CL-1004 Object Oriented Programming

LAB - 03

Classes Objects
Structures VS Classes
Transformation from Procedural to Object Oriented Programming

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
Spring 2024

Classes

A class is a programmer-defined data type that describes what an object of the class will look like when it is created.
It consists of a set of variables and a set of functions.

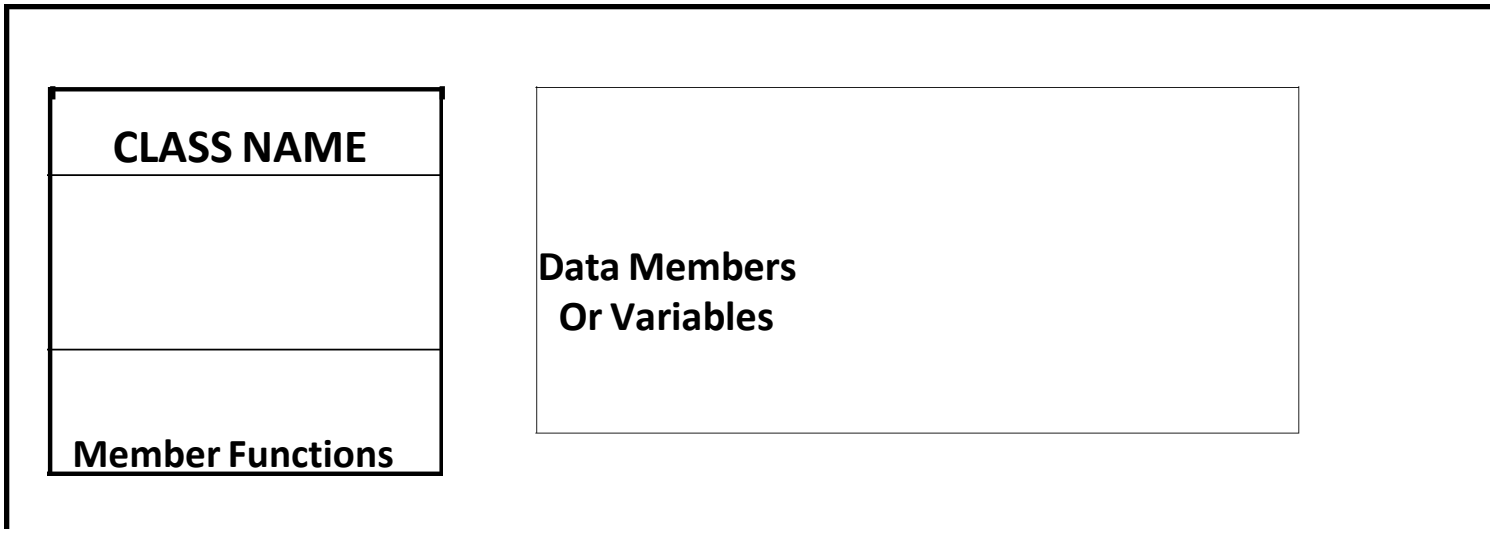
We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

As, many houses can be made from the same description, we can create many objects from a class.

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 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:

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

These attributes form the data in our program. The values that these attributes take (the blue color of the petals, for example) form the state of the object.

MEMBER FUNCTIONS

Consider the operations of some real world objects:

RADIO – setting its station and volume (invoked by the person adjusting the radio's controls)
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.

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
```

RADIO r;

CAR c;

Accessing Public Data Members

The public data members of objects of a class can be accessed using the direct member access operator (.).
However the private data members are not allowed to be accessed directly by the object. Accessing a data member depends solely on the access control of that data member.

Accessing Private Data Members

To access, use and initialize the private data member you need to create getter and setter functions, to get and set the value of the data member.
The setter function will set the value passed as argument to the private data member, and the getter

function will return the value of the private data member to be used. Both getter and setter function must be defined public.

C++ > oop.cpp > main()

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

class Student
{
    private:    // private data member
    int rollno;

    public:
    // public function to get value of rollno - getter
    int getRollno()
    {
        return rollno;
    }
    // public function to set value for rollno - setter
    void setRollno(int i)
    {
        rollno=i;
    }
};

int main()
{
    Student A;
    A.rollno=1;  //Compile time error
    cout<< A.rollno; //Compile time error

    A.setRollno(1);  //Rollno initialized to 1
    cout<< A.getRollno(); //Output will be 1
}

```

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

class Shape
{
   // as private so object.height and object.width is inaccessible
   int height, width;

   public:
   //setters
   void setHeight(int h){
      height = h;
   }

   void setWidth(int w){
      width = w;
   }

   //getters
   int getHeight(){
      return height;
   }

   int getWidth(){
      return width;
   }
};

int main()
{
   Shape shape;

   // setters
   shape.setHeight(5);
   shape.setWidth(2);

   // getters
   cout << "The Height is : " << shape.getHeight() << endl;
   cout << "The Width is : " << shape.getWidth() << endl;

   // shape.height or shape.width wont work as they are private

   return 0;
}
```

Output

```
The Height is : 5
The Width is : 2
```

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-processormacro, 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.

