**S E - C o m p - A** **O b j e c t O r i e n t e d P r o g r a m m i n g L a b**

**Experiment No.2**

1. **Title: Demonstrate reusability of code thru Inheritance and use of exception handling.**
2. **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 class, allows user to enter data and displays the data members. If an exception is caught, replace all the data member values with zero values.**

1. **Objectives:**

**Students will able to understand the concept of inheritance & its various types, Exception & how to handle Exception.**

1. **Outcomes:**

**To learn and understand code reusability and demonstrate it using Inheritance concepts.**

**To learn, understand and demonstrate exception handling in object-oriented environment.**

1. **Software/Hardware/Other Requirements:**

**Any CPU with Pentium Processor / similar, 256 MB RAM/ more, 1GB HDD / more. Operating System – ubuntu/Fedora 64bit OS**

**Software: G++ compiler/ GCC compiler, Code Editor**

1. **Theory:**

**A. Inheritance:**

**C++ supports the concept of reusability once a class has been written and tested, it can be adapted by other programmers to suit their requirements. This is basically done by creating new classes, reusing the properties of the 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 the new one is called as derived class or subclass.**

**A derived class represents a more specialized group of objects. Typically, a derived class contains behaviors inherited from its base class plus additional behaviors.**

**B. Base Class & Derived Class:**

**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 : 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.**

**C++ offers three forms of inheritance—public, protected and private.**

**25 | P a g e** **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**S E - C o m p - A** **O b j e c t O r i e n t e d P r o g r a m m i n g L a b**

**C. Access Control and Inheritance:**

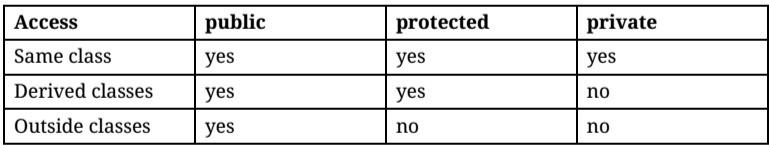
**A derived class can access all the non-private members of its base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class.**

**We can summarize the different access types according to who can access them in the following way:**

**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 as explained above. We hardly use protected or private inheritance, but public inheritance is commonly used.**

**While using different type of inheritance, following rules are applied:**

* 1. **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.**
  2. **Protected Inheritance: When deriving from a protected base class, public and protected members of the base class become protected 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.**



**D. Types of Inheritances:**

**Following are the types of Inheritance:**

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

**c) Hierarchical Inheritance:**

**Properties of one class is inherited in more derived class is known as “hierarchical inheritance”**

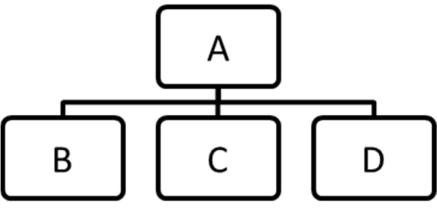
**Syntax:**

**class base\_class\_name{--------};**

**class derived\_class\_name1: visibility mode base\_class\_name{-----------------};**

**class derived\_class\_name2: visibility mode base\_class\_name{----------------};**

**class derived\_class\_name3: visibility mode base\_class\_name{------------------};**



**26 | P a g e** **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**S E - C o m p - A** **O b j e c t O r i e n t e d P r o g r a m m i n g L a b**

**Here base\_class\_name is Super class and derived\_class1, derived\_class2, derived\_class3 are the derived classes which can inherits the base\_class\_name class.**

**Visibility mode has been either from public, protected or private.**

**Here we can create objects of the derived classes from these objects we can call the base class member functions.**

**Example:**

**class Student {-------};**

**class Marksheet : public Student{----------------};**

**class Sport : public Student{----------------};**

**E. Exception Handling:**

**An exception occurs when an unexpected error or unpredictable behaviors happened on your program not caused by the operating system itself. These exceptions are handled by code which is outside the normal flow of control and it needs an emergency exit.**

**Compared to the structured exception 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. Such an object can provide the exception handler with more options for recovery. A clever exception object, for example, can have a member function that returns a detailed verbal description of the error, instead of letting the handler look it up in a table or a file.**

**C++ has incorporated three operators to help us handle these situations: try, throw and catch. The following is the try, throw…catch program segment example:**

**Syntax:**

**try**

**{**

**Compound-statement handler-list**

**handler-list here**

**The throw-expression:**

**throw expression**

**}**

**catch (exception-declaration) compound-statement**

**{**

**Exception-declaration:**

**type-specifier-list here**

**}**

**try: A try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.**

**throw: A program throws an exception when a problem shows up. This is done using a throw keyword.**

**catch: A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.**

**27 | P a g e** **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**S E - C o m p - A** **O b j e c t O r i e n t e d P r o g r a m m i n g L a b**



