UNIVERSITY OF ENGINEERING AND TECHNOLOGY, TAXILA
DEPARTMENT OF COMPUTER SCIENCE
http://web.uettaxila.edu.pk

CS-103 | Object Oriented Programming (Lab)

LAB MANUAL 09

Inheritance cont…

Instructor

Muhammad Faheem Saleem

Inheritance

Inheritance is a mechanism that allows a new class (child class) to acquire the properties and behavior of an existing class (parent class). The child class can reuse, extend, or modify the functionality of the parent class.

Syntax:

```
class Parent {
    // Parent class members
};

class Child : public Parent {
    // Child class inherits Parent class
};
```

Types of Inheritance

Single Inheritance

Single inheritance is when one class (child) is derived from a single parent class. The child class inherits all the **public** and **protected** members of the parent class.

Example

```
#include <iostream>
using namespace std;
class Animal {
public:
    void eat() {
        cout << "This animal is eating." << endl;
    }
};

class Dog : public Animal {
```

```
public:
    void bark() {
        cout << "The dog is barking." << endl;
    }};

int main() {
    Dog d;
    d.eat();  // Inherited from Animal
    d.bark(); // Defined in Dog
    return 0;}
```

Multiple Inheritance

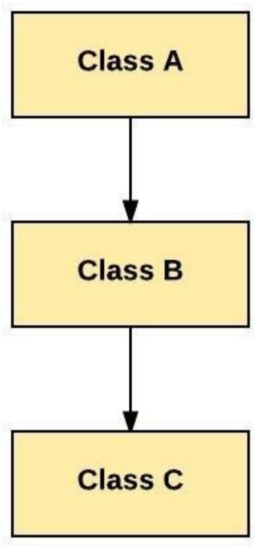
In object-oriented programming, multiple inheritance is a feature where a class can inherit from more than one parent class, allowing it to inherit properties and methods from multiple sources

Example

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

class Engine {
public:
    void startEngine() {
        cout << "Engine started." << endl;
    }
};

class Wheels {
public:
    void roll() {
        cout << "Wheels are rolling." << endl;
    }
};

class Car : public Engine, public Wheels {
public:
    void drive() {
        cout << "Car is moving." << endl;
    }
};

int main() {
    Car myCar;
    myCar.startEngine();
    myCar.roll();
    myCar.drive();
    return 0;
}
```

Multilevel Inheritance

In multilevel inheritance, a derived class inherits from a base class, and another class inherits from this derived class. **Multilevel inheritance** involves a **chain** of inheritance where a child class is derived from another child class. This helps in **hierarchical structuring** of classes



Syntax

```
class A // base class
{
    ...........
};
class B : access_specifier A // derived class
{
    ...........
} ;
class C : access_specifier B // derived from derived class B
{
    ...........
} ;
```

Example:

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

class Vehicle {                              // Base class
public:
    void display() {        cout << "This is a
vehicle." << endl;
    }
};

class Car : public Vehicle {                              // Derived class (inherits from Vehicle)

public:
    void displayCar() {
        cout << "This is a car." << endl;
    }
};

class SportsCar : public Car {                            // Derived class (inherits from Car)

public:
    void displaySportsCar() {
        cout << "This is a sports car." << endl;
    }
};

int main() {
SportsCar myCar;
    myCar.display();              // Accessing base class method
myCar.displayCar();               // Accessing intermediate derived class method
myCar.displaySportsCar();         // Accessing derived class method
return 0;
}
```

OutPut:

```
This is a vehicle.
This is a car.
This is a sports car.
```

Example

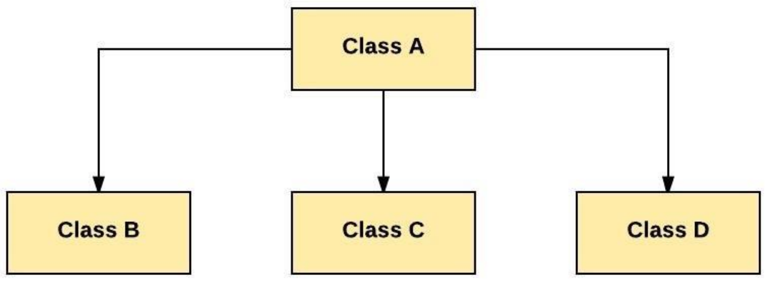
```
#include <iostream>
using namespace std;

class Grandparent {
public:
    void familyName() {
        cout << "This is the Smith family." << endl;
    }
};

class Parent : public Grandparent {
public:
    void parentInfo() {
        cout << "Parent's details." << endl;
    }
};

class Child : public Parent {
public:
    void childInfo() {
        cout << "Child's details." << endl;
    }
};
```

```
int main() {
    Child c;
    c.familyName();
    c.parentInfo();
    c.childInfo();
    return 0;
}
```

Child inherits from Parent, which inherits from Grandparent. This forms a **chain of inheritance (Multilevel Inheritance).**

Hierarchical Inheritance

In object-oriented programming, hierarchical inheritance involves multiple derived classes inheriting from a single base class, creating a tree-like structure of classes where each derived class inherits properties and behaviors from the base class. In hierarchical inheritance, multiple derived classes inherit from a single base class.



