



**Bahria University, Islamabad**

**Department of Software Engineering**

**Object Oriented Programming Lab (Spring-2025)**

**Teacher: Engr. Muhammad Faisal Zia**

---

**Student: Ahtisham khan**

**Enrollment: 09-131242-014**

---

**Lab Journal: 9**

**Date: 05/03/2025**

---

Task No:	Task Wise Marks		Documentation Marks		Total Marks (20)
	Assigned	Obtained	Assigned	Obtained	
1	2		4		
2	2				
3	2				
4	5				
5	5				

**Comments:**

**Signature**

## Lab # 9

### Single Inheritance

#### Objective:

The objective of this lab is to understand the implementation of **inheritance** in object-oriented programming (OOP) using C++. By the end of this lab, you should be able to understand:

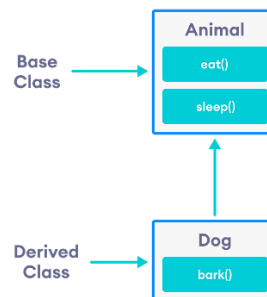
1. **Concept of inheritance** and its importance in OOP.
2. **Implement single inheritance** where a derived class inherits from a single base class.
3. **Use access specifiers (public, protected, private)** to control member accessibility in inheritance.
4. **Reuse base class methods** in derived classes to promote code efficiency.
5. **Design class hierarchies** to model real-world relationships between objects.

---

#### Introduction

One of the most important concepts in object-oriented programming is that of inheritance. *Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application.* This also provides an opportunity to reuse the code functionality and fast implementation time.

```
class Animal {  
    // eat() function  
    // sleep() function  
};  
  
class Dog : public Animal {  
    // bark() function  
};
```



When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the **base** class, and the new class is referred to as the **derived** class.

#### is-a relationship

The idea of inheritance implements the “**is a**” relationship. For example, mammal IS-A animal, dog IS-A mammal hence dog IS-A animal as well and so on.

#### Base & Derived Classes

A class can be derived from more than one class, which means it can inherit data and functions from multiple base classes(multiple inheritance). To define a derived class, we use a **class derivation list** to specify the base class(es). A class derivation list names one or more base classes and has the form:

A derived class inherits from a single base class using:

```
cpp                                                                    Copy
class DerivedClass : access-specifier BaseClass { ... };
```

Where access-specifier is one of **public**, **protected**, or **private**, and base-class is the name of a previously defined class. If the access-specifier is not used, then it is private by default.

#### Example 1:

Consider a base class Animal and its derived class Dog as follows:

```
// C++ program to demonstrate inheritance

#include <iostream>
using namespace std;

// base class
class Animal {

public:
    void eat() {
        cout << "I can eat!" << endl;
    }

    void sleep() {
        cout << "I can sleep!" << endl;
    }
};

// derived class
class Dog : public Animal {

public:
    void bark() {
        cout << "I can bark! Woof woof!!" << endl;
    }
};

int main() {
    // Create object of the Dog class
    Dog dog1;

    // Calling members of the base class
    dog1.eat();
    dog1.sleep();

    // Calling member of the derived class
    dog1.bark();

    return 0;
}
```

#### Example 2:

Consider a base class Shape and its derived class Rectangle as follows:

```
#include <iostream>
using namespace std;
// Base class
class Shape {
public:
    void setWidth(int w) {
```

```
        width = w;
    }

    void setHeight(int h) {
        height = h;
    }

protected:
    int width;
    int height;
};

// Derived class
class Rectangle : public Shape {
public:
    int getArea() {
        return (width * height);
    }
};

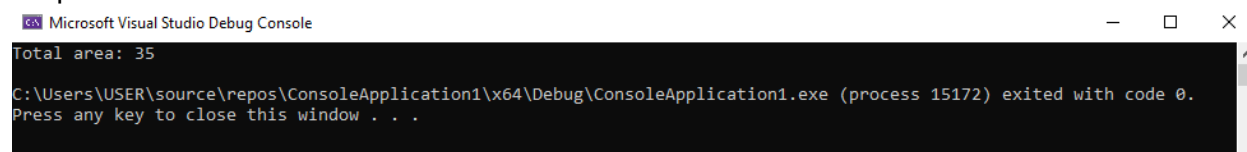
int main(void) {
    Rectangle Rect;

    Rect.setWidth(5);
    Rect.setHeight(7);

    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() << endl;

    return 0;
}
```

## Output



The screenshot shows the Microsoft Visual Studio Debug Console. The output is "Total area: 35". Below the output, it says "C:\Users\USER\source\repos\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe (process 15172) exited with code 0. Press any key to close this window . . .".

## Access Control and Inheritance

A derived class can access all the non-private members of its base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class.

