Name of Teacher: Miss Radhika M. Patil

Class: B.Sc. Computer Science (Entire)- II **Semester : 3**

Course Title: Object Oriented Programming using C++

<u>**Classes and Objects:**</u>

1. Class:

- Class is model for creating object. This means that properties and function of object are written inside the class.
- Class is a way to <u>**bind the data and its associated functions together.**</u>
- Class is <u>**Abstract Data Type (ADT).**</u> Or User Defined Data type.
- When defining class we are creating new ADT that can be treated like any other built in data type.
- Class is a <u>**blueprint**</u> of an object.

General Syntax of class declaration:

Syntax:

```
class  class_name
{
   private:
              variable_declaration;
              function_declaration;
                                                  // Class Members
   public:
               variable_declaration;
              function_declaration;
};
```

- The class body contains declaration of variable and function .
- The variables or data declared inside the class are called members of a class.
- The function declared inside the class are called are called member functions.
- The variables and functions collectively called as class members.
- They are usually grouped under two sections private and public to denote which of them are private or public.
- Class members that have been declared as private can be accessed only from within the class and public members can be accessed from outside the class also.
- Use of keyword private is optional, **by default members of class are private.**

Example:

```
class Student
{
                int roll_no;
                char sname[20];
                int age;
               char cls[10];
    public:
                void input();
                void display();
};
```

2. Object:

- Object is basic **run time entity** in Object Oriented Programming (OOP).
- Objects are also called as **instances of a class.**
- Object has certain **properties and functions** associated with it.
- A class provide blueprint for objects. So basically object is created from class. We declare the objects of a class with exactly the same sort of declaration that we declare the variable of basic data type.
- An object is a block of memory that contains the scope to share all the data members or instance variables.
- To use a class we should create an object of that class.
- Object creation represents allocating memory necessary to share actual data of variable.
- To create an object following syntax is used

Syntax:

```
class_name obj_name;
```

Example:

```
 Student S;
```

Here, Student is a class name and S is object name. S is actually a variable of type Student. i.e. Student is type of variable S.

Accessing the class members:

1. Each object created has own set of variable.
2. To assign value to a class variable, we must use corresponding object and dot operator (.) i.e. the dot operator along with obj name is used to access both instance variable (data members) and methods (functions) within class.
3. To access class members following syntax is used.

```
obj_name.var_name=value.        //to assign value to data member
obj_name.fun_name();            //to call the function of a class.
```

Example:

```
 Student S1;
  S1.roll_no=9134;
   S1.display();
```

Example: Write a program to define a class ‘Demo’ and input value and display it.

```
class Demo
{
                int a;
     public:
                void display()
                {
                        cout<<“\n a= ” <<a;
                 }
};

void main()
{
         Demo d;
          d.a=100;
          d.display();
          getch();
}
```

Output:

```
a=100
```

Assignment: Write an Object Oriented Program (OOP) to define a class ‘Student’ , input student details (roll no, name, class) and display it.

Solve this assignment and attach the screenshot of program in pdf file along with the notes.

Access Specifiers/Access Modifiers/Visibility Modes:

- Access modifiers or specifiers define the scope of variable as well as methods.
- Scope means whether variables or methods defined in a class can be accessed by all the classes of a project or the same class only or if the few other classes which is inherited.
- Access modifiers are used to set boundries foe availability of members of a class.
- Access specifiers in program are followed by colon (:). You can use either 1 or 2 all the three specifiers in same class to set different boundaries for different members.

C++ defines 3 access Modifiers:

1. public
2. private
3. protected

1. public:

All class members declared as public will be available to everyone. Data members and member functions can be accessed outside the class also. Hence, there are chances that they might change them. So, the key members must not be declared as public.

Example:

```
class PublicAccess
{
    public:
            int x;                  // data member declaration
            void display();         // member function declaration
};
```

2. private:

Class members declared as private can't be accessed outside the class. If someone tries to access private members, they will get compile time error.

By default data members and member functions of class are **private**.

Example:

```
class PrivateAccess
{
    private:
            int x;                  // data member declaration
            void display();         // member function declaration
};
```

3. protected:

It is similar to private. i.e. it makes class members inaccessible outside the class. But they can be accessed by **immediate derived class** of that class.
e.g: If class B is created from class A then class B is immediate derived class of A.

Example:

```
class ProtectedAccess
{
   protected:
          int x;                   // data member declaration
          void display();          // member function declaration
};
```

Consider the program that demonstrates use of access modifiers in C++

Program:

```
class AccessDemo
{
   private:
            int a;
   public:
           int b;
           void show()
           {
               a=100;
               cout<<"\n Private member is:"<<a;
           }
};
```

```
void main()
{
     AccessDemo d;     //create object of class AccessDemo
     d.b=200;
     cout<<"\n Public member is:"<<d.b;
      d.show();
     getch();
}
```

Output:

```
Public member is:200

Private member is:100
```

Defining Member Function:

- ➢ Member functions are those functions which are declared inside the class definition and work on data members of a class.
- ➢ Member function can be defined in 2 ways:

1. Inside the class definition.
2. Outside the class definition.

1. Inside the class definition.

The member function can be defined inside the class. If you define the function inside the class then there is no need to declare the function. You can directly define the function inside the class definition.

Example: Write an OOP to add two numbers defining member function input() and show() inside the class name Sum and display the result.

Program:

```
class Sum
{
                int a,b;
    public:
                void input()
                {
                     cout<<“\n Enter a and b:”;
                     cin>>a>>b;
                }

               void show()
               {
                     int s=a+b;
                     cout<<“\n Sum=” <<s;
               }
};
```

```
void main()
{
          Sum s1;
           s1.input();
           s1.show();
            getch();
 }
```

Output:

```
Enter a and b: 19 76
Sum=95
```

2. Outside the class definition:

1. Member function of a class can be defined outside the class definition also. The functions are only declared inside the class but defined outside the class.
2. The general form member function definition outside the class definition is as follows:

```
return_type  class_name :: function_name()
{
    //statements;
}
```

Where :: is a scope-resolution operator. Only scope resolution operator identify the function as a member of a particular class.

Example:
Write an OOP to add two numbers defining member functions input() and show() outside the class named Sum and display the result.

```
class Sum
{
            int a,b;
    public:
            void input();
            void show();
};
```

```
void  Sum:: input()
{
      cout<<"\n Enter a and b:";
      cin>>a>>b;
}

void  Sum:: show()
{
            int s=a+b;
            cout<<"\n Sum="<<s;
}
```

```
void  main()
{
      Sum s;
       s.input();
       s.show();
       getch();
}
```

THANK YOU…