National University
of computer and emerging sciences

Lab 11

Items	Description
Course Title	Object Oriented Programming
Lab Title	Classes (Inheritance)
Duration	3 Hours
Tools	Eclipse/ C++
Objective	To get familiar with the use of different concepts in classes in c++

Inheritance

Inheritance is one of the four pillars of Object-Oriented Programming (OOP), along with encapsulation, abstraction, and polymorphism. Inheritance enables a new class (called the derived class or subclass) to inherit properties and behaviors from an existing class (called the base class or superclass). This allows for code reusability and helps create a hierarchical classification of classes.

```
#include <iostream>

using namespace std;

// Base Class

class Person {

public:

   string name;
```

```
    int age;
    // Base class constructor
    Person(string n, int a) : name(n), age(a) {}
    // Base class member function
    void displayInfo() {
        cout << "Name: " << name << ", Age: " << age << endl;
    }
};
// Derived Class
class Student : public Person {
public:
    int studentID;
    // Derived class constructor
    Student(string n, int a, int id) : Person(n, a), studentID(id) {}
    // Derived class member function
    void displayStudentInfo() {
```

```
        displayInfo();  // Calling base class function
        cout << "Student ID: " << studentID << endl;
    }
};
int main() {
    // Create an object of the derived class
    Student s("Alice", 20, 12345);
    // Call derived class function
    s.displayStudentInfo();
    return 0;
}
```

Types of Inheritance in C++

There are five main types of inheritance in C++:

1. **Single Inheritance**
2. **Multiple Inheritance**
3. **Multilevel Inheritance**
4. **Hierarchical Inheritance**
5. **Hybrid Inheritance**

Let's go through each type in detail, with code examples.

1. Single Inheritance

In single inheritance, a derived class inherits from only one base class.

```
#include <iostream>

using namespace std;

class Animal {

public:

    void eat() {

        cout << "Eating..." << endl;

    }

};
```

```
class Dog : public Animal { // Dog inherits from Animal
public:
    void bark() {
        cout << "Barking..." << endl;
    }
};
int main() {
    Dog d;
    d.eat();   // Inherited from Animal
    d.bark();  // Defined in Dog
    return 0;
}
```

Explanation:

Here, the `Dog` class is derived from the `Animal` class. The `Dog` class inherits the `eat()` function from the `Animal` class and also has its own `bark()` function.

2. Multiple Inheritance

In multiple inheritance, a derived class inherits from more than one base class. C++ is one of the few languages that support multiple inheritance.

```
#include <iostream>

using namespace std;

class Animal {

public:

    void eat() {

        cout << "Eating..." << endl;

    }

};

class Bird {

public:

    void fly() {

        cout << "Flying..." << endl;

    }

};
```

```
class Bat : public Animal, public Bird { // Bat inherits from
Animal and Bird
public:
    void sleep() {
        cout << "Sleeping..." << endl;
    }
};
int main() {
    Bat b;
    b.eat();    // Inherited from Animal
    b.fly();    // Inherited from Bird
    b.sleep();  // Defined in Bat
    return 0;}
```

Explanation:

Here, the `Bat` class inherits from both `Animal` and `Bird` classes. It gains access to both the `eat()` and `fly()` functions from the respective base classes, and it also has its own `sleep()` function.

3. Multilevel Inheritance

In multilevel inheritance, a class is derived from another derived class, creating a chain of inheritance.

```
#include <iostream>

using namespace std;

class Animal {

public:

    void eat() {

        cout << "Eating..." << endl;

    }

};

class Mammal : public Animal { // Mammal inherits from Animal

public:

    void breathe() {

        cout << "Breathing..." << endl;

    }

};
```

```
class Dog : public Mammal { // Dog inherits from Mammal
public:
    void bark() {
        cout << "Barking..." << endl;
    }
};
int main() {
    Dog d;
    d.eat();     // Inherited from Animal
    d.breathe(); // Inherited from Mammal
    d.bark();    // Defined in Dog
    return 0;
}
```

Explanation:

In this example, `Dog` inherits from `Mammal`, which in turn inherits from `Animal`. This creates a chain of inheritance. The `Dog` class can access `eat()` from `Animal`, `breathe()` from `Mammal`, and has its own `bark()` function.

4. Hierarchical Inheritance

In hierarchical inheritance, multiple classes inherit from a single base class. This is useful when multiple classes share common behavior from a single base class.

```
#include <iostream>

using namespace std;

class Animal {

public:

    void eat() {

        cout << "Eating..." << endl;

    }

};

class Dog : public Animal { // Dog inherits from Animal

public:

    void bark() {

        cout << "Barking..." << endl;

    }

};
```

```
class Cat : public Animal { // Cat also inherits from Animal
public:
    void meow() {
        cout << "Meowing..." << endl;
    }
};
int main() {
    Dog d;
    d.eat();   // Inherited from Animal
    d.bark();  // Defined in Dog
    Cat c;
    c.eat();   // Inherited from Animal
    c.meow();  // Defined in Cat
    return 0;
}
```

How to implement inheritance in Header files.