**C++ Program to demonstration of Exception Handling**

**#include<iostream>**

**using namespace std;**

**int main(){**

**int c;**

**try{**

**cout<<"\n\t Enter Positive Number only:";**

**cin>>c;**

**if(c<0)**

**throw c;**

**else**

**cout<<"\n Number is:"<<c;**

**}**

**catch(int x){**

**cout<<"This is Negative Number."<<x;**

**}**

**return 0;**

**}**

**Output:**

**Enter Positive Number only: -53**

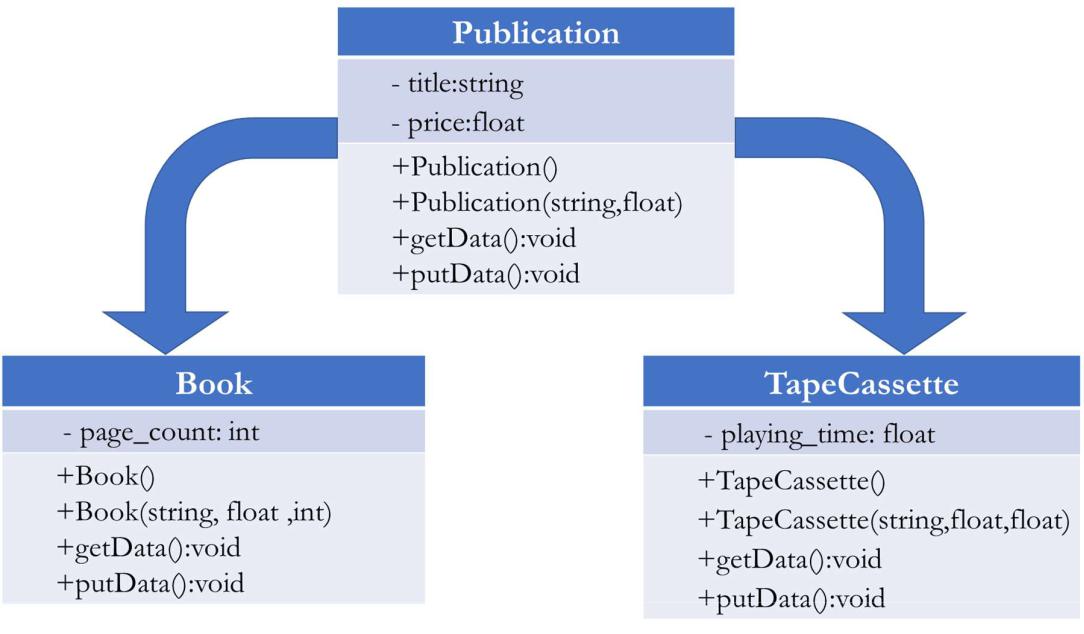
**This is Negative Number. -53**

1. **Algorithm for the problem Statement:**
   1. **Create a class Publication with variable title and price.**
   2. **Define default and parameterized constructors to initialize title and price variables**
   3. **Define function getData() to accept values from user**
   4. **Define a function putData() to print these values.**
   5. **In putData(), check if length of title is more than 3 chars else generate exception and handle exception.**
   6. **In putData(), check if price is non negative else generate exception and handle exception.**
   7. **Create a derived class Book from base Publication with variable number of pages.**
   8. **Define default and parameterized constructors to initialize variables**
   9. **Define function getData() to accept values from user**
   10. **Define a function putData() to print these values.**
   11. **In putData(), check if number of pages is non negative else generate exception and handle exception.**
   12. **Create a derived class TapeCassette from base Publication with variable playing\_time.**
   13. **Define default and parameterized constructors to initialize variables**
   14. **Define function getData() to accept values from user**
   15. **Define a function putData() to print these values.**
   16. **In putData(), check if number of pages is non negative else generate exception and handle exception.**
   17. **In main function, create two objects, one of Book class and other of Tape\_Cassette class. Use appropriate constructors and methods to print details of Book and Tape\_Cassette.**

