Date
Page

Assignment - A3

Student Name :→ Sameer Manoj Bramhecha.
Batch :→ E-1
Roll No :→ 21115
Date of Performance :→ 17-09-2021
Date of Submission :→ 24-09-2021

Title :→ Demonstrate reusability of code through inheritance and use of exception handling.

Problem Statement :→ Imagine a publishing company which does marketing for book and audiocassette versions. Create a class publication that stores the title (a string) and price (type float) of a publication. From this class derive two classes : book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float).
Write a program that instantiates the book and tape classes, allows user to enter data and displays the data members. If an exception is caught, replace all the data member values with zero values.

Learning Objectives :→
① To learn and understand code reusability and demonstrate it using Inheritance concepts.
② To learn, understand and demonstrate exception handling in object oriented environment.

Date
Page

S/w and H/w Requirement: 64 bit, Windows, Open Source C++ programming tool like G++/GCC, OPENGL

Theory :→

Inheritance :→ It is a property in which data members & and member functions of some class are used by some other class. It allows the reusability of code in C++.

Base class and Derived class :→

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 reffered to as the derived class. A derived class represents a more specialized group of objects. Typically, a derived class contains behaviours inherited from its base class plus additional behaviours. A class can be derived from more than one class, which means it can inherit data and functions from multiple base classes.

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 :→

```
class derived-class: visibility-mode base-class [, visibility-mode baseclass2]
{
  //body of derived class.
};
```

Where, visibility_mode is the one of public, protected, or private and base-class is the name of a previously defined class. If By default, visibility mode is private.

→ Access-Specifer :→
There are 3 types of access-specifier's or qualifier's using which the members of the class are accessed by the other class →

1) Private.
2) Public
3) Protected.

- If a base class has private members then those members are not accessible to derived class.
- Protected members are public to derived classes but private to rest of the program.
- Public members are accessible to all.

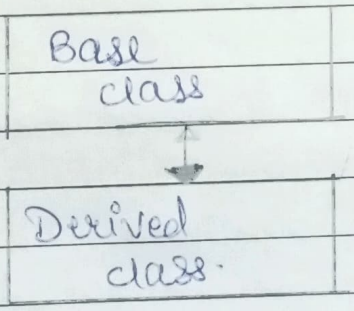
① Public Inheritance: When deriving a class from a public base class, public members of the base class become ~~protected members~~ public members of the derived class and protected members of the base class become protected members of the derived class. A base class's private member's are never accessible directly from a derived class, but can be accessed through calls to the public & protected members of the base class.

② Protected Inheritance :- When deriving from a protected base class, public and protected members of the base class become private members of the derived class.

③ 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 :→

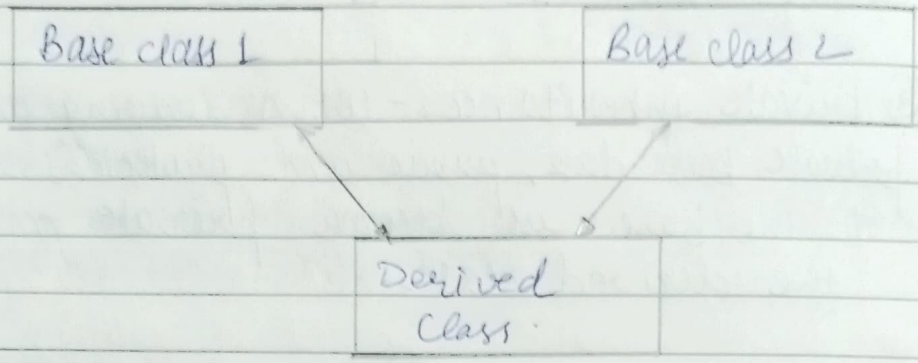
1) Single Level Inheritance :→ In single inheritance there is one parent per derived class.



Syntax:

```
class A1        // Base class.
{
  // body
};
class A2 : public A1        // derived class.
                            // public inheritance.
{
  // body
};
```

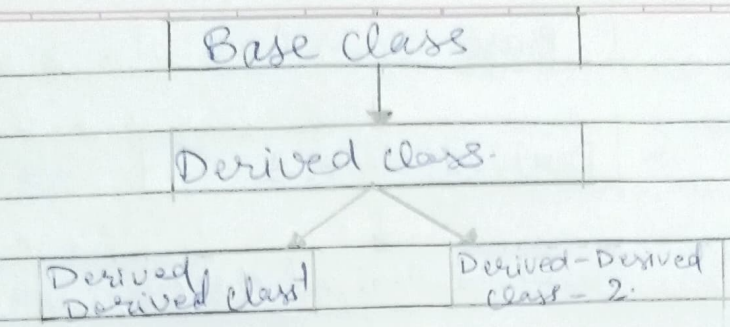
② Multiple Inheritance: In multiple inheritance the derived class is derived from more than one base class.



