CMSC 113 – COMPUTER SCIENCE 1
Lab#5 1-`StdIn` & `StdOut` Libraries – Practice

In this lab you will learn and practice how to use the **`StdIn`** and **`StdOut`** libraries for doing some interactive user input as well as formatted output and also use Linux's I/O redirection features. See Section 1.5 of your text.

Task#1: Review – `StdOut`

The `StdOut` library provides four useful output functions, as show below.

public class StdOut	
void print(String s)	*print `s` to standard output*
void println(String s)	*print `s` and a newline to standard output*
void println()	*print a newline to standard output*
void printf(String format, ...)	*print the arguments to standard output, as specified by the format string `format`*

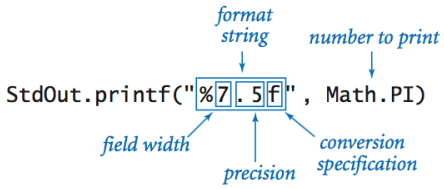
The `StdOut.print()` and `StdOut.println()` functions work very similar to the `System.out.print()` and `System.out.println()` functions. The `StdOut.printf()` is a powerful output function that provides a way to nicely format the output of your programs. The general form of the `StdOut.printf()` can be described as:

`StdOut.printf(`***<how to print>***`, `***<what to print>***`);`

<how to print> is a String (called a *format string*) that encodes exactly how the string to be printed will appear. ***<what to print>*** is a comma-separated sequence of variables (there may be zero or more). Here are some simple examples:

```
StdOut.printf("\n")        // This is equivalent to StdOut.println() or System.out.println()
StdOut.printf("Hello, World!\n")      // This is equivalent to StdOut.println("Hello, World!")
                                      // or System.out.println("Hello, World!")
StdOut.printf("%d\n", x)   // where x is an int variable
                           // This is equivalent to StdOut.println(x) or System.out.println(x);
StdOut.printf("%6d\n", x) // Prints the value of x so that it takes exactly 6 spaces (right justified)
StdOut.printf("%-6d\n", x) // Prints the value of x so that it takes exactly 6 spaces (left justified)
StdOut.printf("%d %d\n", x, Y) // Prints the values of x and y separated by a SPACE
```

The following figure describes more details of the `StdOut.printf()` function:



The codes used in the print specification (for example, **f** in the figure above, or **d** in examples above) indicate the type of value to be printed: **d** (`int`), **f** (`double`), **s** (`String`), **b** (`boolean`). Before proceeding, please be sure to review the information above.

Task#2: Using `StdOut.printf()`

Read the program below carefully.

```
public class Roots {
   // Prints out a table of Square Roots of numbers from 1 to 10
   public static void main(String[] args) {
      int n = 10;

      for (int i=1; i <= n; i++) {
         StdOut.printf("%d. %f\n", i, Math.sqrt(i));
      }
   } // main()
} // class Roots
```

Write and run the program above. Please ensure that you have typed the program EXACTLY as shown above. Compile and run it. Observe the results.

Next, modify the print statement to the following:

```
StdOut.printf("%3d. %5.2f\n", i, Math.sqrt(i));
```

Compile and run the program and observe how the table is now formatted, and the values of square roots are nicely rounded to two decimals places. Play with some other format values and see how the formatting changes.

Finally, add commands to the program above to produce a table that looks exactly like the one shown below:

```
  #   Square Root
-----------------
  1.  1.00
  2.  1.41
  3.  1.73
  4.  2.00
  5.  2.24
  6.  2.45
  7.  2.65
  8.  2.83
  9.  3.00
 10.  3.16
-----------------
```

Show your resulting program and its output to the instructor.

