



National University of Computer & Emerging Sciences, Karachi
Computer Science Department



Spring 2021, Lab Manual - 03

Course Code: CL-217

Course : Object Oriented Programming Lab

Contents:

1. Introduction to Classes & Objects

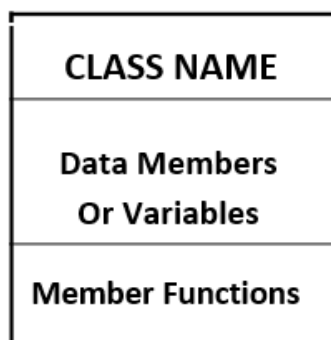
Introduction to Classes & Objects

- A **class** is a definition of objects of the same kind. In other words, a class is a blueprint, template, or prototype that defines and describes the *static attributes* and *dynamic behaviors* common to all objects of the same kind.

Classes are created using the keyword **class**. A class declaration defines a new type that links code and data. This new type is then used to declare objects of that class.

- A Class is a user defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behavior of the objects in a Class.
- A class member can be defined as public, private or protected. By default members would be assumed as private.

In the UML, a class icon can be subdivided into compartments. The top compartment is for the name of the class, the second is for the variables of the class, and the third is for the methods of the class.



```
class class-name
{
    access-specifier:
    data

    access-specifier: functions
};
```

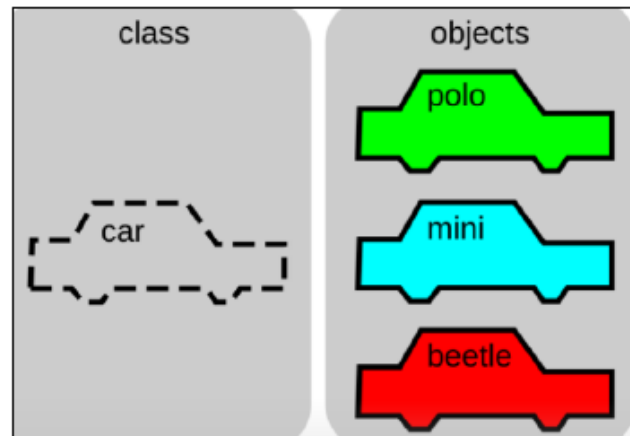
*Summarize***A class:**

- It's a blue print.
- It's a design or template.

An Object:

- It's an instance of a class.
- Implementation of a class.

NOTE: Classes are invisible, object are visible



CLASS NAME

By convention, the name of a user-defined class begins with a capital letter and, for readability, each subsequent word in the class name begins with **a capital letter**.

DATA MEMBERS

Consider the attributes of some real world objects:

Student– personal details(name,last name, mob: number,address etc) courses, cgpa, semester.

CAR – speedometer readings, amount of gas in its tank and what gear it is in.

MEMBER FUNCTIONS

Consider the operations of some real world objects:

Student– Can register in particular course, drop course, cgpa updation, can edit details s

CAR – accelerating (invoked by the driver), decelerating, turning and shifting gears.

These operations form the functions in program. Member functions define the class's behaviors.

Object

- An **Object** is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.
- In C++, when we define a variable of a class, we call it **instantiating** the class. The variable itself is called an **instance** of the class. A variable of a class type is also called an **object**. Instantiating a variable allocates memory for the object.

Syntax to Define Object in C++

```
className objectVariableName;
```

```
Student stdr;
```

```
CAR c;
```

Member Functions in Classes

There are 2 ways to define a member function:

- Inside class definition
- Outside class definition

1. Inside class definition

With an inline function, the compiler tries to expand the code in the body of the function in place of a call to the function.

Note that all the member functions defined inside the class definition are by default **inline**, but you can also make any non-class function inline by using keyword inline with them. Inline functions are actual functions, which are copied everywhere during compilation, like pre-processor macro, so the overhead of function calling is reduced.

