnt is 14, Total Cnt is 17 Nine cnt is 2
End of Loop. Cnt is 15, Total Cnt is 18 Nine cnt is 2
End of Loop. Cnt is 16, Total Cnt is 19 Nine cnt is 2
End of Loop. Cnt is 17, Total Cnt is 20 Nine cnt is 2
End of Loop. Cnt is 18, Total Cnt is 21 Nine cnt is 2
End of Loop. Cnt is 19, Total Cnt is 22 Nine cnt is 2
End of Loop. Cnt is 20, Total Cnt is 23 Nine cnt is 2
End of Loop. Cnt is 21, Total Cnt is 24 Nine cnt is 2
End of Loop. Cnt is 22, Total Cnt is 25 Nine cnt is 2
End of Loop. Cnt is 23, Total Cnt is 26 Nine cnt is 2
End of Loop. Cnt is 24, Total Cnt is 27 Nine cnt is 2
End of Loop. Cnt is 25, Total Cnt is 28 Nine cnt is 2
End of Loop. Cnt is 26, Total Cnt is 29 Nine cnt is 2
End of Loop. Cnt is 27, Total Cnt is 30 Nine cnt is 2
End of Loop. Cnt is 28, Total Cnt is 31 Nine cnt is 2
End of Loop. Cnt is 29, Total Cnt is 32 Nine cnt is 2

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>_
```

Now it is your turn. Try the following exercise:

Create a Java class named MyLoopsDemo using your favorite text editor. Be sure you name the file "MyLoopsDemo.java". Use a while loop to provide functionality to print all of the available byte values from -128 to 127. Print 10 bytes per line with each byte separated by tabs. Repeat the same functionality using a do-while and a for loop.

Exercise 5 – Design, develop and test nested repetition statements in Java

Once you are comfortable using the sequential, selection and repetition programming statements, you can begin building more interesting and complex programs. These programs can contain nested loops, where one loop is nested inside another loop. You can also design loops within loops within loops. Within each of these loops you can add programming statements to provide the desired functionality of the applications.

For this exercise we will build a program that uses a nested loop to calculate and then display the multiplication table with values ranging from 0 to 10. We will then use nested loops to sum products at each iteration of a nested loop.

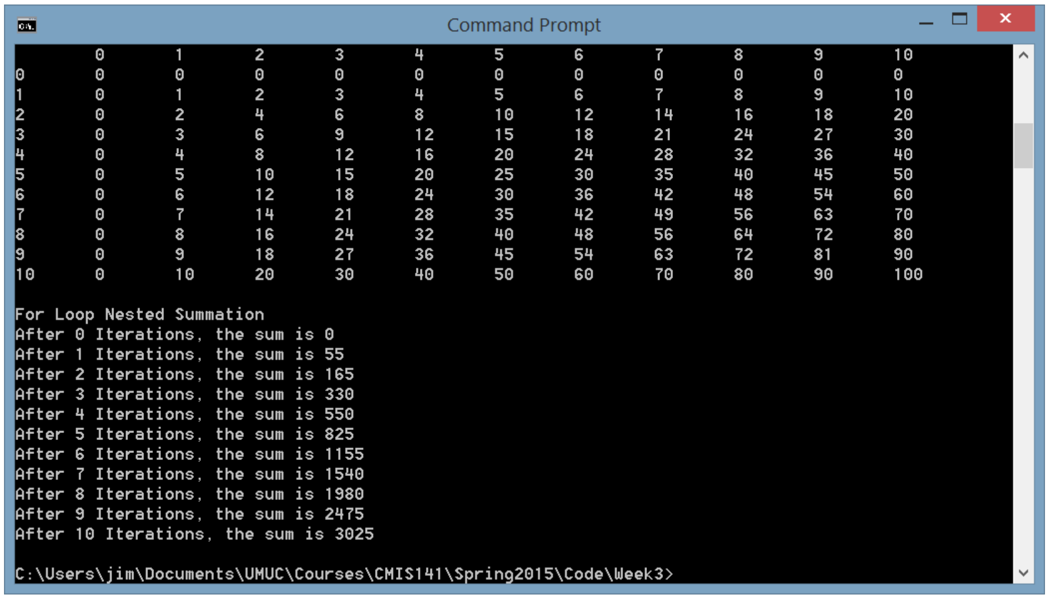
a. Open your favorite text editor and type (or cut and paste) the following code:

```
/*
* File: NestedLoopsDemo.java
* Author: Dr. Robertson
* Date: January 1, 2015
* Purpose: This program demonstrates the use of Java's
* nested repetition statements (while, do-while and for)
* The multiplication table from 1-10 is calculated
* and displayed. The summation of the products
* is also calculated.
*/

public class NestedLoopsDemo {
    public static void main(String[] args)  {

      // Define Number of Rows and Columns as Constants
      final int MAXROWS = 10;
      final int MAXCOLS = 10;