Syntax:

```
class C1
{
//body
};
class C2
{
//body
};
class C3 : public C1, public C2
{ //body };
```

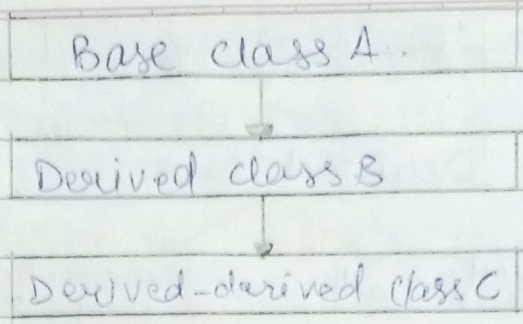
③ Hierarchical Inheritance: It is a kind of inheritance in which one or more classes are derived from the common base class. In this type of inheritance the subclass can inherit the properties of its parent classes and at the same time it can add its new features. The subclass can serve as a base class for the lower level classes.



Syntax.

```
class A   // Base class
{
//body
}
class B: public A   // Derived class
{
//body
}
class C : public B   // Derived-derived class 1
{
//body
}
class D : public B.   // Derived-derived class 2.
{
// body.
}.
```

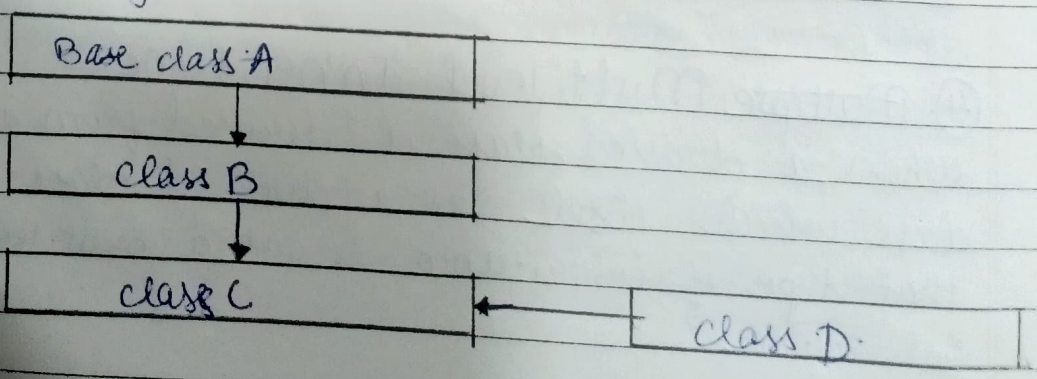
④ ~~Multiple~~ Multilevel Inheritance.
When a derived class is derived from a base class which itself is a derived class then that type of inheritance is called multilevel inheritance.



Syntax:

```
class A          //Base class
{
  //body
}
class B: public A   // Derived class.
{
 //body
}
class C: public B   //derived-derived class.
{
 //body
}
```

(5) Hybrid Inheritance: When 2 or more types of inheritances are combined together then it forms the hybrid inheritance.



Syntax :->

```
class A
{
  // body
}

Class B : public A
{
  // body
}
Class C : public B, public D
{
  // body
}
class D
{
 // body
}
```

Exception Handling :-> An exception occurs when an unexpected error or unpredictable behaviours happened on your program not caused by the operating system itself. These exceptions are handeled by code which is outside the normal flow of control and it needs an emergency exit. Compared to the structured exception ~~or~~ handling, returning an integer as an error flag is problematic when dealing with objects. The C++ exception-handling can be a full fledged object, with data members and member functions.

In C++, exception handling mechanism makes use of three keywords:- try, catch and throw.

The try represents the block of statements in which there are chances of occurring some exceptional conditionals.

When an exception is detected it is thrown using the throw statement.

There exists a block of statements in which the exception thrown is handled appropriately. This block is called as catch block.

Syntax:→

```
try
{
    throw exception;
  // exception is some value.
  /* the portion of the code that is to be monitored for
     error detection */
}
catch (argument)
{
  // catch block softly handles the exception.
}
```

Algorithm :→

1) Start.

2) Create classes Publication, book and tape.

3) Publication class contains data members name (string) and price (type float).

4) Book class contains data members pages (type int) and member function read() and ~~display~~. display().

5) Tape class contains data member minutes (type float) and member functions read() and display().

6) In main(), create an object b of book class and object t of tape class

7) ~~Call the~~ Stop.

Test Cases :→

Test case No.	Test case Description.	Input	Expected Output.	Actual Output.	Result
1.	read data and print data for Book class	name = OOP price = 300.0 pages = 500	name = OOP price = 300.0 pages = 500	name = oop price = 300 pages = 500	Pass.
2.	read data & display data for tape class	name = AB price = 200 minutes = 50	name = AB price = 200 minutes = 50	name = —— price = 0.0 minutes = 0.0	fail

Date
Page

Conclusion:→ Hence, we have learnt to use and demonstrate concepts of inheritance and exception handling.