2. Outside class definition

To define a member function outside the class definition we have to use the scope **resolution ::** operator along with class name and function name.

```
C++ > oop.cpp > main()
1  #include <iostream>
2  using namespace std;
3  class Student
4  {
5      public:
6          string StudentName;
7          int id;
8
9          // printname is not defined inside class definition
10         void printname();
11
12         // printid is defined inside class definition
13         void printid()
14         {
15             cout << " Student id is: " << id;
16         }
17     };
18
19     // Definition of printname using scope resolution operator ::
20     void Student::printname()
21     {
22         cout << " Student name is: " << StudentName;
23     }
24     int main() {
25
26         Student obj1;
27         obj1.StudentName = "xyz";
28         obj1.id=15;
29
30         // call printname()
31         obj1.printname();
32         cout << endl;
33
34         // call printid()
35         obj1.printid();
36         return 0;
37     }
```

Structures VS Classes

By default, all structure fields are public, or available to functions (like the main() function) that are outside the structure. Conversely, all class fields are private. That means they are not available for use outside the class. When you create a class, you can declare some fields to be private and some to be public. For example, in the real world, you might want your name to be public knowledge but your Social Security number, salary, or age to be private.

Reading Material:

<https://techdifferences.com/difference-between-structure-and-class.html>

<http://net-informations.com/faq/oops/struct.htm>

Sample C++ Code:

Code#1

```

#include<iostream>
using namespace std;

class Student
{
    private :
    string F_Name;
    string L_Name;

    public:

    void input_value()
    {
        cout << "Please Enter Your First Name\n";
        cin >> F_Name;
        cout << "Please Enter Your Last Name \n";
        cin >> L_Name;
    }

    void output_value()
    {
        cout << "Your Full Name is "<<F_Name<<" " <<L_Name;

    }
};

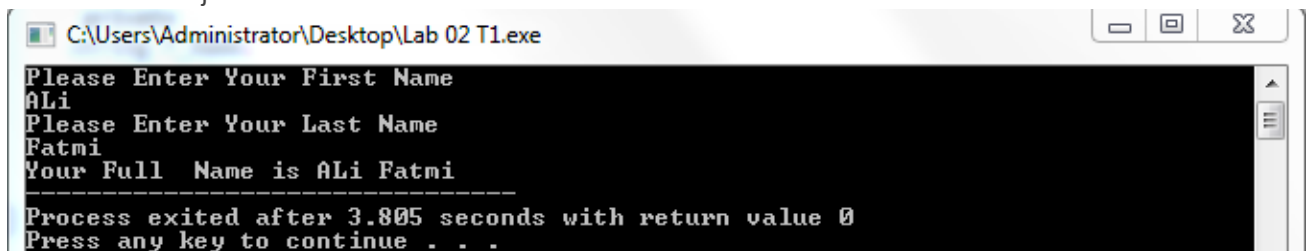
main()
{
    Student object;

    object.input_value();
    object.output_value();

    //object.variable; Will produce an error because variable is private

    return 0;
}

```



```

C:\Users\Administrator\Desktop\Lab 02 T1.exe
Please Enter Your First Name
ALi
Please Enter Your Last Name
Fatmi
Your Full Name is ALi Fatmi
-----
Process exited after 3.805 seconds with return value 0
Press any key to continue . . .

```

Code#2

```

#include <iostream>
using namespace std;

```

```

class Box {

    public:
        double Length;
        double Breadth;
        double Height;

        double Area() {
            return Length * Breadth;
        }

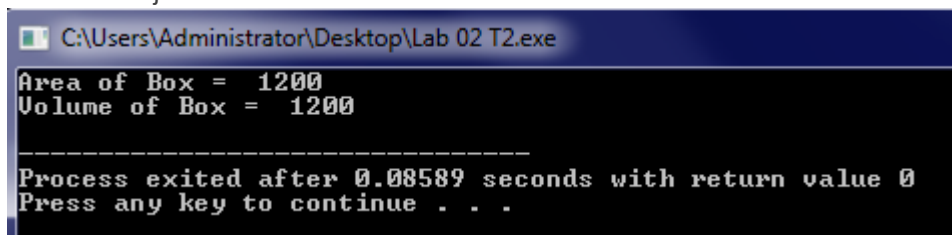
        double Volume() {
            return Length * Breadth * Height;
        }
};

int main ()
{
    Box obj;
    obj.Length = 30;
    obj.Breadth = 40;
    obj.Height = 60;

    cout << "Area of Box = " << obj.Area () << endl;
    cout << "Volume of Box = " << obj.Area () << endl;

    return 0;
}

```



```

C:\Users\Administrator\Desktop\Lab 02 T2.exe
Area of Box = 1200
Volume of Box = 1200
-----
Process exited after 0.08589 seconds with return value 0
Press any key to continue . . .

