NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

CL 1004 – Object Oriented Programing Lab

BCY-1A

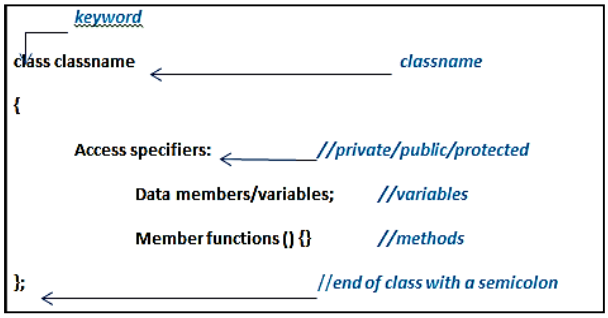
Lab 03

Outline

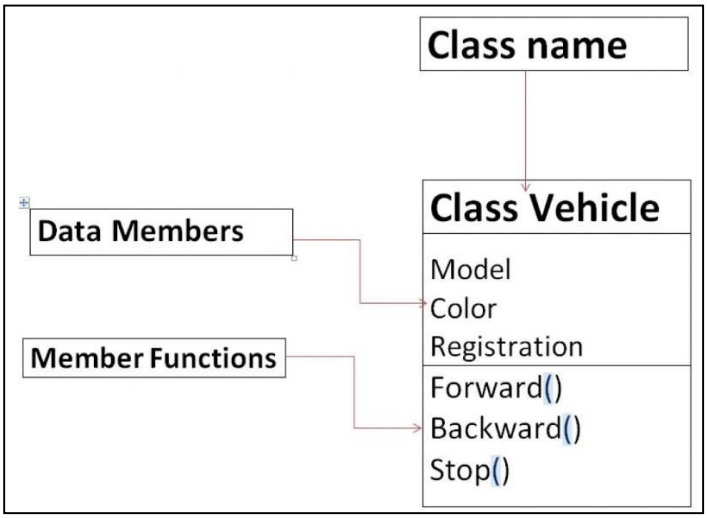
- Classes
- Objects
- Structures VS Classes
- Transformation from Procedural to Object Oriented Programming
- Example Programs
- Exercise

- A class in C++ can be viewed as a blueprint or a skeleton of a particular entity. Class is a user-defined data type. It contains the general information or data for that particular entity and the functions that operate on that entity.
- In C++ syntax, we define a class with keyword "class" followed by the name of the class.
- The class name is followed by the details of the class enclosed in curly braces and is terminated by a semicolon.

Syntax of Class



Example-Class



Specification of Class

- ***Class****: Keyword*
- ***Class-name****: User define*
- ***Access Specifiers****: Public, Private, Protected*
- ***Data members****: Data members are the data variables*
- ***Member Function****: Member functions define the operations on data members.*

Objects

- In order to use the class functionality, we need to instantiate the class to create an object. An object is an instance of a class. I simple words, we can say that an object is a variable of type class.

Classname object name;

- Once the object is created, it can be used to access the data members and functions of that class.

Summarize

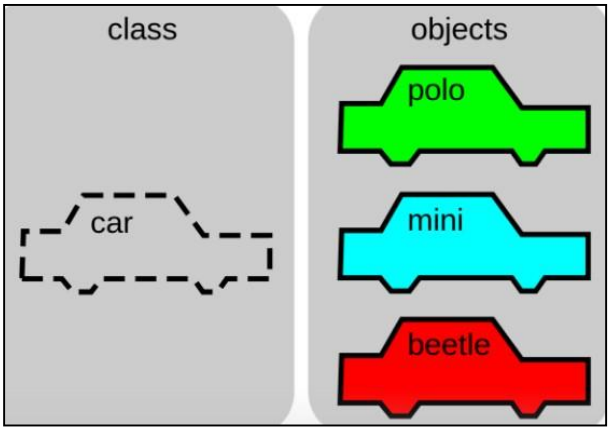
A class:

- 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



Object v/s Classes

BASIS FOR COMPARISON	OBJECT	CLASS
Definition	An instance of a class is known as Object.	A template or blueprint with which objects are created is known as Class.
Type of entity	Physical	Logical
Creation	Object is invoked by new keyword.	Class is declared by using class keyword.
Memory allocation	Creation of object consumes memory.	The formation of a class doesn't allocate memory.