```
#include  <iostream>
using namespace std;
class car
{
  private:
    int car_number;
  char car_model[10];
  public:
    void getdata(); //function declaration
    void showdata();
};
//Outside class function definition
void car::getdata()
 {
   cout<<"Enter car number: "; cin>>car_number;
   cout<<"\n Enter car model: "; cin>>car_model;
 }
//Outside class function definition

void car::showdata()
 {
   cout<<"Car number is "<<car_number;
   cout<<"\n Car model is "<<car_model;
 }
// main function starts
int main()
 {
    car c1;
    c1.getdata();
    c1.showdata();
return 0;
}
```

Output

```
Enter car number : 9999
Enter car model : Sedan
Car number is 9999
Car model is Sedan
```

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.

TRANSFORMATION FROM PROCEDURAL TO OBJECT ORIENTED PROGRAMMING

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

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";// so on.
return status;
}

int main()
{
    double bmi, weight, height;
    string status;
    cout<<"Enter weight in Pounds ";
    cin>>weight;
    cout<<"Enther height in Inches ";
    cin>>height;
    bmi=calculateBMI(weight,height);
    cout<<"Your BMI is "<<bmi<<" Your status is "<<findStatus(bmi);

}
```

Procedural Approach

```
#include<iostream>
using namespace std;
class BMI
{
    double weight, height,bmi;
    string status;
    public:
        void getInput() {
            cout<<"Enter weight in Pounds ";
            cin>>weight;
            cout<<"Enther height in Inches ";
            cin>>height;   }
        double calculateBMI() {
             return weight / (height*height)*703; }
         string findStatus() {
            if(bmi < 18.5)
                status = "underweight";
            else if(bmi < 25.0)
                status = "normal";// so on.
            return status; }
        void printStatus()  {
            bmi = calculateBMI();
            cout<< "You BMI is "<< bmi<< "your status is " << findStatus(); }
};

int main()
{
    BMI bmi;
    bmi.getInput();
    bmi.printStatus();
}
```

OBJECT ORIENTED APPROACH
EXAMPLE PROGRAM


```
#include<iostream>
using namespace std;
class Account
{
private:
        double balance; // Account balance
public: //Public interface:
        string name; // Account holder long accountNumber;
        // Account number void setDetails(double bal)
        {
                balance = bal;
        }
        double getDetails()
        {
                return balance;
        }
        void displayDetails()
        {
                cout<<"Details are: "<<endl;
           cout<<"Account Holder:
           "<<name<<endl;
                cout<<"Account Number:
                "<<
           accountNumber <<endl; cout<<"Account
            Balance: "<<getDetails()<<endl;
        }
  };

int main()
{ double accBal;
Account
currentAccount;
currentAccount.getDetails();

cout<<"Please enter the details"<<endl;
cout<<"Enter Name:"<<endl; getline(cin,
currentAccount.name); cout<<"Enter
Account Number:"<<endl;
cin>>currentAccount.accountNumber;

cout<<"Enter Account
Balance:"<<endl; cin>>accBal;
currentAccount.setDetails(accBal);
```

Set and get functions to manipulate Private data member

Publically available data: Assigning values from

Private data:

```
cout<<endl;
currentAccount.displayDetails();
return 0;
                }
```

Accessing private data using member function

Exercise Lab 03

Question # 01:

Create a class called water bottle.
The water bottle has a company (made by), color and water capacity. The water capacity is stored in both liters(*l*) and milliliters(*ml*).
Create variables and methods for your class. Methods should include getters and setters.
Also create an additional method that updates the water capacity (both in *l* and *ml*) after asking the user how much water a person has drank. Assume that the user always enters the amount in *ml*.
Demonstrate the functionality of the water bottle in your main method.

Question # 02:

You are tasked with implementing a School Management System in C++. The system should be able to manage information about students, teachers, and courses. Each student has attributes such as student ID, name, age, and grade. Teachers have attributes like teacher ID, name, and subject taught. Courses are identified by a course code, name, and the teacher who conducts the course.
Implement the following classes:

1. **Student** class with the following attributes:
    - Student ID (integer)
    - Name (string)
    - Age (integer)
    - Grade (char)

   Include setter and getter functions for each attribute. Implement a function **displayStudentInfo()** to display the student's information.
2. **Teacher** class with the following attributes:
    - Teacher ID (integer)
    - Name (string)
    - Subject Taught (string)

   Include setter and getter functions for each attribute. Implement a function **displayTeacherInfo()** to display the teacher's information.
3. **Course** class with the following attributes:
    - Course Code (string)
    - Course Name (string)
    - Teacher (an instance of the **Teacher** class)

   Include setter and getter functions for each attribute. Implement a function **displayCourseInfo()** to display the course information.
4. **SchoolManagementSystem** class to manage students, teachers, and courses. Include the following functions:
    - **addStudent(const Student& newStudent)**: Adds a new student to the system.
    - **addTeacher(const Teacher& newTeacher)**: Adds a new teacher to the system.
    - **addCourse(const Course& newCourse)**: Adds a new course to the system.
    - **displayAllStudents()**: Displays information for all students in the system.
    - **displayAllTeachers()**: Displays information for all teachers in the system.
    - **displayAllCourses()**: Displays information for all courses in the system.

   Implement the solution in C++ using setter and getter functions for each class.

Question # 03:

Create a class called Employee that includes three pieces of information as data members a first name (type string), a last name (type string) and a monthly salary (type int). Provide a set and a get function for each data member. If the monthly salary is not positive, set it to 0. Write a test program that demonstrates class Employee’s capabilities.
Create two Employee objects and display each object’s yearly salary. Then give each Employee a 10

percent raise and display each Employee's yearly salary again.

Question # 04:

Define a class Car with private attributes model, year, and price. Provide setter and getter functions for each attribute.

Question # 05:

Develop a class Temperature with a private attribute celsius. Include setter and getter functions for the temperature in Celsius and a method to convert it to Fahrenheit.