Syntax

```
class BaseClass {                    // Base class
    // Base class members and methods
};
```

```
class DerivedClass1 : public BaseClass {                    // Derived class 1 (inherits from BaseClass)

    // Derived class 1 members and methods
};

class DerivedClass2 : public BaseClass {                    // Derived class 2 (inherits from BaseClass)

    // Derived class 2 members and methods
};
```

Example

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

// Base class class
Shape { public:
    void display() {        cout << "This is a shape." << endl;
    }
};

// Derived class 1 (inherits from Shape)
class Rectangle : public Shape {
public:
    void displayRectangle() {
cout << "This is a rectangle." << endl;    } };

// Derived class 2 (inherits from Shape)

class Circle : public Shape {
public:
    void displayCircle() {
        cout << "This is a circle." << endl;     } };

int main()
{
 Rectangle rect;
 Circle circle;
rect.display();         // Accessing base class method
rect.displayRectangle();// Accessing derived class 1 method
circle.display();       // Accessing base class method
circle.displayCircle(); // Accessing derived class 2 method
return 0;
}
```

Output

```
 This is a shape.
This is a rectangle.
This is a shape.
This is a circle.
```

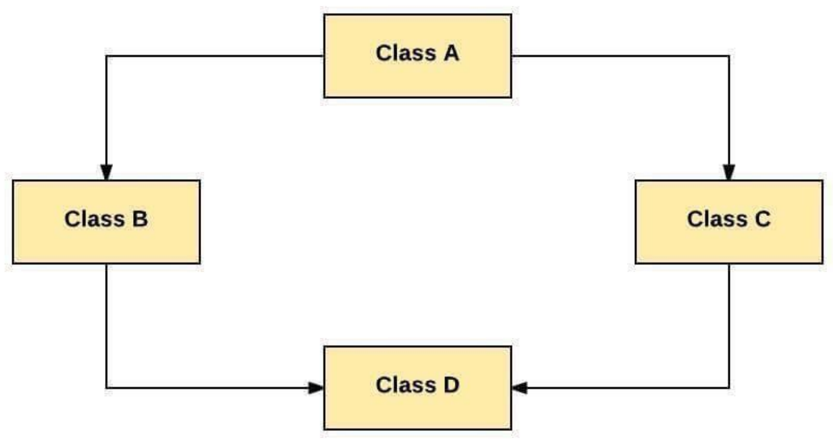
Example

```
#include <iostream>

using namespace std;


class Vehicle {

public:

    void move() {

        cout << "This vehicle moves." << endl;

    }

};
```

```
class Car : public Vehicle {
public:
    void carType() {
        cout << "This is a car." << endl;
    }
};

class Bike : public Vehicle {
public:
    void bikeType() {
        cout << "This is a bike." << endl;
    }
};

int main() {
    Car c;
    Bike b;
    c.move();
    c.carType();
    b.move();
    b.bikeType();
    return 0;
}
```

Hybrid Inheritance:

Hybrid inheritance is a complex form of inheritance in object-oriented programming (OOP). Hybrid inheritance is a combination of multiple types of inheritance. i.e is a combination of Single and Multiple inheritance. It combines multiple inheritance with other inheritance types, such as single or multilevel inheritance, resulting in a class inheriting from multiple classes, some of which might also be derived classes.



Syntax

```
class BaseClass1 {                    // Base class 1

    // Base class 1 members and methods
};

class BaseClass2 {                            // Base class 2

    // Base class 2 members and methods
};

// Derived class (inherits from BaseClass1 and BaseClass2)
class DerivedClass : public BaseClass1, public BaseClass2
{
    // Derived class members and methods
};
```

Example

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

```
class A {
public:
    void showA() {
        cout << "Class A" << endl;
    }
};

class B : public A {
public:
    void showB() {
        cout << "Class B" << endl;
    }
};

class C {
public:
    void showC() {
        cout << "Class C" << endl;
    }
};

class D : public B, public C {
public:
    void showD() {
        cout << "Class D" << endl;
    }
};

int main() {
    D obj;
    obj.showA();
    obj.showB();
    obj.showC();
    obj.showD();
    return 0;
}
```

Example

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

// Base class
class Person {
protected:
    string name;

public:
    Person(const string& name) : name(name) {}
    void display() {
        cout << "Name: " << name << endl;
    }
};

// Derived class 1: Employee
class Employee : public Person {
protected:
    int employeeId;
```

```
public:
    Employee(const string& name, int id) : Person(name), employeeId(id) {}
    void displayEmployee() {
        display();
        cout << "Employee ID: " << employeeId << endl;
    }
};

// Derived class 2: Student
class Student : public Person {
protected:
    int studentId;

public:
    Student(const string& name, int id) : Person(name), studentId(id) {}
    void displayStudent() {
        display();
        cout << "Student ID: " << studentId << endl;
    }
};

// Derived class 3: StudentIntern (Multiple Inheritance)
class StudentIntern : public Employee, public Student {
public:
    StudentIntern(const string& name, int empId, int stuId)
        : Employee(name, empId), Student(name, stuId) {}

    void displayStudentIntern() {
        cout << "Student Intern Details:" << endl;
        displayEmployee();
        displayStudent();
    }
};

int main() {
    StudentIntern SI("Hassan", 67537, 2215);
    SI.displayStudentIntern();
    return 0;
}
```

Lab Tasks

- **Implement a multilevel inheritance scenario in C++. Define a base class Person with attributes such as name and age. Derive a class Employee from Person with additional attributes like employeeID and salary. Further, derive a class Manager from Employee with a new attribute department. Create objects of each class, set values for their attributes, and display the details.**

- **Implement hierarchical inheritance in C++ to model different types of animals. Define a base class Animal with attributes such as species and sound. Derive classes Mammal and Bird from Animal, each with additional attributes specific to mammals and birds, respectively. Create objects of each class and demonstrate their attributes.**

- **Create A BankAccount class with accountNumber and balance. A SavingsAccount class that inherits from BankAccount and has interestRate. Implement functions to calculate interest and display balance.**