National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Spring 2023, Lab Manual - 03

Course Code: CL-1004	**Course : Object Oriented Programming Lab**
Instructor(s) :	**Shahroz Bakht**

Lab # 03

Outline

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

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.

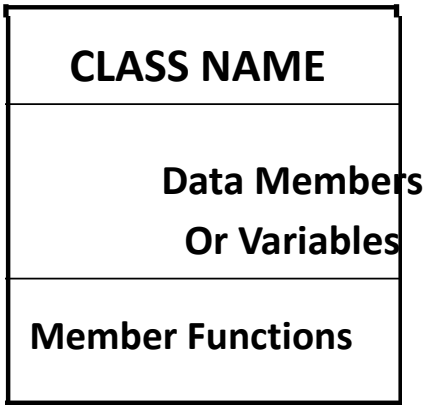
As, many houses can be made from the same description, we can create many objects from a class.

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 class-name
{
access-specifier:
data

access-specifier: functions
};
```

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
}
```

Getter & Setter , this Keyword in C++ Programming

The getter function is used to retrieve the variable value and the setter function is used to set the variable value. They this is a keyword that refers to the current instance of the class. They are getters and setters the standard way to provide access to data in Java classes. Setters and Getters allow for an object to contain private variables which can be accessed and changed with restrictions.

CL1004 - Object Oriented Programming Lab 2023

```
#include<iostream>
using namespace std;
class student{
private:
    string name;
    int age;
public:
    student(string n,int a) {
        this->setName(n);
         this->setAge(a);  }
    string getName() {
        return this->name; }
    void setName(string n) {
        this->name=n; }
 int getAge()  {
        return this->age;  }
    void setAge(int a)  {
        this->age=a; }
    void printDetails(){
        cout<<"Name : "<<name<<endl;
        cout<<"Age  : "<<age<<endl; }};

int main(){
    student o("Ali",25);
    o.printDetails();
    o.setName("M.Ali");
     o.printDetails();
     cout<<o.getName()<<endl;
    return 0;}
```

Output

```
Name : Ali
Age  : 25
Name : M.Ali
Age  : 25
```

M.Ali

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()

```cpp
#include <iostream>
using namespace std;
class Student
{
    public:
    string StudentName;
    int id;

    // printname is not defined inside class definition
    void printname();

    // printid is defined inside class definition
    void printid()
    {
        cout << " Student id is: " << id;
    }
};

// Definition of printname using scope resolution operator ::
void Student::printname()
{
    cout << " Student name is: " << StudentName;
}
int main() {

    Student obj1;
    obj1. StudentName = "xyz";
    obj1.id=15;

    // call printname()
    obj1.printname();
    cout << endl;

    // call printid()
    obj1.printid();
    return 0;
}
```

```
C:\Users\Administrator\Documents\C++>cd "c:\Users\Administrator\Documents\C++\" && g++ oop.cpp -o oop && "c:\Users\Administrator\Documents\C++\"
 Student name is: xyz
 Student id is: 15
```

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);
cout<<endl;
currentAccount.displayDetails();
```

Set and get functions to manipulate private data member

Publically available data:
Assigning values from

Private data:
Accessing private data using member function

```
return 0;
        }
```

Lab Tasks

Task_01:

You have been assigned a task where you have to calculate and display the surface area of a cylinder. Following are the details:

The Surface Area of Cylinder = Curved Surface + Area of Circular bases
S.A. (in terms of π) = $2\pi r (h + r)$

Where, π (Pi) = 3.142 or = 22/7

r = Radius of the cylinder

h = Height of the cylinder

The values for radius and height need to be asked from the user and a function should print the result.

Task_02:

Construct a class named as "SalariedEmp", this class must achieve the outcome of displaying the defined salary of any employee and for this purpose it must contain appropriate functions.

1. There must be a function that inquires the current salary of a worker as well as the number of hours he/she works for in a day and pass them as arguments to that function, one more option should be present which asks the employee if he/she will work overtime that day and how many hours will he/she work overtime?

2. A function must be present that increments the salary of an employee as overtime allowance by 10% of his/her current salary if he/she works overtime (works more than 8 hours in a day).

3. A function must be present that increments the salary of a person by 5% of his/her current salary if he/she works below the minimum wage (the minimum wage is the average salary of all registered employees).

4. Run this program for at least 4 employees.

Task_03:

Your job is to manage the transactions of a customer that wishes to open a bank account. The bank should enable a customer to:

1). Provide their full name.
2). Let the customer choose their bank account number but the number should be restricted to 5 digits.
3). Let the customer choose between a saving or a current account.
4). Let the customer deposit any initial amount but the amount must be more than 10000PKR.

The procedural working of this application should be as follows:

1). A function that lets the customer deposit the initial amount.
2). When the customer wishes to withdraw an amount from their account, provide validations on the amount if it is enough to withdraw and also restrict the amount to 25000 PKR per withdrawal.

3). An initial function should initialize all the attributes with default values.
4). A function must be present that prints the name as well as the updated account balance after every transaction made.

Task_04:

You are required to calculate the electricity consumption bill of customers. The customers have been divided into two categories:

1). Safe
2). Unsafe

The customers that come under the category of safe are those that have been consuming electric units for the last 6 bills under 200 and not more than that and vice versa for the Unsafe ones.

Your job is to first determine whether the customer you are addressing is safe or unsafe (assumptions can be made here but logically prove them) and then calculate the bill under the following scheme.

Safe:
1) From unit 1 to 50, 5 PKR per unit
2) From unit 51 to 100, 7 PKR per unit
3) From unit 100 to 150, 9 PKR per unit
4) From unit 151 to 200, 11 PKR per unit

Unsafe:
1) From unit 1 to 100, 8 PKR per unit
2) From unit 101 to 200, 12 PKR per unit
3) From unit 201 to 300, 15 PKR per unit
4) Anything above 300 17 PKR per unit

Task_05:
Create a class named “PhoneNumbers”, the class should contain the following attributes:
1) STD
2) Number
You need to accept a valid number from at least 6 users and then rephrase the input number to a new changed number based on the following conditions:
1) The STD code should be added by 1 digit (Ex: 7 becomes 8)
2) Next you should reverse the first two digits of the numbers. (Ex: 823-38-985334 932-38-985334)
Print the changed numbers.