Assignment - 03 (OOPs)

Roll No. 21123

Batch - SE1 (F1)

DOP:

DOS:

Title - Demonstrate reusability of code through inheritance and use of exceptional handling.

Objective - 1) To learn and understand code reusability and demonstrate it using inheritance concept.

2) To learn, understand and demonstrate exception handling in object oriented environment.

Problem statement -

Imagine a publising company which does marketing for book and audio cassette versions. Create a class publication that stores the title(string) and price(float) of a publication from this class drive two classes book, which adds a page count (type int) and tape, which adds a play time in minutes (float).

Write a program that instantianties the book and tape classes allows user to enter data and displays data members. If an exception is catch, replace all data members values with zero values.

S/W and H/W Requirements - Open source c++ programming tool like G++/GCC, eclipse IDE.

11/24/2021

Concept Related theory:

Inheritance- c++ support concept of Reusability once class has been written and tested, it can be adapted by other programmers to suit their requirement. This is basciaUy done by creating new classes, reusing the properTities of existing ones. The mechanism of deriving a new class from an old one is called inheritance. the old class is referred to as base class or super class and new one is called as derived class or subclass.

Base class and derived class: when creating a class, instead of writing compeletly new data members and member functions the programmer can designate that new class should inherit members of an existing class. This existing class is called Base calass and new class is referred to as the derived class. A derived class represents a more specialized group of objects. Typically, a derived class contains behaviour inherited from its base class plus additional behaviour. A class can be derived from more than one class, which means it can inherit data and function from multiple base classes.

Syntax for derived class:

```
class derived-class-name : visibity-mode base-class_name
{
    // body of derived class
};
```

where visibity mode is one of public, protected and private. and base class is the name of previously defined class. If the visibity-mode is not used, then it is private by default, c++ offers three forms of Inheritance, public protected and private.

Access control and Inheritance - A derived class can access all non-private members of its base class. Thus base class members that should not be accessible to members function of derived class should be decleared private in base class.

Access	Public	Protected	Private
Same class	yes	yes	yes
Derived class	yes	yes	NO
outside class	yes	NO	NO.

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

2) Protected Inheritance - when deriving from protected base class public and protected members of base class become protected member of derived class.

3) Private Inheritance - when deriving from private base class, public and Inherited members of base class become private members of derived class.

Types of Inheritance - New classes can be built from the existing class. It means that we can add additional features to an existing class. The new class is reffered as derived class or subclass and original class is known as base class or super class. following are types - 1) single level Inheritance
2) Multiple Inheritance 3) Heirarchical Inheritance
4) Multilevel Inheritance 5) Hybrid Inheritance.

Exceptional Handling - An exceptional occurs when an unexcepted error behaviour happened on your program not caused by operating system itself. These exceptions are handled by code which is outside the normal flow of control & its needs an error flag is problematic when dealing with objects. the c++ exception handling can be full-fledged object, with data members and member function.

Following is try, throw, catch program segment example:
Syntax:

```
try
{
  compound-statement handler-list
  handler list here
  throw expression
}
```

throw expression
}
catch (exception-declearation) compound statement
{
exception-declaration:
type specifier -list here
}
try: try block is group of c++ statement, enclosed in curly braces{}, that might cause exception. This grouping restricts exception handlers to exception generated within try block may have one or more associated catch blocks.

If no exception is thrown during execution of graduated section, the catch causes that follow try block are not executed or bypassed. Execution continous at statement after last catch cause following try block in which exception is thrown.

If an exception is thrown during execution of guarded section or in any routine the guarded section calls either directly or indirectly such as function, an exception object will be created from object created by thrown operand. At this point, the compiler looks for catch clause in higher execution context that can handle an exception.

Throw: The throw statement is used to throw an exception & its value to a matching catch exception handler. A regular throw consist of keyword throw & expression.

catch:

1) A catch block is grp of c++ statements that are used to handle or specific thrown exception. one or more catch block, or exception handlers, can be place for single try block. A catch block is specified by key keyword catch.

2) A catch parameter, enclosed in parenthesis (), which corresponds to specific type of exception that may be thrown by try block.

3) A grp of statements, enclosed in curly braces {}, whose purpose is to handle the exception.

Algorithm.

1) Declare class publication.
2) Declare two data members title & price.
3) Declare & define class inherited from publication and declare datamember pagecount and two function setdata & display()
4) Declare & define class tape inherited from publication & details data member play time & two fn; setdata() & display().
5) Define set details() fn to accept input for both derived class.
6) Use exception handling to set conditions to these inputs such as len(title) <3 or price < 0 or pagecount < 0 or playtime < 0 or playtime < 0.
7) Define display() fn to output these values for both derived class
8) In main() fn to output these vaules of two object, one for each class both tape & class
9) Call set data () & display() fn through both object.
10) stop.

Test cases.

Sr.No.	Description.	Input	Excepted O/P	Actual O/P	Result.
1.	Taking complete title-name as I/p	title- A B C	A B C	A B C	Pass.
2.	No title is given	title=""	""	title="" price = 0 pagecount = 0	Pass.
3.	title given as No.	title =123	Invalid book	title="" price = 0 pagecount = 0	Pass.
4.	Entering choice other than available Choice	11	Invalid choice	Please enter choice in range	Pass.
5.	Deleting the book having No data.	A	Invalid book	data was not created for book	Pass.
6.	Searching book	A	BOOK found	details of book A are displaced	Pass.