Task 3: Review – StdIn

The **StdIn** library provides several useful input functions to perform interactive input of various types of data. The table below summarizes all the functions available in StdIn library:

public class StdIn

methods for reading individual tokens from standard input		
boolean	isEmpty()	*is standard input empty (or only whitespace)?*
int	readInt()	*read a token, convert it to an int, and return it*
double	readDouble()	*read a token, convert it to a double, and return it*
boolean	readBoolean()	*read a token, convert it to a boolean, and return it*
String	readString()	*read a token and return it as a String*
methods for reading characters from standard input		
boolean	hasNextChar()	*does standard input have any remaining characters?*
char	readChar()	*read a character from standard input and return it*
methods for reading lines from standard input		
boolean	hasNextLine()	*does standard input have a next line?*
String	readLine()	*read the rest of the line and return it as a String*
methods for reading the rest of standard input		
int[]	readAllInts()	*read all remaining tokens and return them as an int array*
double[]	readAllDoubles()	*read all remaining tokens and return them as a double array*
boolean[]	readAllBooleans()	*read all remaining tokens and return them as a boolean array*
String[]	readAllStrings()	*read all remaining tokens and return them as a String array*
String[]	readAllLines()	*read all remaining lines and return them as a String array*
String	readAll()	*read the rest of the input and return it as a String*

Review these quickly and focus especially on the first group of functions. We will use some of these next.

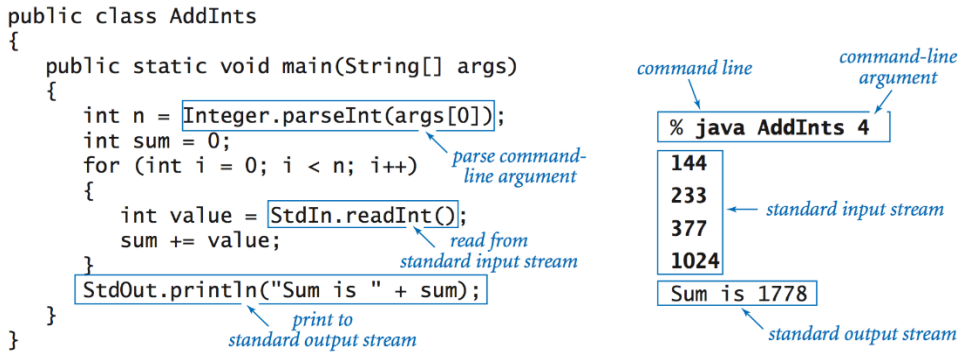
Task 4: First, consider the program shown below:

```
public class Hello {
   // Inputs a name and says Hello
   public static void main(String[] args) {
      String name;
      StdOut.print("Please enter your name: ");
      name = StdIn.readString();

      StdOut.printf("Hello, %s!\n", name);
   } // main()
} // class Hello
```

The program above prompts the user to enter a name (using `Std.ReadString()`) and then prints out a greeting (using `StdOut.printf()`). Enter, compile, and run the program above and make sure you understand how the simple interaction is coded.

Task 5: Read, study, enter, compile, and run the program below:



Next, create a small data file (**`data.txt`**) in the same directory as your `AddInt.java` program. In it, enter the four numbers shown in the example above (144, 233, 377, and 1024) so that one number appears on each line. Once done, run your program using the command:

```
$ java AddInts 4 < data.txt
```

What happened? You just used **Linux's I/O redirection** so that your program, instead of taking input from the keyboard, takes it from the file **`data.txt`**.

Try create a much longer data file, say with 20 numbers in it. Then run the program:

```
$ java AddInts 20 < data.txt
```

Task 5: Write a program **`RandomInts.java`**, that generates **n** random numbers in the range [1..1000] and prints them out on the screen (one number per line). Here is an example use:

```
$ java RandomInts 5
765
211
90
512
67
```

Once done, You can use the two programs above (`RandomInts` to generate 20 random numbers, and `AddInts` to add them):

```
$ java RandomInts 20 | java AddInts 20
```

Task 6: Modify `AddInts` to compute the average of all the integers and print it out (only up to 2 decimal places). Test your program using the commands:

```
$ Java AddInts 4
144
233
377
1024
Average is 444.50
```

Once correct, try the following:

```
$ java RandomInts 1000 | java AddInts 1000
```