We can summarize the different access types according to who can access them in the following way:

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

A derived class inherits all base class methods with the following exceptions:

- Constructors, destructors and copy constructors of the base class.
- Overloaded operators of the base class.
- The friend functions of the base class.

### Type of Inheritance based on Access

When deriving a class from a base class, the base class may be inherited through **public**, **protected** or **private** inheritance. The type of inheritance is specified by the access-specifier as explained above.

We hardly use **protected** or **private** inheritance, but **public** inheritance is commonly used. While using different types of inheritance, the following rules are applied:

- **Public Inheritance:**  
When deriving a class from a **public** base class, **public** members of the base class become **public** members of the derived class and **protected** members of the base class become **protected** members of the derived class. A base class's **private** members are never accessible directly from a derived class but can be accessed through calls to the **public** and **protected** members of the base class.
- **Protected Inheritance:**  
When deriving from a **protected** base class, **public** and **protected** members of the base class become **protected** members of the derived class.
- **Private Inheritance:**  
When deriving from a **private** base class, **public** and **protected** members of the base class become **private** members of the derived class.

### Types of Inheritance

- **Multilevel Inheritance**
- **Multiple Inheritance**
- **Hierarchical Inheritance**
- **Virtual Inheritance**

We will discuss the types in the upcoming lab.

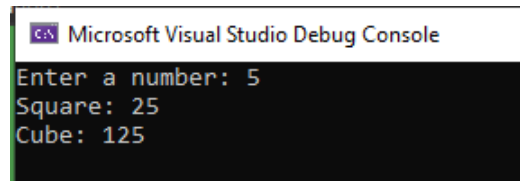
### Conclusion:

In this lab, we learned:

- **Single Inheritance** allows a derived class to extend a single base class.
  - **Code reusability** is improved by inheriting methods and attributes.
  - **Access specifiers** (public, protected, private) control inheritance visibility.
-

## Lab Tasks/ Homework

**Task 01:** Write a C++ program that demonstrates single inheritance by computing the square and cube of a given number. Create a base class Square containing a computeSquare() function to calculate the square of a number. Derive a class Cube from Square that implements a computeCube() function, which should utilize the base class's computeSquare() method to calculate the cube (by multiplying the square result with the original number). The output of program should be like this



```
Microsoft Visual Studio Debug Console
Enter a number: 5
Square: 25
Cube: 125
```

### Solution:

Enter your code here...

```
#include<iostream>
using namespace std;
class Square{

public:
    int square(int num);
};
class Cube :public Square {
public:
    int cube(int n);
};
int Square::square(int num) {
    return num * num;
}
int Cube::cube(int n) {
    int Square = square(n);
    return Square * n;
}
int main() {
    system("title Ahtisham khan (014) OOP LAB 9");
    Cube obj;
    int number;
    cout << "Enter the number: " << endl;
    cin >> number;
    cout << "Square: " << obj.square(number) << endl;;
    cout<<"Cube: "<<obj.cube(number)<<endl;
    system("pause");
}
```

### Output:

Your Output here...

```
C:\> Ahtisham khan (014) OOP LAB 9
Enter the number:
4
Square: 16
Cube: 64
Press any key to continue . . .
```

In case of output snippet please make sure output snippet contains student name and id. AliAhmed\_123\_Lab09\_T1.exe

**Task 02:** Create a base class Student that stores name, roll number and address (type string) of the student. Student class must have getdata () function to get its data (name, roll no and address of the student) from the user.

From this class derive another class Marks. This class must have inputmarks() function to get its data (marks in three subjects) from the user. Then calculate the total marks and average marks in the three subjects. Create another function show\_detail() to display the marks in three subjects, total marks and average marks. The output of the program should be like this:

```
C:\> Microsoft Visual Studio Debug Console
Enter name: Name
Enter roll number: 01
Enter address: St#65 Satt Town, Rawalpindi
Enter marks for 3 subjects:
Subject 1: 23
Subject 2: 24
Subject 3: 26

--- Student Detail ---
Name: Name
Roll No: 1
Address: St#65 Satt Town, Rawalpindi

--- Marks ---
Subject 1: 23
Subject 2: 24
Subject 3: 26
Total Marks: 73
Average Marks: 24.3333
```