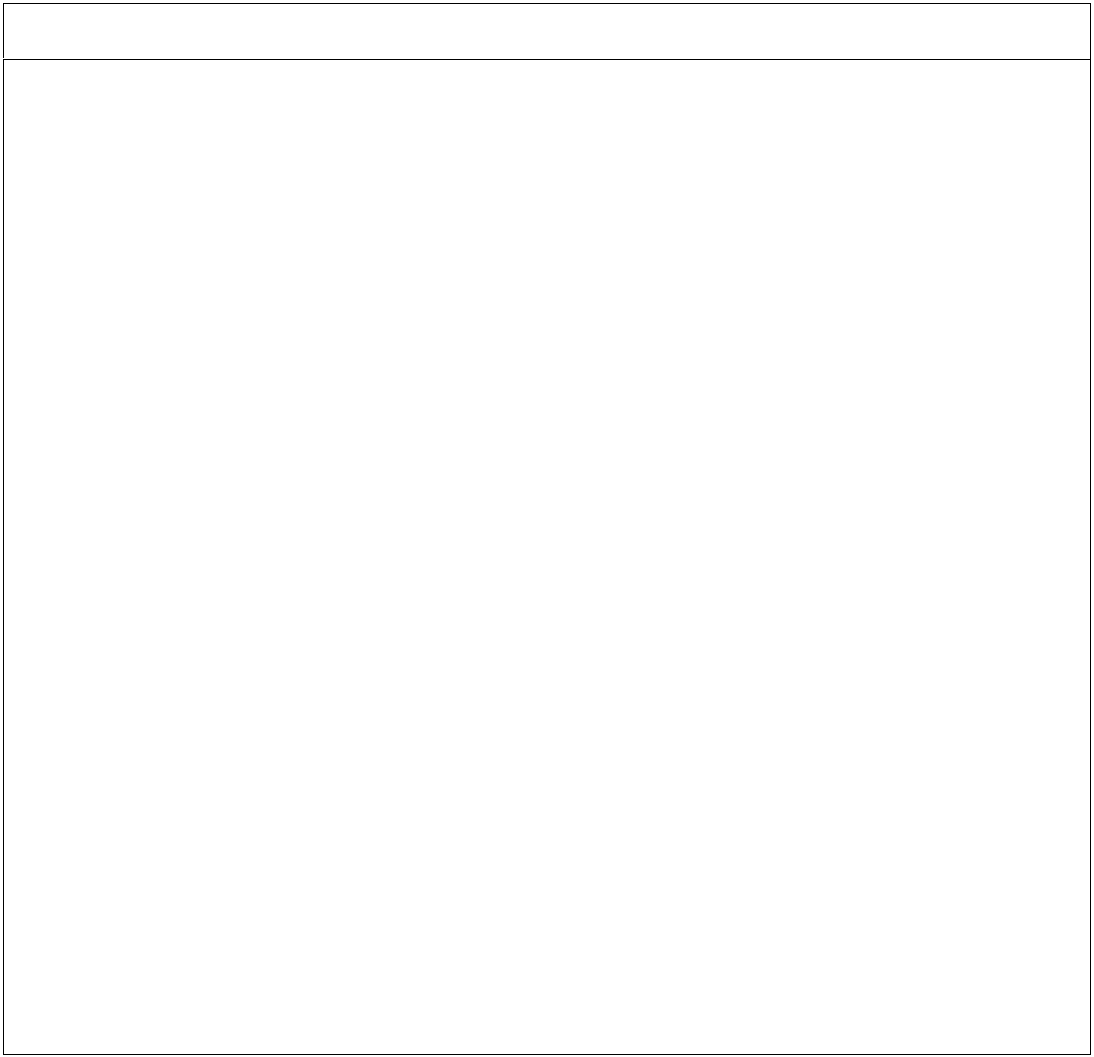
**28 | P a g e** **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**S E - C o m p - A** **O b j e c t O r i e n t e d P r o g r a m m i n g L a b**

