## The Math Class (Outsource: Math Class Supplement)

The Math Class includes a number of constants and methods you can use to perform common mathematical functions. A commonly used constant found in the Math class is **Math.PI** which is defined as **3.14159265358979323846.** Also included in the Math class are many methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions. Some of the methods found in the Math Class are:

| Method Summary | | |
|---|---|---|
| *static* **double** | **abs(***double* **a)** | Returns the absolute value of a `double` value. |
| *static* **float** | **abs(***float* **a)** | Returns the absolute value of a float value. |
| *static* **int** | **abs(***int* **a)** | Returns the absolute value of an integer value. |
| *static* **long** | **abs(***long* **a)** | Returns the absolute value of a long value. |
| *static* **double** | **ceil(***double* **a)** | Returns the smallest whole value that is not less than the argument. |
| *static* double | **cos(**double **a)** | Returns the trigonometric cosine of an angle. |
| *static* **double** | **floor(***double* **a)** | Returns the largest whole value that is not greater than the argument. |
| *static double* | **max(***double* **a,** *double* **b)** | Returns the greater of two `double` values. |
| *static int* | **max(***int* **a,** *int* **b)** | Returns the greater of two `int` values. |
| *static double* | **min(***double* **a,** *double* **b)** | Returns the smaller of two double values. |
| *static int* | **min(***int* **a,** *int* **b)** | Returns the smaller of two int values. |
| *static* **double** | **pow(***double* **a,** *double* **b)** | Returns the value of the first argument raised to the power of the second argument. |
| *static* **double** | **random()** | Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`. |
| *static* **double** | **sqrt(***double* **a)** | Returns the correctly rounded positive square root of a `double` value. |
| *static* **double** | **toDegrees(***double* **angrad)** | Converts an angle measured in radians to the equivalent angle measured in degrees. |
| *static* **double** | **toRadians(***double* **angdeg)** | Converts an angle measured in degrees to the equivalent angle measured in radians. |

## Random Numbers

Random numbers are used to generate chance in computer programs. For example, if you want to simulate a roll of a die, you need to obtain a random number from 1 to 6 (with any one of these values appearing with the same probability). "Random" numbers are not truly random – their sequence is generated using a certain formula – but they are good enough for most applications. The Java Math class provides a **random()** method that returns a random number as a double. The number returned will be somewhere between 0.000 and 0.999. Since this seldom meets the needs of a program you will usually have to work with the numbers a bit. For example, if you wanted to generate a random number between 0 and 9, you could simply multiply the

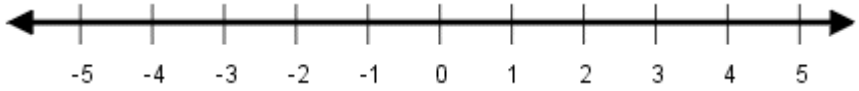
results of a call to Math.random( ) times 10 and type cast the resulting value into an integer. I.E. **int num = (int) (Math.random() * 10)**. If you were simulating a card game then the possibilities would be between 1 and 52 since there are 52 different cards in a deck. **int thisCard = (int) (Math.random() * 52) + 1**. Since the call to **Math.random()** * 52 results in a number between 0 and 51 we simply add 1 to achieve the range of numbers that we desire.

## LAB 06 - ASSIGNMENT

## Lab 06A - 50 points

### OBJECTIVE

WAP to calculate the distance between two numbers on a number line. The two numbers will be entered from the keyboard.




### FIELD SUMMARY

- **int a** – the real first number on the number line.
- **int b** – the second number on the number line.
- **int distance** – the distance between the two numbers.

### METHOD SUMMARY

- **main** – instantiate an object of your class. Call input, process, and output.
- **input** – declare a `Scanner` object and read two integers from the keyboard using appropriate prompts.
- **process** – calculate the distance between the two points using the appropriate Math method. The distance formula is: `dx = | a – b |`
- **output** – display the results on the console screen.

### SAMPLE KEYBOARD INPUT

*Enter two numbers:* **3 -5**

### SAMPLE OUTPUT

**The distance from 3 to -5 is 8.**

## Lab 06B - 55 points

### OBJECTIVE

WAP to (a) find the area of a square whose side length is read from a data file.
(b) find the volume of a sphere whose radius is read from a data file.
(c) find the length of the side of a square whose area is read from a data file.
The name of the data file is ("lab06b.dat").

### FIELD SUMMARY

- **int s** – the length of the side of a square (read from the data file).
- **double r** – the length of the radius of a sphere a data file (read from the data file).
- **int a** – the area of a square (read from the data file).
- **double area** – the area of a square.
- **double volume** – the volume of a sphere.
- **double side** – the side length of a square.

### METHOD SUMMARY

- **main** – instantiate an object of your class. Call input, process, and output.
- **input** – declare a `Scanner` object and read the length of the side of a square, the length of the radius of a sphere, and the area of a square from a data file.
- **process** – calculate the area of a square, volume of a sphere, and length of the side of a square when given the area using the appropriate Math methods.
- **output** – display the results on the console screen. Format all doubles to display *two* decimal places suppressing trailing zeros.