      System.out.println("Welcome to the NestedLoops Demo");

      // While Loop
      int rowCnt = 0;
      int colCnt = 0;

      // Print the Header
      System.out.println("");
      System.out.println("While Loop Multiplication Table");
      System.out.println("\t0\t1\t2\t3\t4\t5\t6\t7\t8\t9\t10");

      while (rowCnt <= MAXROWS) {
         // Print the Left column
         // Reset colCnt
         colCnt = 0;
         System.out.print(rowCnt + "\t");
         while (colCnt <= MAXCOLS) {
           System.out.print(colCnt*rowCnt + "\t");
             colCnt++;
         }
          System.out.print("\n");
            rowCnt++;
        }

      // Perform the same with a do-while
      // Reset values
      rowCnt=0;
      colCnt=0;
      System.out.println("");
      System.out.println("Do-While Loop Multiplication Table");
      System.out.println("\t0\t1\t2\t3\t4\t5\t6\t7\t8\t9\t10");
      do  {
         // Print the Left column
         // Reset colCnt
         colCnt = 0;
        System.out.print(rowCnt + "\t");
        do  {
           System.out.print(colCnt*rowCnt + "\t");
             colCnt++;
         } while (colCnt <=MAXCOLS);

          System.out.print("\n");
            rowCnt++;
        } while (rowCnt <=MAXROWS);

      // Perform the same with for loop
      // Reset values
```

```
        rowCnt=0;
        colCnt=0;
        System.out.println("");
        System.out.println("For Loop Multiplication Table");
        System.out.println("\t0\t1\t2\t3\t4\t5\t6\t7\t8\t9\t10");

        for (rowCnt=0; rowCnt <= MAXROWS; rowCnt++) {
           // Print the Left column
            System.out.print(rowCnt + "\t");
          for (colCnt=0; colCnt<= MAXCOLS; colCnt++) {
             System.out.print(colCnt*rowCnt + "\t");
           }
            System.out.print("\n");
          }

        System.out.println("");
        System.out.println("For Loop Nested Summation");
        int productSum = 0;
        // Finally we use nested loops
        // And sum a variable to print the sum of products
        for (rowCnt=0; rowCnt <= MAXROWS; rowCnt++) {
          for (colCnt=0; colCnt<= MAXCOLS; colCnt++) {
              productSum += rowCnt*colCnt;
           }
            System.out.print("After " + rowCnt +
                 " Iterations, the sum is " + productSum + "\n");
          }
      }
}
```

b. Save the file as "NestedLoopsDemo.java" in a location of your choice.

c. Compile and run the code. The results should look similar to this:



Command Prompt

```
        0       1       2       3       4       5       6       7       8       9       10
0       0       0       0       0       0       0       0       0       0       0       0
1       0       1       2       3       4       5       6       7       8       9       10
2       0       2       4       6       8       10      12      14      16      18      20
3       0       3       6       9       12      15      18      21      24      27      30
4       0       4       8       12      16      20      24      28      32      36      40
5       0       5       10      15      20      25      30      35      40      45      50
6       0       6       12      18      24      30      36      42      48      54      60
7       0       7       14      21      28      35      42      49      56      63      70
8       0       8       16      24      32      40      48      56      64      72      80
9       0       9       18      27      36      45      54      63      72      81      90
10      0       10      20      30      40      50      60      70      80      90      100

For Loop Nested Summation
After 0 Iterations, the sum is 0
After 1 Iterations, the sum is 55
After 2 Iterations, the sum is 165
After 3 Iterations, the sum is 330
After 4 Iterations, the sum is 550
After 5 Iterations, the sum is 825
After 6 Iterations, the sum is 1155
After 7 Iterations, the sum is 1540
After 8 Iterations, the sum is 1980
After 9 Iterations, the sum is 2475
After 10 Iterations, the sum is 3025

C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week3>
```

Be sure to scroll up to view the output of all 3 multiplication tables. The tables were generated using while, do-while and for statements. Notice the attention to resetting the rowCnt and colCnt variables at the proper times. The while and do-while statements can be trickier for nested loops as counters may need to be reset.

For the Nested Loop Summation, notice the summation takes place in the inner loop. It also uses the short-cut operator (+=). This is the same as using

```
    productSum = productSum + rowCnt*colCnt;
```

When analyzing the code of nested loops, you may need to walk through each iteration to better understand how the counters work. Although tedious, this technique is useful and may help you design more complex nested loops.

Now it is your turn. Try the following exercise:

Create a Java class named MyNestedLoopsDemo using your favorite text editor. Be sure you name the file “MyNestedLoopsDemo.java”. Create an application that sums the products of the multiplication table from the values of 11 to 99. (Hint, this is similar to what we did for multiplication tables with the values of 0 to 10). Print the final sum of all of the products along with the average of the product values.