**8. Flowchart/Class Diagram**



**9. Implementation:**



**C++ Program for Hierarchical Inheritance**

**#include <iostream>**

**using namespace std;**

**class Publication**

**{**

**public:**

**Publication ()**

**{**

**title="";**

**price=0.0;**

**}**

**Publication (string title, float price)**

**{**

**this->title=title;**

**this->price=price;**

**}**

**void getData ()**

**{**

**cout<<"\nEnter title and price\n";**

**cin>>title>>price;**

**}**

**void putData ()**

**{**

**try**

**{**

**if(title.length()<3)**

**throw title;**

**cout<<"\nTitle is :"<<title;**

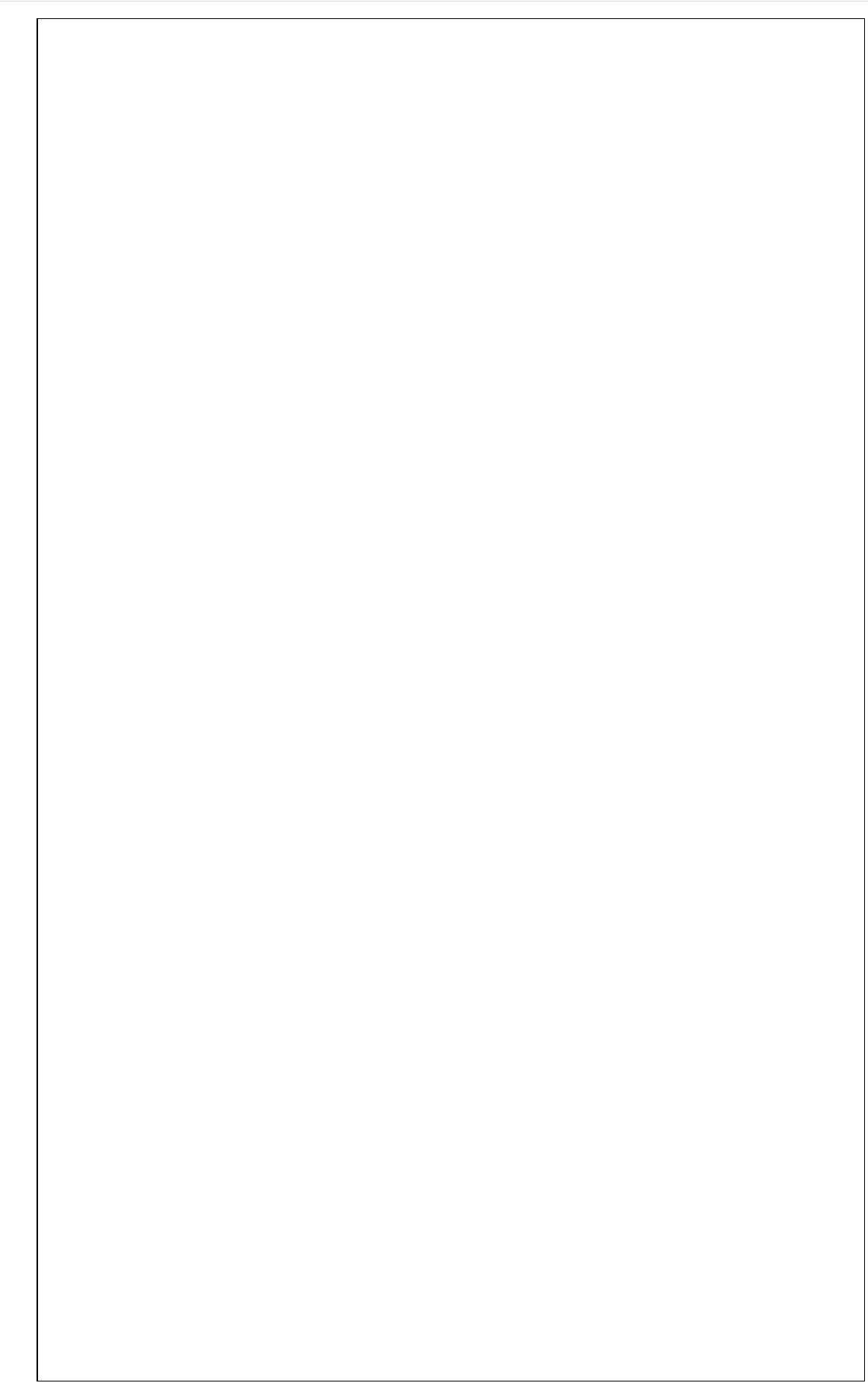
**}**

**catch(string)**

**{**

**29 | P a g e** **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**S E - C o m p - A** **O b j e c t O r i e n t e d P r o g r a m m i n g L a b**