### Solution:

Enter your code here...  
#include<iostream>

```
#include<string>
using namespace std;
class Students {
private:
    string name;
    int rollnumber;
    string address;
public:
    Students() {
        name = " ";
        rollnumber = 0;
        address = " ";
    }
    Students(string na, int roll, string add);
    void getdata();
    void display();
};
class Marks :public Students {
private:
    int total;
    int average;
    int marks[3];
public:
    Marks() {
        total = 0;
        average = 0;
    }
    Marks(int t, int a, int m) {
        total = t;
        average = a;
    }
    void inputmarks();
    void show_detalis();
};
Students::Students(string na, int roll, string add) {
    name = na;
    rollnumber = roll;
    address = add;
}
void Students::getdata() {
    //cin.ignore();
    cout << "Enter Your Name: " << endl;
    getline(cin,name);
    cout << "Enter Your Roll Number: " << endl;
    cin >> rollnumber;
    cin.ignore();
    cout << "Enter Your Address: " << endl;
    getline(cin, address);
}
```



```
        cout << endl;
    }
    void Marks::inputmarks() {

        cout << "Enter Marks of 3 Subjects: " << endl;
        for (int i = 0; i < 3; i++) {
            cout << "Marks of Subject " << i + 1 << ":" << endl;
            cin >> marks[i];
            total += marks[i];
        }
        average = total / 3;
    }
    void Marks::show_detalis() {
        cout << "\t***Marks***" << endl;
        cout << "Marks Of 3 subjects: " << endl;
        for (int i = 0; i < 3; i++) {

            cout << "Marks of subject " << i + 1 << ":" << marks[i] << endl;
        }
        cout << "Total Marks: " << total << endl;
        cout << "Average Marks: " << average << endl;
    }
    void Students::display() {
        cout << "\t**Student Detalis: " << endl;
        cout << "Name: " << name << endl;
        cout << "Roll Number: " << name << endl;
        cout << "Address: " << address << endl;
        cout << endl;
    }
    int main() {
        system("title Ahtisham khan (014) OOP LAB 9");
        Marks obj1;
        obj1.getdata();
        obj1.inputmarks();
        obj1.display();
        obj1.show_detalis();
        system("pause");
        return 0;
    }
}
```

**Output:**

Your Output here...

CS Ahtisham khan (014) OOP LAB 9

```
Enter Your Name:
Ahtisham khan
Enter Your Roll Number:
014
Enter Your Address:
Chak Shahzed House 12,street 10

Enter Marks of 3 Subjects:
Marks of Subject 1:
23
Marks of Subject 2:
45
Marks of Subject 3:
66

    **Student Detalis:
Name: Ahtisham khan
Roll Number: Ahtisham khan
Address: Chak Shahzed House 12,street 10

    ***Marks***
Marks Of 3 subjects:
Marks of subject 1:23
Marks of subject 2:45
Marks of subject 3:66
Total Marks: 134
Average Marks: 44
Press any key to continue . . . _
```

In case of output snippet please make sure output snippet contains student name and id. AliAhmed\_123\_Lab09\_T2.exe

**Task 03:** Create a base class Employee that stores name (a string) and identification number (type integer) of an employee. From this class derive three more classes:

**Manager:** that stores the title and salary (string) of the manager.

**Scientist:** that stores the number of articles they have published (type integer) and salary (a string) of the scientist.

**Clerks:** that stores the overtime (type int) of the clerk.

Each of the four classes should have an input () function to input data from the user, and show () function to display data. Write a main function to test the three classes by creating instances of them, asking the user to fill in data with input (), and then displaying data with show () function.

The output of the program should be like:

```
Enter the data for manager:
Enter name of the employee: Ali
Enter the identification number of the employee: 1
Enter Title: President
Enter the Salary: 2,00000

*****

Enter the data for manager:
Enter name of the employee: Hashim
Enter the identification number of the employee: 2
Enter Title: Vice-President
Enter the Salary: 90,000

*****

Enter the data for scientist:
Enter name of the employee: Dr.Rehman
Enter the identification number of the employee: 3
Enter the number of publications: 7
Enter the Salary: Rs.3,000000

*****

Enter the data for clerk:
Enter name of the employee: Rashid
Enter the identification number of the employee: 4
Enter the number of Overtime hours worked: 60

*****

Data for manager:
Name is: Ali
ID is: 1
Title: President
Salary: 2,00000

Data for manager:
Name is: Hashim
ID is: 2
Title: Vice-President
Salary: 90,000

Data for Clerk:
Name is: Rashid
ID is: 4
Overtime: 60

Press any key to continue . . .
```