```

Code#3

```

#include <iostream>
using namespace std;

class Sample Class {
public:
    void o_method();
    void o_method2(int value);

    void i_method ()
{
    cout << "This method is defined inside the class\n";
};
}

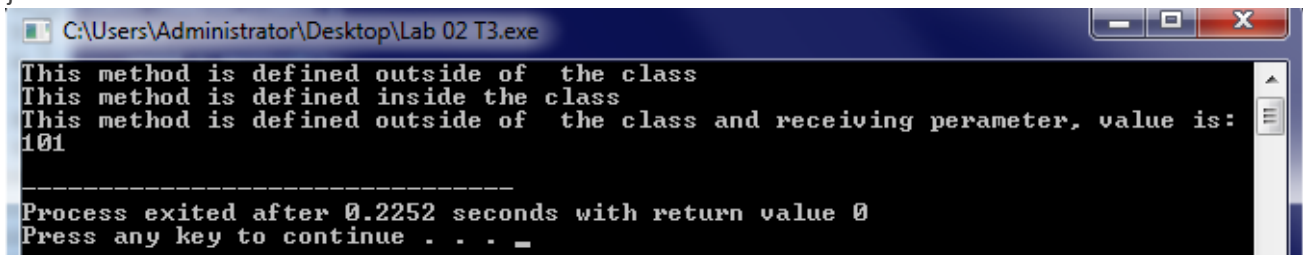
```

```

void Sample Class: :o_method ()
{
    cout << "This method is defined outside of the class\n";
}
void Sample Class: :o_method2(int number)
{
    cout << "This method is defined outside of the class and receiving parameter, value is: " <<
number << "\n";
}
int main ()
{Sample Class obj;
    obj.o_method ();
    obj.i_method ();
    obj.o_method2(101);

    return 0;
}

```



```

C:\Users\Administrator\Desktop\Lab 02 T3.exe
This method is defined outside of the class
This method is defined inside the class
This method is defined outside of the class and receiving parameter, value is:
101
-----
Process exited after 0.2252 seconds with return value 0
Press any key to continue . . . _