**cout<<"\nError: Title below 3 characters is not allowed"; title="";**

**}**

**try**

**{**

**if(price<=0.0)**

**throw price;**

**cout<<"\nPrice is :"<<price;**

**}**

**catch (float f)**

**{**

**cout<<"\nError: Price not valid: \t"<<f;**

**price=0.0;**

**}**

**}**

**private:**

**string title;**

**float price;**

**};**

**class Book: public Publication**

**{**

**public:**

**Book (): Publication ()**

**{**

**pages=0;**

**}**

**Book (string title, float price, int pages): Publication (title,price)**

**{**

**this->pages=pages;**

**}**

**void getData()**

**{**

**Publication:: getData ();**

**cout<<"\nEnter no. of pages in book\n";**

**cin>>pages;**

**}**

**void putData()**

**{**

**Publication::putData();**

**try**

**{**

**if(pages<0)**

**throw pages;**

**cout<<"\nPages are :"<<pages;**

**}**

**catch(int f)**

**{**

**cout<<"\nError: Pages not valid: \t"<<f;**

**pages=0;**

**}**

**}**

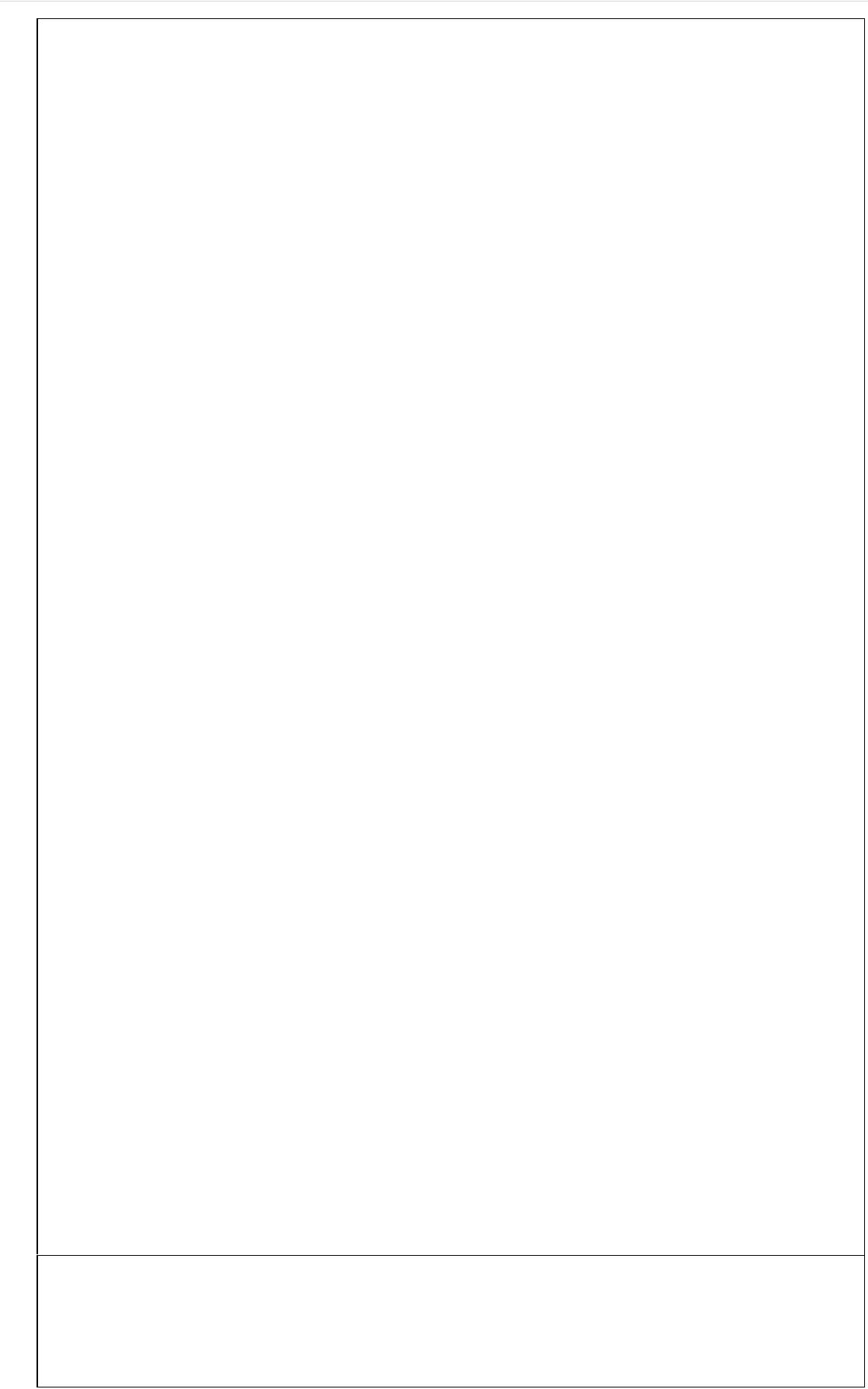
**private:**

**int pages;**

**};**

**30 | P a g e** **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**S E - C o m p - A** **O b j e c t O r i e n t e d P r o g r a m m i n g L a b**



**class Tape\_Cassette: public Publication**

**{**

**public:**

**Tape\_Cassette (): Publication ()**

**{**

**playtime=0.0;**

**}**

**Tape\_Cassette(string title, float price, float playtime):Publication(title,price)**

**{**

**this->playtime=playtime;**

**}**

**void getData()**

**{**

**Publication::getData();**

**cout<<"\nEnter play time of cassette\n";**

**cin>>playtime;**

**}**

**void putData()**

**{**

**Publication::putData();**

**try**

**{**

**if(playtime<0.0)**

**throw playtime;**

**cout<<"\nPlaytime is :"<<playtime;**

**}**

**catch(float f)**

**{**

**cout<<"\nError: Playtime not valid: \t"<<f;**

**playtime=0.0;**

**}**

**}**

**private:**

**float playtime;**

**};**

**int main ()**

**{**

**Book book;**

**cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BOOK\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**book.getData();**

**cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CASSETTE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**Tape\_Cassette cassette;**

**cassette.getData();**

**cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BOOK\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**book.putData();**

**cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CASSETTE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**cassette.putData();**