### SAMPLE DATA FILE INPUT

**5 6.4 38**

### SAMPLE OUTPUT

**The area of a square whose side length is 5 is 25.**
**The volume of a sphere whose radius is 6.4 is 1,098.07.**
**The side length of a square whose area is 38 is 6.16.**

## Lab 06C - 60 points

### OBJECTIVE

WAP to find the minimum number of 4 X 8 sheets of plywood it would take to build a storage shed when the amount needed in square feet is input from the keyboard. *Use a constant to represent the size of the plywood.*

### FIELD SUMMARY

- **final int boardSize** – the size of each sheet of plywood.
- **int area** – the square footage required for the storage shed.
- **int boards** – the number of boards required.

### METHOD SUMMARY

- **main** – instantiate an object of your class. Call input, process, and output.
- **input** – declare a `Scanner` object and read the amount of plywood using appropriate prompts.
- **process** – calculate the number of sheets of plywood required to build the storage. The number of sheets of plywood is calculated as the total square footage required for the storage shed divided by the total square footage of one sheet of plywood. If there is a fraction involved, i.e. **6.2** boards, then **7** boards would have to be purchased.
- **output** – display the results on the console screen.

### SAMPLE KEYBOARD INPUT

*Enter the number of square feet:* **200**

### SAMPLE OUTPUT

**A storage shed requiring 200 square feet of plywood will need 7 sheets of plywood to finish the job.**

## Lab 06D - 70 points

**OBJECTIVE**

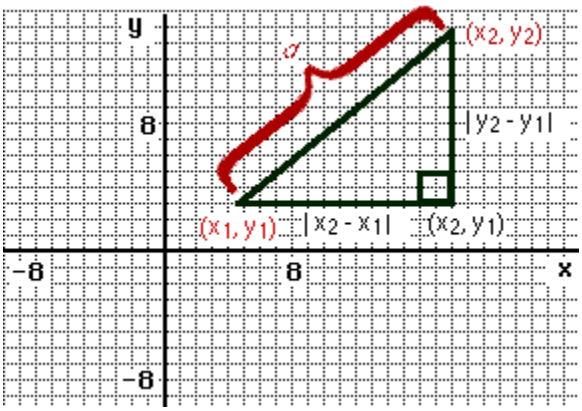
Open the file *ErrorProneEllie.java* and debug Error Prone Ellie's latest program. Use the following chart format to describe the syntactical errors discovered and what you did to eliminate each error. To receive credit for this assignment you must turn in the chart and demonstrate a working program.

| Line # | Error as reported by Java | What the error really was | What you did to fix the error. |
|---|---|---|---|
| | | | |
| | | | |

## Lab 06E - 80 points

### OBJECTIVE

WAP to calculate the distance between two points in a coordinate plane, reading the x and y coordinate for each point from the keyboard. Input the coordinates in the order - **x1, y1, x2, y2.** Format all doubles to display one decimal place suppressing trailing zeros.




The distance formula is $d = \sqrt{(x2-x1)^2 + (y2-y1)^2}$

### SAMPLE KEYBOARD INPUT

*Enter the x and y coordinate for the first point:* **3 5**
*Enter the x and y coordinate for the second point:* **6 10**

### SAMPLE OUTPUT

**The distance between the points (3, 5) and (6, 10) is 5.8.**

## Lab 06F - 90 points

### OBJECTIVE

WAP that simulates the rolling of a pair of dice. For each die in the pair, the program should generate a random number between 1 and 6 (inclusive). Display the results of the roll of the dice by showing the number of each die and the total roll (the sum of the two dice).

### SAMPLE OUTPUT

**You rolled 6 and 5**
**Total This Roll: 11**

## Lab 06G - 100 points

### OBJECTIVE

The law of cosines states that the third side of an `oblique` triangle (any triangle that is not a right triangle) can be found by knowing the length of any two sides and the measure of the included angle (SAS). WAP to input two sides and the included angle of an `oblique` triangle and find the length of the third side. **Note**: Angle **C** will be entered as a measurement of degrees. The argument ***a*** in the **Math.cos(***double* **a)** method is a measurement of radians not degrees. You must convert the angle measurement from degrees to radians before using this method.

The formula for the law of cosines is $c = \sqrt{a^2 + b^2 - 2ab\cos(C)}$

### SAMPLE KEYBOARD INPUT

*Enter the lengths of two sides and their included angle:* **5.18 6.0 60**

### SAMPLE OUTPUT

**A triangle with sides of length 5.18 and 6 and an included angle measuring 60 degrees has a third side of length 5.63.**

### SAMPLE KEYBOARD INPUT

*Enter the lengths of two sides and their included angle:* **45 67 35**

### SAMPLE OUTPUT

**A triangle with sides of length 45 and 67 and an included angle measuring 35 degrees has a third side of length 39.68.**