Person.h

```
// Person.h
#ifndef PERSON_H
#define PERSON_H

#include <string>
using namespace std;

class Person {
protected:
    string name;
    int age;

public:
    Person(string n, int a);
    void displayInfo();
};

#endif
```

Student.h

```
// Student.h
#ifndef STUDENT_H
#define STUDENT_H

#include "Person.h"

class Student : public Person {
private:
    int studentID;

public:
    Student(string n, int a, int id);
    void displayStudentInfo();
};

#endif
```

Person.cpp

```cpp
// Person.cpp
#include "Person.h"
#include <iostream>
using namespace std;

Person::Person(string n, int a) : name(n), age(a) {}

void Person::displayInfo() {
    cout << "Name: " << name << ", Age: " << age << endl;
}
```

Student.cpp

```cpp
// Student.cpp
#include "Student.h"
#include <iostream>
using namespace std;

Student::Student(string n, int a, int id) : Person(n, a),
studentID(id) {}

void Student::displayStudentInfo() {
    displayInfo();  // Call the base class method
```

```
    cout << "Student ID: " << studentID << endl;
}
```

Main.cpp

```
// main.cpp
#include "Student.h"
#include <iostream>
using namespace std;

int main() {
    // Create a Student object
    Student s("Alice", 20, 12345);

    // Display the student's information
    s.displayStudentInfo();

    return 0;
}
```

National University
of computer and emerging sciences

Lab Task

Create Header files

Task 1: Employee Management System

Objective: Create a simple employee management system with a base class `Employee` and a derived class `Manager`.

Requirements:

1. **Base Class** (`Employee`):
    - Attributes: `name` (string), `employeeID` (int), `baseSalary` (double).
    - Method: `displayInfo()` to print the employee's name, ID, and base salary.
2. **Derived Class** (`Manager`):
    - Additional Attribute: `bonus` (double).
    - Additional Method: `calculateTotalSalary()` which calculates the total salary by adding `baseSalary` and `bonus`.
    - Override the `displayInfo()` method to include the `bonus` and `total salary` as well.
3. **Main Program**:
    - Create an `Employee` object and a `Manager` object, initialize them, and call their respective `displayInfo()` functions.

This task will help students understand how to override methods and manage data specific to derived classes.

Task 2: Library System

Objective: Develop a library system that uses inheritance to separate general item attributes from specific book attributes.

Requirements:

1. **Base Class** (`LibraryItem`):
    - Attributes: `title` (string), `author` (string), `publicationYear` (int).

- Method: `displayItemInfo()` to print title, author, and publication year.

2. **Derived Class** (`Book`):
   - Additional Attribute: `isbn` (string) to store the book's ISBN.
   - Method: `isAvailable()` to randomly (or with a predefined boolean value) return whether the book is available or checked out.
   - Override `displayItemInfo()` to include `isbn` and availability status.
3. **Main Program**:
   - Create and initialize a few `Book` objects, then display their information along with their availability status.

This task challenges students to extend a general class for a more specific purpose and modify the base class method.

Task 3: E-Commerce Product System

Objective: Design a system where a `Product` class provides basic attributes, and a `Clothing` class adds specific attributes for clothing items.

Requirements:

1. **Base Class** (`Product`):
   - Attributes: `productName` (string), `price` (double).
   - Method: `displayProductInfo()` to show product name and price.
2. **Derived Class** (`Clothing`):
   - Additional Attributes: `size` (string), `color` (string).
   - Method: `applyDiscount(double percentage)` to reduce the price by a given percentage.
   - Override `displayProductInfo()` to show the `size`, `color`, and discounted price if a discount has been applied.
3. **Main Program**:
   - Create a `Clothing` object, apply a discount, and display the product information before and after applying the discount.

This task encourages students to manage additional attributes and behaviors related to derived classes while understanding the importance of price calculations.

Task 4: Bank Account System

Objective: Build a basic banking system where a `BankAccount` class is inherited by a `SavingsAccount` class.

Requirements:

1. **Base Class** (`BankAccount`):
    - Attributes: `accountNumber` (string), `balance` (double).
    - Method: `deposit(double amount)` to add an amount to the balance, and `displayAccountInfo()` to print account number and balance.
2. **Derived Class** (`SavingsAccount`):
    - Additional Attribute: `interestRate` (double, percentage).
    - Additional Method: `applyInterest()` that calculates and adds the interest to the balance.
    - Override `displayAccountInfo()` to also show the `interestRate`.
3. **Main Program**:
    - Create a `SavingsAccount` object, make a few deposits, apply interest, and display the account details each time.