**return 0;**

**}**

**Output:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BOOK\*\*\*\*\*\*\*\*\*\*\*\*\*\***

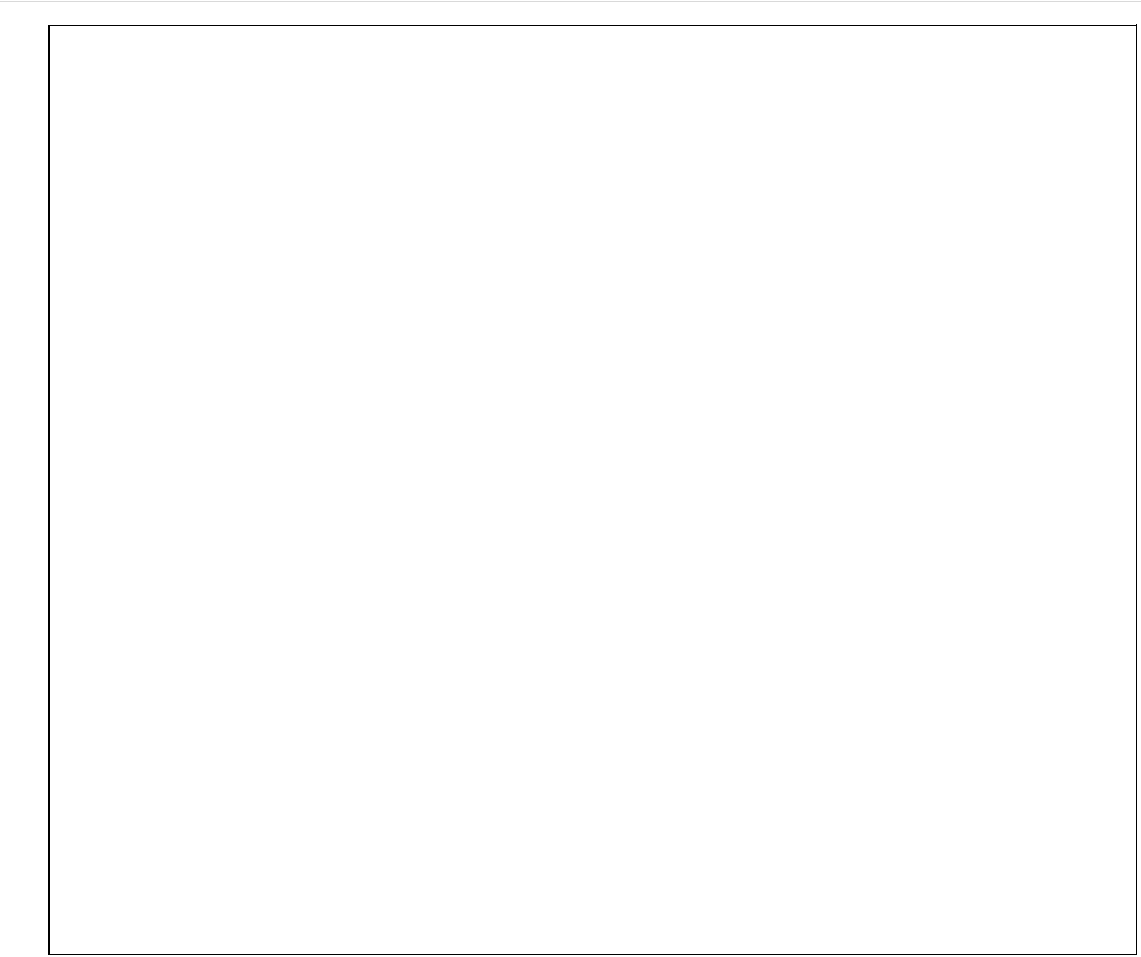
**Enter title and price c++ 111.44**

**Enter no. of pages in book 234**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CASSETTE\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**31 | P a g e** **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**S E - C o m p - A** **O b j e c t O r i e n t e d P r o g r a m m i n g L a b**



**Enter title and price ddlj 100**

**Enter play time of cassette 23.4**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BOOK\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Title is :c++**

**Price is :111.44**

**Pages are :234**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CASSETTE\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Title is :ddlj**

**Price is :100**

**Playtime is :23.4**

**OUTPUT 2:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BOOK\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Enter title and price C 100**

**Enter no. of pages in book -34**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CASSETTE\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Enter title and price DDLJ 100**

**Enter play time of cassette 23.4**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BOOK\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Error: Title below 3 characters is not allowed**

**Title is :**

**Price is :100**

**Error: Pages not valid: -34**

**Pages are :0**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CASSETTE\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Error: Price not valid: -100**