```

Code#4

```

#include <iostream>
using namespace std;

class employee
{
    public:
    int E_id;
    string E_name;
    float E_basic;
    float E_da;
    float E_it;
    float E_net_sel;

    public:
float find_net_salary(float basic, float da, float it);
    void show_emp_details();};

float employee :: find_net_salary(float basic, float da, float it)
{
    return (basic+da)-it;
}

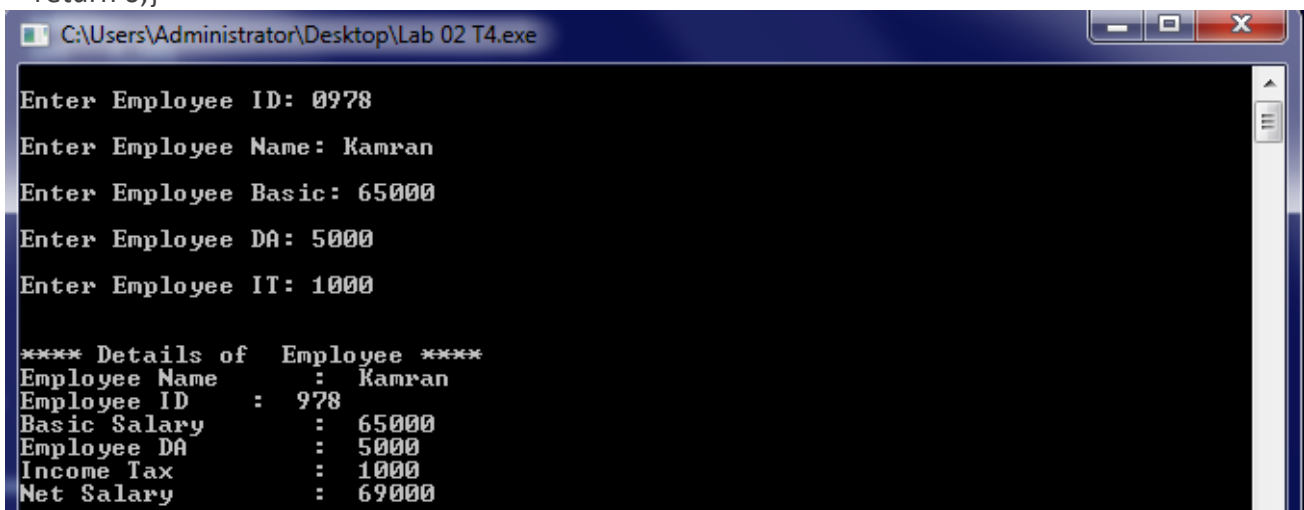
```

```

void employee :: show_emp_details()
{
    cout<<"\n\n**** Details of Employee ****";
    cout<<"\nEmployee Name   : "<<E_name;
    cout<<"\nEmployee ID    : "<<E_id;
    cout<<"\nBasic Salary    : "<<E_basic;
    cout<<"\nEmployee DA     : "<<E_da;
    cout<<"\nIncome Tax      : "<<E_it;
    float net_salary=find_net_salary(E_basic, E_da, E_it);
    cout<<"\nNet Salary     : "<<net_salary;
    cout<<"\n-----\n\n";}

int main()
{
    employee emp;
    cout<<"\nEnter Employee ID: ";
    cin>>emp.E_id;
    cout<<"\nEnter Employee Name: ";
    cin>>emp.E_name;
    cout<<"\nEnter Employee Basic: ";
    cin>>emp.E_basic;
    cout<<"\nEnter Employee DA: ";
    cin>>emp.E_da;
    cout<<"\nEnter Employee IT: ";
    cin>>emp.E_it;
    emp.show_emp_details();
    return 0;}

```



```

C:\Users\Administrator\Desktop\Lab 02 T4.exe
Enter Employee ID: 0978
Enter Employee Name: Kamran
Enter Employee Basic: 65000
Enter Employee DA: 5000
Enter Employee IT: 1000

**** Details of Employee ****
Employee Name   : Kamran
Employee ID    : 978
Basic Salary    : 65000
Employee DA     : 5000
Income Tax      : 1000
Net Salary     : 69000

```

Code#5

```

#include<iostream>
using namespace std;

