|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | **Programming Fundamentals** | |  | **(CL214)** | |  | **LABORATORY MANUAL** | |  | **Spring 2021** | |  | **C:\Users\Aamer\Desktop\nu-new.png**  **LAB 11** | |  | **Class Relationship**  **Engr. Ibrar Khan** | |  | **Engr. Sana Saleh** |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_\_\_\_\_\_\_\_\_ | | | \_\_\_ | | STUDENT NAME | | ROLL NO | | | SEC | |  | | | | | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | | LAB ENGINEER SIGNATURE & DATE | | | | | | | **MARKS AWARDED: /10** | | | | | | |  | | | | | | | **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), ISLAMABAD** | | | | | | |  | | | | | | | Prepared by: | Engr. Sana Saleh | | Date: | 08 April, 2019 | | | Verified by: | Engr. Shahid Qureshi | | Date: | 08 April, 2019 | | |

|  |  |
| --- | --- |
| **LAB 11** | **Class Relationship** |

**Lab Objectives:**

1. To learn one of the class relationships i.e. association, composition and aggregation.
2. To learn the advantages of inheritance.
3. To learn different types of inheritance.
4. To learn to create and use object files.

**Software Required:**

* Dev C++

**Introduction:**

1. **Inheritance**

Inheritance is the way for building new classes from existing classes. This is one of the strongest features of OOP. It is mainly because of this feature, we can claim that we can reuse the existing code. Now the question arises that how we can reuse code? The answer is very simple. Suppose we have two classes which are named as class A and class B. We also suppose that class A is the parent class and class B is the child or inherited class. Further suppose that the parent class carries four methods, one for setting name, one for displaying name, one for setting DOB and one for displaying DOB. Now coming towards child class we are supposing that it owns two methods, one for setting registration number and the other for displaying the registration number. According to the doctrine of inheritance we can say with full confidence that without mentioning the child class, which in this case is class B, by default contains the public methods of the parent class, which in our case is class A. So according to this feature, it can be safely concluded that class B contains six methods.

Two terms are strongly related with the concept of inheritance:

* + Base Class
  + Derived Class

The class from which we build new class would be termed as base class and the new class would be termed as derived class.

**Note:** *Any class can be a base class in the derivation. Base classes have no special code features.*

A derived class contains all the member and functions of a base class. The derived class can add new member variables and new member functions. The additional functions are often called **incremental function** and additional data is often called **incremental data**.

From coding perspective the primary benefit of derivation is:

* Higher level of code modularization and localization.
* Code is easier to maintain and modify.
* Instead of duplicating the variables and code of a base class when building a derived class, the derived class incorporates the existing code of the base class.

**What is the proper time for the application of inheritance?**

When "IS A" relationship is developed between two classes: for example if machine is the base class and the derived class is computer then we can say that computer "IS A" machine. Similarly if shape is the super class and square is the subclass then we are comfortable in saying that square "IS A" a shape. In this relationship an object of the derived class may also be treated as object of the base class.

In the above examples machine and shape are the base classes whereas; computer and square are the derived classes.

## Access Control & Inheritance

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 Class | Yes | Yes | No |
| Outside Class | Yes | No | No |

1. **Types of Inheritance**

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:

*class derived-class: access-specifier base-class*

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.

We hardly use **protected** or **private** inheritance, but **public** inheritance is commonly used. While using different type of inheritance, 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.

1. **Example Program of Inheritance**

The following program tells you in what order constructors and destructors are called. For learning the syntax for inheritance and one of its types (public inheritance), please consider this code:

|  |
| --- |
| #include <iostream>  using namespace std;  class Person { public:  string profession;  int age;  Person(): profession("unemployed"), age(16) { }  void display()  {  cout << "My profession is: " << profession << endl;  cout << "My age is: " << age << endl;  walk();  talk();  }  void walk() { cout << "I can walk." << endl; }  void talk() { cout << "I can talk." << endl; }  };  // MathsTeacher class is derived from base class Person.  class MathsTeacher : public Person  {  public:  void teachMaths() { cout << "I can teach Maths." << endl; }  };  // Footballer class is derived from base class Person.  class Footballer : public Person  {  public:  void playFootball() { cout << "I can play Football." << endl; }  };  int main()  {  MathsTeacher teacher;  teacher.profession = "Teacher";  teacher.age = 23;  teacher.display();  teacher.teachMaths();  Footballer footballer;  footballer.profession = "Footballer";  footballer.age = 19;  footballer.display();  footballer.playFootball();  return 0;  } |

**Output:**

My profession is: Teacher

My age is: 23

I can walk.

I can talk.

I can teach Maths.

My profession is: Footballer

My age is: 19

I can walk.

I can talk.

I can play Football.

1. **Association:**

Association is a simple structural connection or channel between classes and is a relationship where all objects have their own lifecycle and there is no owner.

1. **Composition:**

To qualify as a **composition**, an object and a part must have the following relationship:

* The part (member) is part of the object (class)
* The part (member) can only belong to one object (class) at a time
* The part (member) has its existence managed by the object (class)
* The part (member) does not know about the existence of the object (class)

A good real-life example of a composition is the relationship between a person’s body and a heart.

A chart below shows the difference between composition and association:

|  |  |  |
| --- | --- | --- |
| **Property** | **Composition** | **Association** |
| Relationship type | Whole/part | Otherwise unrelated |
| Members can belong to multiple classes | No | Yes |
| Members existence managed by class | Yes | No |
| Directionality | Unidirectional | Unidirectional or bidirectional |
| Relationship verb | Part-of | Uses-a |

**Practice Problems:**

***Association***

Q1: Create two classes patient and doctor. A patient class contain patient name, age, disease name, doctor’s data (i.e. ID) he wants to visit and duration of time he is taking medicine whereas doctor class contains doctor name, ID and his specialization.

In main create two objects of doctor class and one object of patient class. Fill doctor’s information using parametrized constructor and show that data on console. After that ask the patient to enter his data and ID of Doctor he wants to visit. Finally print the complete patients and Doctor information (that he is going to visit) on the console.

|  |
| --- |
| class Doctor  {  private:  char name[40];  char specialization[40];  int doctor\_ID;  public:  Doctor(char \*,char \*);  void get\_data();  void set\_data();  };  class patient  {  private:  char name[40]; char disease\_name[40]; int doctor\_ID; int duration, age;  public:  patient();  void set\_data();  void get\_data();  }; |

***Composition***

Q2: Create two classes question and option. Each option class has an integer option number and a string that stores the option. Question class contains 4 object of option class and correct option number. From a text file given read and store 2 questions statement and options.

Then ask user each question on console question, and check option given by user is correct or not.

|  |
| --- |
| class Option  {  private:  int option\_no;  string option\_statement;  public:  Option();  void set\_option(fstream &infile,**int**);  void get\_option();  };  class Question  {  private:  string question\_statement;  Option A,B,C,D;  int correct\_answer;    public:  void set\_question(fstream &infile);  void get\_question();  int get\_correct\_answer();  }; |