**Title is :DDLJ**

**Price is :0**

**Playtime is :23.4**

1. **Conclusion:**

**From above experiment, we have studied & implemented the concept of Inheritance and Exception Handling.**

1. **Review Questions & Exercises:**
   1. **Fill in the Blanks**
      1. **An overridden base class function may be called by a function in a derived class by using the scope resolution(::) operator.**
      2. **A(n) virtual member function in a base class expects to be overridden in a derived class.**
      3. **A(n) virtual member function in a base class expects to be overridden in a derived class.**
         1. **The line containing a throw statement is known as the throwpoint.**
         2. **The Try block contains code that directly or indirectly might cause an exception to be thrown. And Catch block handles an exception.**
   2. **Answer the following:**
      1. **What is the difference between a protected class member and a private class member?**

**Private class members are hidden from outside world. The perivate members member implement the oop concept of data hiding. Private member can be used by the member functions of the class in which it is declared. Private members cannot be inherited by other classes.**

**Eg:-  
class example  
{  
 private:**

**int a;  
}**

**Whereas protected members are the members that can only be used by the member functions and friends of the class in which it is declared. The protected members cannot be accessed by non member function. Protected members can be inherited by other classes**

**Eg:-**

**Class example**

**{  
 protected:**

**int b;**

**}**

* + 1. **Draw and explain multiple inheritance with suitable example.**

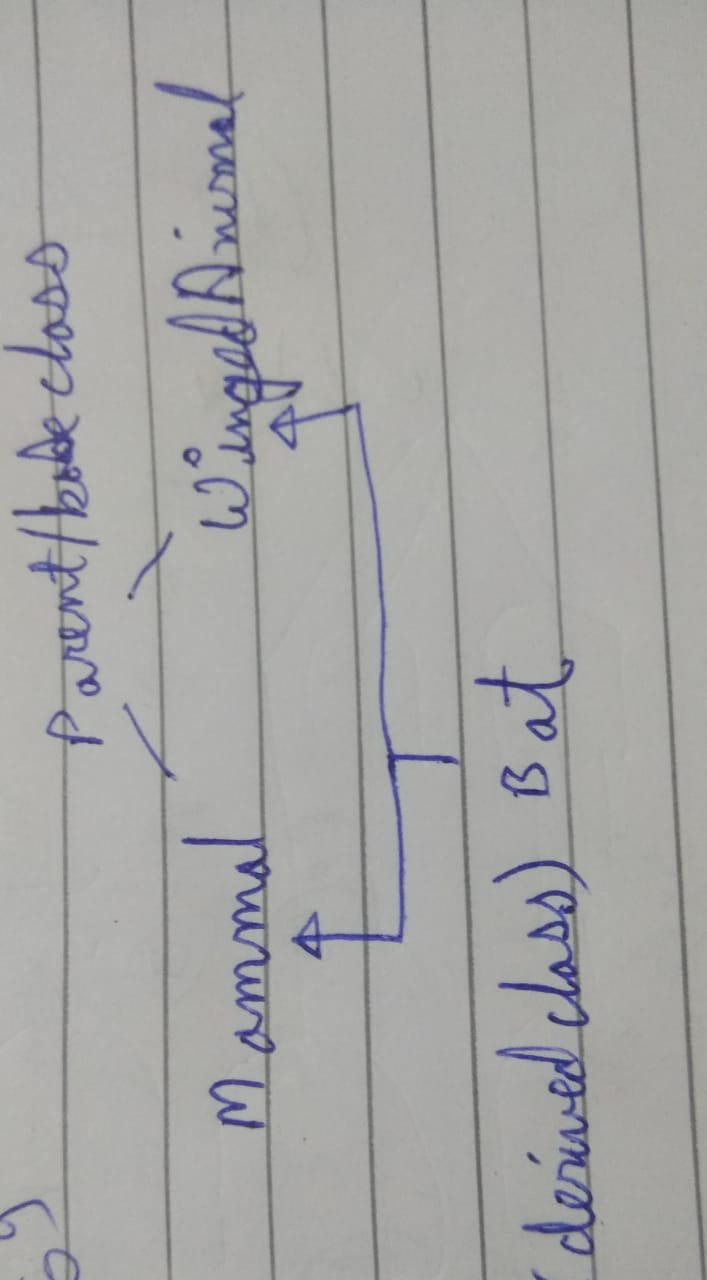
**In c++ , a class can be derived from more than one parents. For example if we create a class bat which is derived from mammal and wingedAnimal, it makes sense as a bat is mammal as well as a winged animal.**