```

```

class Bank

```



```

{
public:
string name;
string account_type;
int account_number;
int balance;

//member functions of the class Bank
// initialize function to initialize data members
void initialize(){
cout<<"\nEnter Account Holders Name:";
cin>>name;
cout<<"\nEnter Account type:";
cin>>account_type;
cout<<"\nEnter account number:";
cin>>account_number;
cout<<"\nEnter balance to deposit:";
cin>>balance;}
void deposit(){
int bal;
cout<<"\nEnter the amout to deposit:";
cin>>bal;
balance+=bal;
cout<<"\nAmount deposited successfully\nYour New Balance:"<<balance;}

//check() function to withdraw amount and check remaining balance
void check(){
int bal;
cout<<"\nYour balance :"<<balance<<"\nEnter amount to withdraw:";
cin>>bal;
if(bal<=balance){
balance-=bal;
cout<<"\nRemaining Balance:"<<balance;}
else{
exit(0);}}
//display function to display user information
void display(){
cout<<"\nName :";
cin>>name;
cout<<"\nBalance :"<<balance;}};
int main(){
int i;

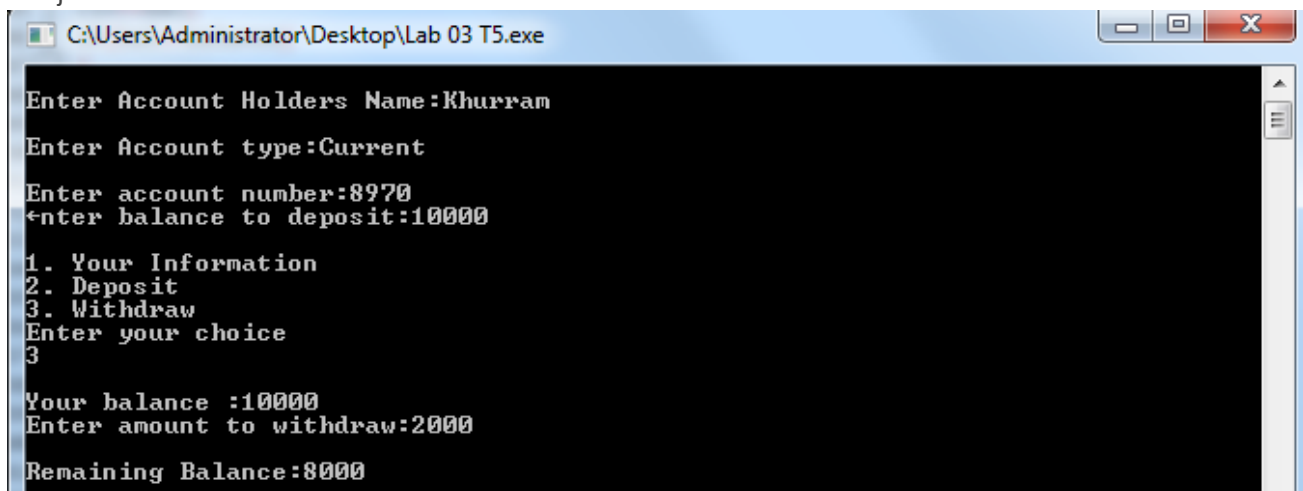
Bank bk;
bk.initialize();
cout<<"\n1. Your Information\n2. Deposit\n3. Withdraw\nEnter your choice\n";
cin>>i;
if(i==1)

```

```

{
    bk.display();
}
else if(i==2)
{
    bk.deposit();
}
else if(i==3)
{
    bk.check();
}
return 0;
}

```



LAB TASKS:

Task - 01:

Create a class User with two public fields: int Age and string Name. In the Main method, create an object of class User and set Name to "Teo" and Age to 24. Then, output to the screen: "My name is {Name} and I'm {Age} years old." using object fields for Name and Age.

Task - 02:

Create a class called Date that includes three pieces of information as instance variables—a month (typeint), a day (typeint) and a year (typeint). Provide a method displayDate that displays the month, day and year separated by forward slashes(/). Write a test application named DateTest that demonstrates classDate's capabilities.

Task - 03:

We are prototyping a robot that refills glasses during dinner. Every glass holds 200 milliliters. During dinner, people either drink water or juice, and as soon as there is less than 100 ml left in the glass, the robot refills it back to 200 ml.

Create a class Glass with one public int field LiquidLevel and methods public Drink(int milliliters) that takes the amount of liquid that a person drank and public Refill() that refills the glass to be 200 ml full. Both methods should not return any value. Initially set

LiquidLevel to 200. In the Main method create an object of class Glass and read commands from the screen until the user terminates the program (see next). Don't forget to refill the glass when needed!

Task - 04:

Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary (double). If the monthly salary is not positive, set it to 0.0. Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10% raise and display each Employee's yearly salary again.

Task - 05:

- . Create a class called Book to represent a book. A Book should include four pieces of information as instance variables—a book name, an ISBN number, an author name and a publisher. Provide methods (query method) for each instance variable. In addition, provide a method named getBookInfo that returns the description of the book as a String (the description should include all the information about the book). You should use this keyword in member methods and constructor. Write a test application named BookTest to create an array of object for 5 elements for class Book to demonstrate the class Book's capabilities.