# Lab 1 Instructor: Saroj Shakya

# Class, Methods and Messages

A class is **an abstract data type** that **combines related data and functions together**, which can be used to create objects (*instances/variables*) of this type.

Class descriptions begin with the key word **class**

C++ provides the key words **private** and **public** which may be used in class declarations.

These key words are known as **visibility modes** or **access specifiers** because they specify how class members may be accessed.

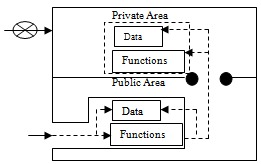
The keyword **private** precedes those portions of code that can be accessed only by methods in the class itself

The keyword **public** indicates the true interface - those elements that are accessible from outside the class

**Syntax:**

|  |  |
| --- | --- |
| class name\_of\_class{  private: //visibility Mode  variable declaration;//Member Data  function declaration; //Member Function  public: //visibility Mode  variable declaration;  function declaration;  protected:  variable declaration;  function declaration;  }; | For example, definition of class Student:class Student { private:  char name[10];  int roll;  public:  void init\_data();  void display\_data();  }; |

Class



**Entry Allowed to Public Area**

**No Entry to Private Area**

Defining Member Functions

Member Functions can be defined inside the class definition or outside the class definition.

A member function of a class is defined inside a class like a normal function and it is treated like an inline function

If a method is defined outside the body of a class, a scope resolution operator :: is used

**Syntax:**

return type class-name :: function-name (argument list)

{

}

For example, the following is the definition of the member function display\_data() outside the class definition of the class Student:

void Student :: display\_data(){

cout << “ Name:” << name;

cout << “\n Roll:” << roll;

}

Defining an Instance/Object of a Class

Class is used to create objects called **instances** of the class.

Objects are created with simple definition statements, just like regular variables int age, float price.

**Syntax:**

className objectName;

For example, the following statement defines s1 as an object of the Student class:

Student s1;

Defining a class object is called the**instantiation** of a class. In this statement, s1 is an **instance**of the Student class.

Message Passing Syntax in C++/Accessing Class Members

In C++ a method is called **a member function** and **passing a message** to an object is referred to as **invoking a function**

The syntax used to invoke a member function is **similar to that used to access data members**

The notation states the *receiver*, followed by a *dot/period* then the *message selector* (*the name of a member in the class of the receiver*) and finally the *arguments* inside the braces.

**Syntax:**

receiverObject.message(arguments);

For example, the following statement is used to request the student object s1 to initialize and display data

s1.init\_data(); //message passing

s1.display\_data();//message passing

Complete Example

class Student {

private:

char name[10];

int roll;

public:

**//definition of member function inside the class, inline function**

void init\_data(){

cout << ”Enter Name and Roll”;

cin >> name >> roll;

}

**//declaration of member function inside class but definition is outside of class**

void display\_data(); //

};

**//definition of member function outside the class**

void Student :: display\_data(){

cout << “ Name:” << name;

cout << “\n Roll:” << roll;

}

void main(){

Student s1; **//object s1 of class Student created/instantiated**

s1.init\_data(); **//passing of message init\_data() to object s1**

s1.display\_data();**//passing of message display\_data() to object s1**

}

Exercises

**1. Circle Class**

Write a Circle class that has the following member variables:

radius: a double

pi : a double initialized with the value 3.14159

The class should have the following member functions:

**• setRadius**. A mutator function for setting the radius variable.

**• getRadius**. An accessor function for getting the radius variable.

**• getArea**. Returns the area of the circle, which is calculated as area = pi \* radius \* radius

**• getDiameter**. Returns the diameter of the circle, which is calculated as diameter = radius \* 2

**• getCircumference**. Returns the circumference of the circle, which is calculated as

circumference = 2 \* pi \* radius

Write a program that demonstrates the Circle class by asking the user for the circle’s radius, creating a Circle object, and then reporting the circle’s area, diameter, and circumference.

**2. 3D Point**

Create a class called Point that represents a 3D coordinate system. Create two objects of the Point class called P1 and P2. Each object of Point should have coordinates (x,y,z) and methods to assign coordinates to the point objects and display points in the form (x,y,z). Now add a method to calculate the distance from point P1 to P2.

**3. Date**

Design a class called Date. The class should store a date in three integers: month, day, and year. There should be member functions to print the date in the following forms:

12/25/10

December 25, 2010

25 December 2010

Demonstrate the class by writing a complete program implementing it.

*Input Validation: Do not accept values for the day greater than 31 or less than 1. Do not accept values for the month greater than 12 or less than 1.*

**4. Employee Class**

Write a class named Employee that has the following member variables:

**• name**. A string that holds the employee’s name.

**• idNumber**. An int variable that holds the employee’s ID number.

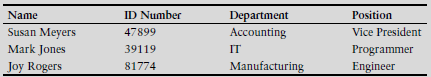
**• department**. A string that holds the name of the department where the employee works.

**• position**. A string that holds the employee’s job title.

The class should have the following functions:

**•** A function that accepts the following values as arguments and assigns them to the appropriate member variables: employee’s name, employee’s ID number, department, and position.

Write appropriate accessor functions that return the values in these member variables. Once you have written the class, write a separate program that creates three Employee objects to hold the following data.



The program should store this data in the three objects and then display the data for each employee on the screen.

**5. Car Class**

Write a class named Car that has the following member variables:

**• yearModel.** An int that holds the car’s year model.

**• make.** A string that holds the make of the car.

**• speed.** An int that holds the car’s current speed.

In addition, the class should have the following constructor and other member functions.

**•**The member function should accept the car’s year model and make as arguments. These values should be assigned to the object’s yearModel and make member variables and assign 0 to the speed member variables.

**• Accessor.** Appropriate accessor functions to get the values stored in an object’s yearModel, make, and speed member variables.

**• accelerate.** The accelerate function should add 5 to the speed member variable each time it is called.

**• brake.** The brake function should subtract 5 from the speed member variable each time it is called.

Demonstrate the class in a program that creates a Car object, and then calls the accelerate function five times. After each call to the accelerate function, get the current speed of the car and display it. Then, call the brake function five times. After each call to the brake function, get the current speed of the car and display it.

**6. Rational Numbers**

Write a program to implement a class called Rational with the following integer member variables in the private section of the class:

num

den

and the following member functions in the public section

void assign(int , int); //to assign numerators and denominators

double convert(); //to return double value of the rational number

void print(); //to print rational number as num/den

void invert(); //to invert numerator and denominator

. The program should

* Define a Rational object called x
* Assign 22/7 to x
* Print the objects value as a rational number (22/7)
* Print the same rational number as a real number
* Invert x such that x is 1/x

Check if you can print num and den of x from main. Discuss the reason behind your observation

**Note:**

All students are required to submit the lab report in proper format including class diagrams, source code along with comments, screen shots of outputs etc by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.