**Eg:**

**Class mammal{};**

**Class wingedAnimal{};**

**Class bat:public mammal,public wingedAnimal{};**

****

* + 1. **Describe the concept of virtual base class with suitable example.**

**When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class’ name with the word virtual**

**class A   
{   
   public:   
       int i;   
};  
  
class B : virtual public A   
{   
   public:   
       int j;   
};  
  
class C: virtual public A   
{   
   public:   
       int k;   
};  
  
class D: public B, public C   
{   
   public:   
       int sum;   
};  
  
int main()   
{   
   D ob;   
   ob.i = 10; //unambiguous since only one copy of i is inherited.   
   ob.j = 20;   
   ob.k = 30;   
   ob.sum = ob.i + ob.j + ob.k;   
   cout << “Value of i is : ”<< ob.i<<”\n”;   
   cout << “Value of j is : ”<< ob.j<<”\n”; cout << “Value of k is :”<< ob.k<<”\n”;   
   cout << “Sum is : ”<< ob.sum <<”\n”;   
  
   return 0;   
}**

* + 1. **Explain the difference between a try block and a catch block.**

**We use try catch block when there is chance of getting error(may be due to bad user input). We write code that we want to execute in try block and if while executing if it get error or some exception then the code written in catch block is executed which prevent the program for getting crashed.**

**Eg:**

**Int a,int b;**

**Cout<<”Enter number 1:”;**

**Cin>>a;**

**Cout<<”Enter number 2:”;**

**Cin>>b;**

**Try{**

**Int d=a/b;**

**Cout<<”Result of number1/number2 is:”<<d<<endl;**

**} catch(const char\* e){**

**Cout<<”Bad inputs. Try re-run program”<<endl;**

**}**

**Cout<<”the program execution continue”;**

**In this case if user enter number 2 as 0 then it will throw an exception divide by zero if we would not use try catch the program would abruptly ended. But now it will continue execution after bad inputs also.**

* + 1. **Explain different kinds of errors.**

**types of errors:**

1. **Syntax errors: errors due to the fact that the syntax of the language is not respected.**
2. **Semantic errors: errors due to an improper use of program statements.**
3. **Logical errors: errors due to the fact that the specification is not respected.**

**From the point of view of when errors are detected, we distinguish:**

1. **Compile time errors: syntax errors and static semantic errors indicated by the compiler.**
2. **Runtime errors: dynamic semantic errors, and logical errors, that cannot be detected by the compiler (debugging).**

**32 | P a g e** **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**S E - C o m p - A** **O b j e c t O r i e n t e d P r o g r a m m i n g L a b**

1. **Algorithm Assignments:**
   1. **Write the first line of the declaration for a SoundSystem class. Use multiple inheritance to base the class on the CDplayer class, the Tuner class, and the CassettePlayer class. Use public base class access in all cases.**

**Class SoundSystem : public CDplayer, public Turner, public CassettePlayer**

* 1. **Write an exception handling code which can check the following exception in inputed date(you have to use multiple try-catch blocks):**
     1. **Invalid Day Throw when an invalid day (< 1 or > 31) is passed to the class.**
     2. **Invalid Month Throw when an invalid month (< 1 or > 12) is passed to the class.**

**int day,month;**

**try{**

**cout<<"Enter date:";**

**cin>>day;**

**if(day>31 || day<1)**

**throw "Invalid Date";**

**}**

**catch(char const\* s){**

**cout<<s<<endl;**

**}**

**try{**

**cout<<"\nEnter month:";**

**cin>>month;**

**if(month>12 || month<1)**

**throw "Invalid Month";**

**cout<<"OK month";**

**}**

**catch(char const\* st){**

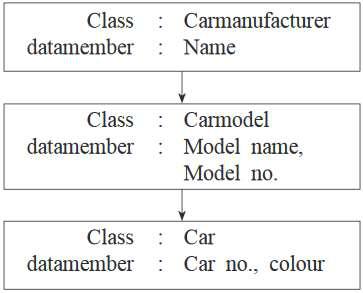
**cout<<st<<endl;**

**}**

1. **Programming Assignments:**
   1. **Design a class named Course with two data members: name and units. Then**

**design a class named Student with three data members: name, gpa, and a list of courses taken. The list must be implemented as an array in heap memory. Create constructors, destructor, and all necessary member functions for the operation of the Course and Student class. Test both classes in an application program.**

* 1. **Write C++ program for following multilevel inheritance.:**



**Accept and display data for one car with all details.**

1. **Write a C++ program to check the following conditions and throw the exception if the criterion does not meet.**
   1. **User has age between 18 and 55**
   2. **User stays has income between Rs. 50,000 – Rs. 1,00,000 per month**
   3. **User stays in Pune/ Mumbai/ Bangalore / Chennai**
   4. **User has 4-wheeler**

**Accept age, Income, City, Vehicle from the user and check for the conditions mentioned above. If any of the condition not met then throw the exception.**

1. **References:**
   1. **E Balagurusamy Object-Oriented Programming with C++.7th edition. McGraw-Hill Publication, ISBN 10: 9352607996 ISBN 13: 9789352607990**
   2. **Tony Gaddis- “STARTING OUT WITH C++ From Control Structures through Objects”, Pearson Education, ISBN 13: 978-0-13-376939-5**

**33 | P a g e** **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**