**IT 1090C Computer Programming I   
IT 6090C Java Programming**

**Prof. Tom Wulf**

# Lab 06 Input with Scanner

**20 points**

# Learning Goals:

* **Use Scanner to get console input from the user:**
  + Use next() and nextLine() to read Strings
  + Use nextInt() to read int values
  + Use nextDouble() to read double values
* **Safely input numbers by checking before reading them:**
  + hasNextInt() before reading with nextInt()
  + hasNextDouble() before reading with nextDouble()
* **Use the java if.. else control structure**

Mini-lecture:  
We use the Scanner class to get input from the user through the console. We have to import the Scanner class and create an instance of Scanner using the default input stream System.in.  
  
 **import java.util.Scanner;** // imports go at the top of the file BEFORE the class!

**Scanner in = new Scanner(System.in);** // it is very common to name the Scanner “in” or *console*  
  
// Scanner has a variety of methods that read data of a specific Java type:

int val = in.**nextInt()**; // reads an int value throws an error if the input cannot be an int

double doubleVal = in.**nextDouble()**; // reads a double value error if the input isn’t a double

String line = in.**nextLine**(); // reads an entire line to a \n newline character as a String

String word = in.**next**(); // reads part of the line up to a delimiter (space) or the \n …  
  
The methods that return numeric types can throw an error if the input cannot be parsed into the correct type. The following methods allow you to safely determine if you can successfully read the numeric value:

in.**hasNextInt();** // returns true if nextInt will succeed

in.**hasNextDouble();** // returns true if nextDouble will succeed

Since anything the user types can be read in as a legal java String, the next() and nextLine() methods cannot generate an error. We will need to use the java if and if .. else structures for this. So let’s take a look at them now.  
  
Here is our key example for using the hasNextX to bulletproof the code:  
  
**import java.util.Scanner; // before the class at top of the file**

**// this code in main:**

**Scanner in = new Scanner(System.in);  
  
double wage = 0;**

**String trash = “”; // use for bad input which will read as a String**

**System.out.print(“Enter your hourly wage: “);**

**if(in.hasNextDouble())**

**{  
 // OK safe to read in a double  
 wage = in.nextDouble();**

**in.nextLine(); // clears the newline from the buffer  
}**

**else**

**{**

**// Not a double can’t use nextDouble() read as String with nextLine() instead**

**trash = in.nextLine();**

**System.out.println(“\nYou said your wage was: “ + trash);**

**System.out.println(“Run the program again and enter a valid amount!”);**

**System.exit(0); // terminate the program**

**}**

So this block of code represents input for one variable. You will use this block multiple times once for each variable that you need to read in. After you have all the inputs, complete the calc and display the results. Of course the example is for a double var you have to modify it slightly for an int var.

For the extra credit you have to create a random number between 1 and 10:  
  
**import java.util.Random;**

**Random generator = new Random();**

**int val = generator.nextInt(10) + 1; val is 1 to 10**

Lab Directions:  
Complete each of the programs here. Create a separate IntelliJ project for each program using the name I specified. You can help one another during the lab session but you should enter and submit the code yourself. Make sure you understand any code you submit.

**If your instructor directed you to use GitHub, create a repo for each task. Be sure to commit and push your project!!!**

# Task 1: (5 points)

**Project name: Lab\_06\_01\_CtoFConverter**

**A program that converts an inputted temperature in C and provides the equivalent temperature in F. Hint: Google is your friend! Given C, solve for F. Again, check for a valid input value and only respond with the F value if you got it, otherwise output an appropriate error msg to the user. Testing: 3 conditions: Bad Input, then test for the known freezing and boiling points.**

**Text

Description automatically generated  
  
EMBED SCREEN SHOT(S) OR COPY THE OUTPUT WINDOW OF INTELLIJ HERE SHOWING YOUR PROGRAM TEST RUN(S):**

Task 2: (5 points)  
**Project name: Lab\_06\_02\_FuelCosts**

**Write a program that asks the user to input**

* **The number of gallons of gas in the tank**
* **The fuel efficiency in miles per gallon**
* **The price of gas per gallon**

**Then print the cost per 100 miles and how far the car can go with the gas in the tank. Again, check for valid input and exit with an error msg if you do not have it. Testing: here just use some reasonable values that you can inspect the calculations and determine they are correct. Be sure to check of integer calculations truncating the results.**

**Text

Description automatically generated**

**EMBED SCREEN SHOT(S) OR COPY THE OUTPUT WINDOW OF NETBEANS HERE SHOWING YOUR PROGRAM TEST RUN(S):**

Task 3: (5 points)  
**Project name: Lab\_06\_03\_RectangleInfo**

**Write a program that asks the user for the lengths of the sides of a rectangle. Again, check for valid input and exit with an error msg if you don’t get it. Testing: use some known values to confirm that the calculations are correct. E.g. 3 – 4 - 5 triangle >> 3 X 4 rectangle  
  
Then print**

* **The area and perimeter of the rectangle**
* **The length of the diagonal (use the Pythagorean theorem)**

Graphical user interface, text

Description automatically generated

**EMBED SCREEN SHOT(S) OR COPY THE OUTPUT WINDOW OF NETBEANS HERE SHOWING YOUR PROGRAM TEST RUN(S):**

Task 4: (5 points)  
**Project name: Lab\_06\_04\_MetricConverter  
Main file name: metric.java  
  
Write a program that prompts the user for a measurement in meters and then converts it to miles, feet, and inches. Again, check for valid input and exit with an error msg if you don’t get it. Testing: use some known values here and watch for integer truncation.**

**Graphical user interface, text

Description automatically generated**

**EMBED SCREEN SHOT(S) OR COPY THE OUTPUT WINDOW OF NETBEANS HERE SHOWING YOUR PROGRAM TEST RUN(S):**

Task 5 Graduate or Extra Credit: (3 points)  
**Project name: Lab\_06\_05\_HighorLow  
  
Write a program that has the computer generate a random int value between 1 to 10 inclusive. The program then asks the user to guess the number with a single guess. The program displays the random number and then indicates if the users guess was high low or on the money!**

**Graphical user interface, text

Description automatically generated**

**EMBED SCREEN SHOT(S) OR COPY THE OUTPUT WINDOW OF NETBEANS HERE SHOWING YOUR PROGRAM TEST RUN(S):**

# Submitting your work:

Everyone submits this file **Lastname\_Firstname\_Lab\_06.docx** using your name.

If you were directed to use GitHub, create an additional word.docx file and include the link to each of the repos yo created for these tasks. Submit it.  
  
If not, create a new compressed .zip archive folder. (Don’t give me any other type of archive, it will be returned to you ungraded!) called **Lastname\_Firstname\_Lab\_06.zip** using your name.

Place each of your IntelliJ project folders in this archive. (**Do not individually zip the projects!**) Place this word doc file with your screen shots in the archive as well. (Don’t put this in the individual project folders, put it in the top level in the archive so I can access it easily.)

Use the Canvas Assignment mechanism to submit your archive file or repo link file in addition to this word.docx.

If you did the extra credit option, which I urge you to try, submit the entire assignment again exactly the same way using the extra credit link.