**Part 1: Conversion**

Write a Distance class that will hold a distance, in meters, and provide methods to get the distance in meters, inches and feet. The class should have the following field:

* meters – A double that holds a distance in meters.

The class should have the following methods:

* Constructor – The constructor accepts a meter distance (as a double) and stores it in the meters field.
* setMeters – The setMeters method accepts a meters distance (as a double) and stores it in the meters field.
* getMeters – Returns the value of the meters field, as a meters distance (no conversion required).
* getInches – Returns the value of the meters field converted to inches.
* getFeet – Returns the value of the meters field converted to feet.

Use the following formula to convert the meters temperature to inches:

*inches = meters \* 39.37*

In order to convert the meters distance to feet, you must first convert it to inches, and then divide by 12. Hint: Use the getInches() method inside of the getFeet() method!

Demonstrate the Distance class by writing a separate program that asks the user for a Fahrenheit temperature. The program should create an instance of the Distance class, with the value entered by the user passed to the constructor. The program should then call the object’s methods to display the distance in inches and feet. Call this program DistanceRunner.

For submission, I need the following items:

* Your source code (with comments!)
* The Console **output** of running your program. (NO OUTPUT = NO CREDIT)
* A UML Diagram of your Distance class, ideally done in LucidChart or draw.io. You can export your diagram as a .pdf and attach it to Blackboard.

**Part 2: Imaginary Numbers**

For the second part of Lab1, create a simple class that emulates some mathematical functions in relation to complex numbers. A complex number is a number of the form a + bi, where a is a real number and bi is an imaginary number. Create a class **Complex** having two private data members **real** and **imag** of type double. The class has two constructors, one default (no parameters) and one whose parameters initialize the instance variables. It also needs the following member functions, all of which are public:

* getReal – returns the real number component
* getImag **–** returns the imaginary number component
* setReal – sets the value of the real number component
* setImag – sets the value of the imaginary component
* addComplex – has one parameter, another Complex object to add to this one. This method does not return a value (it updates the calling object’s variables).
* subtractComplex - has one parameter, another Complex object to subtract to this one. This method does not return a value (it updates the calling object’s variables).
* multiplyComplex **-** has one parameter, another Complex object to multiply to this one. This method does not return a value (it updates the calling object’s variables).
* print– prints the data in the form a + bi

Hints: For complex numbers:

Addition: add the real and imaginary components separately:

(7 + 4i) + (23 – 12i) = 30 – 8i

Subtraction: subtract the real and imaginary components separately:

(7 + 4i) - (23 – 12i) = -16 + 16i

Multiplication: More complicated (will be discussed in lab).

You must write a driver program to test all of the above functions. You can call it ComplexRunner. I also want a UML diagram for the Complex class.

For submission, I need the following items:

* Your source code (with comments!)
* The Console **output** of running your program. (NO OUTPUT = NO CREDIT)
* A UML Diagram of your Complex class, ideally done in LucidChart or draw.io. You can export your diagram as a .pdf and attach it to Blackboard.