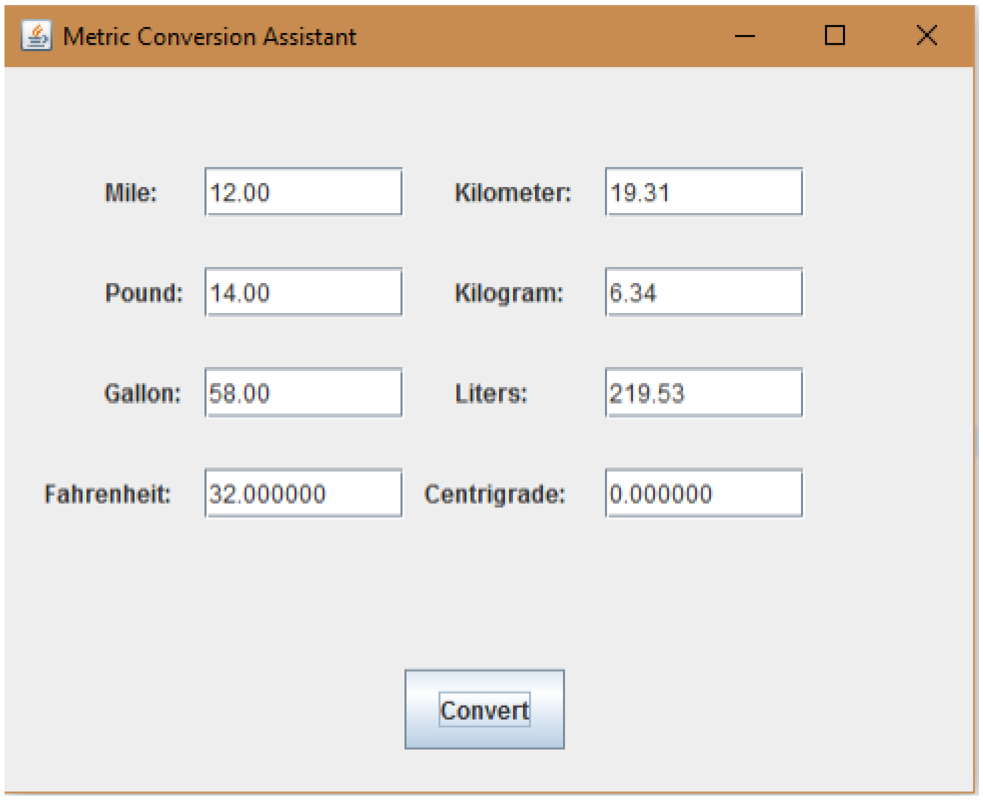
Home Work – Lesson -7 – Inner class

1. Create a GUI application to convert between the following pairs of units of measure by **clicking convert button**. Implement Anonymous Inner class for the listener using Lambda.

    1. miles and kilometers,
    2. pounds and kilograms,
    3. gallons and liters,
    4. Fahrenheit and Centigrade.



Metric Conversion Assistant

Mile:	12.00	Kilometer:	19.31
Pound:	14.00	Kilogram:	6.34
Gallon:	58.00	Liters:	219.53
Fahrenheit:	32.000000	Centrigrade:	0.000000

2. An array allows you to match objects of a particular type with integer indices. For instance, the array ["Harry", "Jim", "Sally"] matches 0 with "Harry", 1 with "Jim", and 2 with "Sally".

Create a class MyTable that will allow you to match objects of String type with a *character* index (where, for this lab, we will just use the characters ‘a’ – ‘z’ not allow for ‘A’-‘Z’). Your class should be able to be used like this:

```
MyTable t = new MyTable();
t.add('a', "Andrew");
t.add('b',"Billy");
t.add('c',"Charlie");
String s = t.get('b');
System.out.println(s);

//output
Billy
```

Implement MyTable by creating an inner class Entry whose constructor has the following signature:

```
Entry(char c, String s)
```

As an instance variable in MyTable, store an array with the following initialization:

```
Entry[] entries = new Entry[26];
```

Each of the 26 Entry instances corresponds to one of the lower case characters of the alphabet – 'a' through 'z'.

The add method will compute the position in the entries array that corresponds to the character passed in, and will create an Entry object to place into that position. For example, if a call add('b', "Billy") is made, the add method will compute that 'b' corresponds to position 1 in the entries array.

It will then create a new Entry instance, passing in the pair ('b',"Billy"), and place that new Entry instance in position 1 of the entries array.

If you add one more entry for the character b, example again if a call add(‘b’,”Bonu”), the new input will be stored in the index 1.

Entry and MyTable should also each implement a toString() method

```
public String toString()
```

The toString method in Entry should join the contents of its character and String variables with an arrow, as in the following:

a->Andrew

The toString method() of MyTable should adjoin the output of repeated calls to the toString() method of the objects stored in the Entry[] array.

Here is an example of how it should look: If the following appears in the main method:

```
MyTable t = new MyTable();
t.add('a', "Andrew");
t.add('b',"Billy");
t.add('w',"Willie");
System.out.println(t);
```

then the output should look like this:

```
a->Andrew
b->Billy
w->Willie
```

See the source code skeleton for this exercise below:

Hint : If you need, add some helper methods.

```
public class MyTable {
        private Entry[] entries;

        //returns the String that is matched with char c in the table
        public String get(char c){
                //implement
                return null;
        }
        //adds to the table a pair (c, s) so that s can be looked up using c
        public void add(char c, String s) {

                //implement
        }
        //returns a String consisting of nicely formatted display
        //of the contents of the table
        public String toString() {
                //implement
                return null;
        }
private class Entry {
                Entry(char ch, String str){
                        //implement
                }
                //returns a String of the form "ch->str"
                public String toString() {
                        //implement
                        return null;
                }
        }

}
```