### Solution:

Enter your code here...

```
#include<iostream>
using namespace std;
class Empolyee {
private:
    string name;
    int idnumber;
public:
```

```
    Empolyee() {
        name = " ";
        idnumber=0;
    }
    Empolyee(string na,int id) {
        name = na;
        idnumber = id;
    }
    void setdata();
    void displayemployee();
};
class Manager :public Empolyee {
private:
    string title;
    double salary;
public:
    Manager() {
        title = " ";
        salary = 0;
    }
    Manager(string tit, double sa) {
        title = tit;
        salary = sa;
    }
    void inputdata();
    void displaymanager();
};
class Scientist :public Empolyee {
private:
    int numberofart;
    double salary;
public:
    Scientist() {
        numberofart=0;
        salary = 0;
    }
    Scientist(int num, double sa) {
        numberofart= num;
        salary = sa;
    }
    void inputscientist();
    void displayscientist();
};
class Clerk :public Empolyee {
private:
    int numberofhours;
public:
    Clerk() {
        numberofhours = 0;
    }
};
```

```
    }
    Clerk(int hours) {
        numberofhours = hours;
    }
    void inputclerk();
    void displayclerk();
};

void Empolyee::setdata() {
    //cout << "DATA OF MANAGER:" << endl;
    cout << "Enter the name of employee: " << endl;
    cin>> name;
    cout << "Enter the Identification Number: " << endl;
    cin >> idnumber;
}

void Empolyee::displayemployee() {
    cout << "Name: " << name << endl;
    cout << "ID: " << idnumber << endl;
}

void Manager::inputdata() {
    cout << "Enter Title: " << endl;
    cin >> title;
    cout << "Enter the salary: " << endl;
    cin >> salary;
}

void Manager::displaymanager() {
    cout << "Title: " << title << endl;
    cout << "Salary: " << salary << endl;
}

void Scientist::inputscientist() {
    cout << "Enter the number of Publications: " << endl;
    cin >> numberofart;
    cout << "Enter the salary: " << endl;
    cin >> salary;
}

void Scientist::displayscientist() {
    cout << "Number of Publications: " << numberofart << endl;
    cout << "Salary: " << salary << endl;
}

void Clerk::inputclerk() {
    cout << "Enter the Number Of Working Hours: " << endl;
    cin >> numberofhours;
}

void Clerk::displayclerk() {
    cout << "Number Of Working Hours: " << numberofhours << endl;
}

int main() {
    system("title Ahtisham khan (014) OOP LAB 9");
    Manager m1,m4;
    Scientist m2;
```

```
Clerk m3;
cout << "\t**Enter the Data For Manager** " << endl;
m1.setdata();
m1.inputdata();
cout << "\t*****" << endl;
cout << endl;
cout << "\t**Enter the Data For Manager** " << endl;
m4.setdata();
m4.inputdata();
cout << "\t*****" << endl;
cout << endl;
cout << "\t**Enter the Data for Scientist**" << endl;
m2.setdata();
m2.inputscientist();
cout << "\t*****" << endl;
cout << endl;
cout << "\t**Enter the Data for Clerk**" << endl;
m3.setdata();
m3.inputclerk();
cout << "\t*****" << endl;
cout << endl;
cout << "Data for Manager: " << endl;
m1.displayemployee();
m1.displaymanager();
cout << "\t*****" << endl;
cout << endl;
cout << "Data for Manager: " << endl;
m4.displayemployee();
m4.displaymanager();
cout << "\t*****" << endl;
cout << endl;
cout << "Data for Scientist: " << endl;
m2.displayemployee();
m2.displayscientist();
cout << "\t*****" << endl;
cout << endl;
cout << "Data for Clerk: " << endl;
m3.displayemployee();
m3.displayclerk();
system("pause");
return 0;
}
```

**Output:**

Your Output here...

Ahtisham khan (014) OOP LAB 9


```

    **Enter the Data For Manager**
Enter the name of employee:
Ahtisham
Enter the Identification Number:
1
Enter Title:
CEO
Enter the salary:
123000
*****

    **Enter the Data For Manager**
Enter the name of employee:
Maha
Enter the Identification Number:
2
Enter Title:
Founder
Enter the salary:
124000
*****

    **Enter the Data for Scientist**
Enter the name of employee:
Jibran
Enter the Identification Number:
3
Enter the number of Publications:
3
Enter the salary:
239000
*****

    **Enter the Data for Clerk**
Enter the name of employee:
Sufyan
Enter the Identification Number:
4
Enter the Number Of Working Hours:
8
```

 Ahtisham khan (014) OOP LAB 9

**Data for Manager:**

**Name: Ahtisham**

**ID: 1**

**Title: CEO**

**Salary: 123000**

\*\*\*\*\*

**Data for Manager:**

**Name: Maha**

**ID: 2**

**Title: Founder**

**Salary: 124000**

\*\*\*\*\*

**Data for Scientist:**

**Name: Jibran**

**ID: 3**

**Number of Publications: 3**

**Salary: 239000**

\*\*\*\*\*

**Data for Clerk:**

**Name: Sufyan**

**ID: 4**

**Number Of Working Hours: 8**

**Press any key to continue . . .**



Name of Student  
Reg ID

Object Oriented Programming Lab  
Lab # 04

Engr. Faisal Zia  
Dept of SE, BUIC