

Lab Exercise 7**IT1050 – Object Oriented Concepts****Semester 2, 2023**

Objectives:

- Implementing constructors and destructors
- Creating static and dynamic objects
- Passing objects as arguments

Exercise 1:

In Exercise 1 we will implement the *Distance* class that allows us to handle length given in Feet and Inches. e.g. Although we are now using metric units like metres, we still occasionally use imperial units (e.g. a person's height = 5 feet 6 inches (5'6"))

- (a) In Visual C++, create a new Win32 Console Application project. Save the project in your Desktop. We will name the project as **Lab9**
- (b) Add a new Class to the project from the main menu.

Select **Project -> Add Class**

- (c) We will create a Class called **Distance**. When you specify the Class Name the Wizard creates the header file and the .cpp file.
- (d) Write the definition of the Distance class in **Distance.h** header file.

Distance
- feet : int - inches : float
+ Distance() + Distance(ft : int, in : float) + inputDistance() : void + printDistance() : void + ~Distance()

Lab Exercise 7**IT1050 – Object Oriented Concepts****Semester 2, 2023**

- (e) Implement Distance class in **Distance.cpp**.
- i) In the default constructor, set the feet and inches in to 0
 - ii) In the overloaded constructor, set the feet and inches to the values in the parameters of the constructor.
 - iii) In the *inputDistance()* method, you should get the feet and the inches through the keyboard.
 - iv) The *printDistance()* method should display the feet and the inches
 - v) Display a message “Distance deleted” with the values of the feet and inches variables, in the destructor.
- (f) In the main program do the following;
- i) Create a static object of the **Distance** class by using the default constructor.
 - ii) Display the details by using the *printDistance()* method.
 - iii) Create another object by using the overloaded constructor where the feet and inches will be set to 11 and 6.25.
 - iv) Display the details by using the *printDistance()* method.
 - v) When exiting the program observe the messages displayed by the destructor.

Exercise 2:

Modify the main program written above to do the following;

- i) Create a pointer of the **Distance** type and allocate memory dynamically for a new object.

`Distance * dist1 = new Distance();`
- ii) Display the details using *printDistance()* method. (use -> to call the methods of dynamic objects)

Lab Exercise 7**IT1050 – Object Oriented Concepts****Semester 2, 2023**

- iii) Create another dynamic object using the overloaded constructor.

`Distance * dist2 = new Distance(11, 6.25);`

- iv) Display the details using *printDistance()* method. (use -> to call the methods of dynamic objects)
- v) When exiting the program observe the messages displayed by the destructor.

Exercise 3:

- i) Modify the **Distance** class to add a method called *addDistance()* to send two objects of **Distance** as a parameter to the method and find the sum of the two distances.

```
void Distance::addDistance( Distance d2, Distance d3)
{
    inches = d2.inches + d3.inches;
    if (inches >= 12.0)
    {
        inches = inches - 12.0;
        feet++;
    }
    feet = feet + ( d2.feet + d3.feet);
}
```

e.g. 4'8" + 3'6" = 7'14" = 8'2"

Note : Here *d2* and *d3* are objects of the **Distance** Class. Since *addDistance()* is a method of the **Distance** class you have direct access to the private properties (*inches* and *feet*) of the objects *d2* and *d3*.

- ii) In the main program;
- Create two objects of **Distance** called *dist1* and *dist3* using default constructor.
 - Create another object of **Distance** called *dist3* using the overloaded constructor to set the *feet* and the *inches* to 11 and 6.25
 - Call the *inputDistance()* method to input a feet and a distance to object *dist1*
 - Call the *addDistance()* method for object *dist3* by sending *dist1* and *dist2* as parameters.

`dist3.addDistance(dist1, dist2);`

- e) Print the distance of each object using the *printDistance()* method.