Syntax of Class and Object

```cpp
#include <bits/stdc++.h>
using namespace std;
class OppClass
{
    // Access specifier
    public:

    // Data Members
    string name;

    // Member Functions()
    void showname()
    {
        cout << "your name  is: " << name;
    }
};
```

```cpp
int main() {

    // Declare an object of class OppClass
    OppClass obj1;

    // accessing data member
    obj1.name = "Ali";

    // accessing member function
    obj1.showname();
    return 0;
}
```

Program as Exampele

- Write Code that take input from user student-id , student-age , student-name and print all information.
    - Hint: Two method will be declare

```cpp
#include <iostream>
#include <string>
using namespace std;


        int student_id;
        string student_name;
        int student_age;

class StudentInfoClass
{
        public:
        void student_info_input()
        {
                cout<<"enter student id"; cin>>student_id;
                cout<<"enter student name"; cin>>student_name;

                cout<<"enter student age"; cin>>student_age;

        }
```

```cpp
        void student_info_print()
        {
student_info_input();
                cout<<"student id"<<student_id<<endl;
                cout<<"student name"<<student_name<<endl;
                cout<<"student age"<<student_age<<endl;
        }
};

int main()
        {
                StudentInfoClass obj1; //object declare

        //obj1.student_info_input();

        obj1.student_info_print();
                return 0;
        }
```

Multiple Objects

- You can create multiple objects of one class.
- The benefit of multiple instances of a single class is that each of them can hold different values to their variables, and the methods only affect that specific object, so each of them have the same skeleton, but are in independent states.

- An example of this is an enemy class in a game. If you have multiple enemies, you can create multiple instances of a class, where each enemy is an instance. They all may be in a different position, which could be a private variable of the class.

Multiple Objects-Example

```
#include <iostream>
#include <string>
using namespace std;
// class definition
// "student" is a class
class employee
        {
                public: // Access specifier
                        int id;
                        string name;
                        string place;
                //      static int
objectCount;

                        /*employee()
                                {

        objectCount++;
                                }*/
                        void
EmployeeDetails()
                                {

        cout << "Enter Id: "<<endl;
                                cin >> id;

        cout << "Enter name: "<<endl;
                                cin >>
name;
                                    cout <<
"Enter place: "<<endl;
                                    cin >>
place;
                                }
                                void printInfor()
                                        {

                cout << "ID: " << id << "\n";

                cout << "Name: " << name << "\n";

                cout << "Place: " << place << "\n";

                cout<< "\n";

                                        }

        };
//int employee::objectCount = 0;
int main()
        {
                // multiple object creation
                employee emp1, emp2;
                // Accessing attributes and
setting the values
                emp1.EmployeeDetails();
                emp1.printInfor();
                emp2.EmployeeDetails();
                emp2.printInfor();
                // cout << "Total
employee created = " <<
employee::objectCount << endl;
                return 0;
        }
```

Structure vs Classes

- In C++, a structure is the same as a class except for a few differences.
- The most important of them is security.
- A Structure is not secure and cannot hide its implementation details from the end user while a class is secure and can hide its programming and designing details.
    - Following are the points that expound on this difference:
    - What if we forget to put an access modifier before the first field?

1) Data Members of a class are private by default and members of a struct are public by default.

```
struct Robot {    OR    class Robot {
  float locX;             float locX;
```

Structure vs Classes

Class has limitless features.	Struct has limited features.
All the reference types are allocated on heap memory.	All the value types are allocated on stack memory.
Class is generally used in large programs.	Struct are used in small programs.
A Class can inherit from another class.	A Struct is not allowed to inherit from another struct or class.

Transformation From Procedural To Object Oriented Programming-Example

```cpp
#include <iostream>
using namespace std;
//Procedural way
double calculateBMI(double w, double h){
    return w/(h*h)*703;
    }
string findstatus(double bmi){
string status;
if(bmi < 18.5)
    status = "underweight";
else if(bmi < 25.0)
    status = "normal";
return status;
}
int main(){
    double bmi,weight,height;
    string status;
    cout<<"Enter weight in pounds";
    cin>>weight;
    cout<<"Enter height in inches";
    cin>>height;
    bmi = calculateBMI(weight,height);
    cout<<"your BMI is "<<bmi<<" your status is "<<findstatus(bmi);
}
```

Procedural way

```cpp
#include <iostream>
using namespace std;
class BMI{
    double height,width,bmi;
    string status;
    public:
    void getInput(){
        cout<<"Enter width in pounds: ";
        cin>>width;
        cout<<"Enter height in inches: ";
        cin>>height;}
    double calculateBmi(){
        return width/(height*height)*703;}
    string displayStatus(){
        if(bmi < 18.5)
            status = "underwigth";
        else if(bmi > 25.0)
            status = "normal";
        return status;
    }
    void printOutput(){
        bmi = calculateBmi();
        cout << "your BMI IS "<<bmi<< " bmi status is "<<displayStatus();
    }
};
int main(){
    BMI ob1;
    ob1.getInput();
    ob1.printOutput(); }
```

OOP way

Getter/Setter Functions

- Getters and setters are as the name suggests functions that are created to set values and to fetch i.e. get values.
- These vastly work with classes a lot and are famous ways of doing the same in the coding paradigm