|  |  |  |
| --- | --- | --- |
| **Rubrics for Object Oriented Programming Lab** | | |
| **Lab #:** | **06** | |
| **Lab Title:** | **Inheritance** | |
| **Submitted by:** | | |
| **Name** | | **Registration #** |
| **AMMAR**  **MUHAMMAD KALEEM ULLAH** | | **FA19-BCE-001**  **FA19-BCE-007** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Rubrics name & number** | | | **Marks** | | |
| **In-Lab** | | **Post-Lab** |
| **Engineering Knowledge** | ***R2: Use of Engineering Knowledge and follow Experiment Procedures:***  *Ability to follow experimental procedures, control variables, and record procedural steps on lab report.* | |  | | |
| **Problem Analysis** | | ***R5: Data/Evidence Measurements:***  *Ability to record raw data / evidence.* | |  | |
| **Design** | | ***R8: Best Coding Standards:***  *Ability to follow the coding standards and programming practices.* | |  | |
| **Modern Tools Usage** | | ***R9: Understand Tools:*** *Ability to describe and explain the principles behind and applicability of engineering tools.* | |  | |
| **Individual and Teamwork** | | ***R12: Individual Work Contributions:*** *Ability to carry out individual responsibilities.* | |  | |
| ***R13: Management of Team Work:***  *Ability to appreciate, understand and work with multidisciplinary team members.* | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rubrics #** | R2 | R5 | R8 | R9 | R12 | R13 |
| **In –Lab** |  |  |  |  |  |  |
| **Post- Lab** |  |  |  |  |  |  |

**Lab#06**

**Inheritance**

1. **Objectives:**

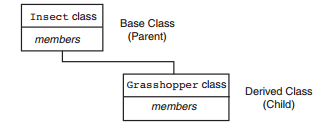
Objectives of the lab is to familiarize the student with various concepts and terminologies of inheritance in object-oriented Programming.

1. **Introduction:**
   1. **Inheritance:**

Inheritance is a way of creating a new class by starting with an existing class and adding new members. It allows a new class to be based on an existing class. The new class can replace or extend the functionality of the existing class. The existing class is called the **base class** and the new class is called the **derived class**. The new class inherits all the member variables and functions (except the constructors and destructor) of the class it is based on.

* 1. **Inheritance and the “Is a” Relationship:**

When an “is a” relationship exists between classes, it means that the specialized class has all of the characteristics of the general class, plus additional characteristics that make it special. In object-oriented programming, inheritance is used to create an “is a” relationship between classes.



* 1. **Protected Members and Class Access:**

Protected members of a base class are like private members, but they may be accessed by derived classes. The base class access specification determines how private, public, and protected base class members are accessed when they are inherited by the derived classes.

* 1. **Derived class constructor**

Constructors are not inherited, even though they have public visibility. However, the super reference can be used within the child's constructor to call the parent's constructor. In that case, the call to parent's constructor must be the first statement.



1. **In-Lab Tasks:**
   1. **Task#01**

Imagine a publishing company that markets both book and audiocassette versions of its works. 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). Each of these three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata() function to display its data.  
Write a main() program to test the book and tape classes by creating instances of them, asking the user to fill in data with getdata() , and then displaying the data with putdata() .

* **Code:**

#include<iostream>

#include<string>

using namespace std;

class publication

{

private:

string title;

float price;

public:

void getData()

{

cin.ignore();

cout<<"\nEnter The Title of the Book : "; getline(cin,title,'\n');

cout<<"\nEnter the Price of the Book : "; cin>>price;

}

void putData()const

{

cout<<"\nTitle of the book is : "<<title;

cout<<"\nPrice of the book is : "<<price;

}

};

class sales

{

private:

enum { MONTHS = 3 };

float salesArr[MONTHS];

public:

void getData();

void putData() const;

};

void sales::getData()

{

cout << " Enter sales for 3 months\n";

for(int j=0; j<MONTHS; j++)

{

cout << " Month " << j+1 << " : ";

cin >> salesArr[j];

}

}

void sales::putData() const

{

for(int j=0; j<MONTHS; j++)

{

cout << "\n Sales for month " << j+1 << ": ";

cout << salesArr[j];

}

}

class book:public publication,public sales

{

private:

int pageCount;

public:

void getData()

{

publication::getData();

cout<<"\nEnter the Number of Pages of the Book : "; cin>>pageCount;

sales::getData();

}

void putData()const

{

publication::putData();

cout<<"\nThe Number of pages of book are : "<<pageCount;

sales::putData();

}

};

class tape:public publication,public sales

{

private:

float minutes;

public:

void getData()

{

publication::getData();

cout<<"\nEnter the Audio Minute of the Book : "; cin>>minutes;

sales::getData();

}

void putData()

{

publication::putData();

cout<<"\nThe Number of Minute of this Audio Book are : "<<minutes;

sales::putData();

}

};

int main()

{

int n;

cout<<"\nHow many Book's Data You Want To Enter : "; cin>>n;

book b[n];

tape t[n];

for(int i=0;i<n;i++)

{

cout<<"\nEnter the Books Data : ";

b[i].getData();

cout<<endl;

cout<<"\nEnter the Tape Data : ";

t[i].getData();

}

system("CLS");

for(int i=0;i<n;i++)

{

cout<<endl;

cout<<"\nStore Data : ";

cout<<"\nBook information: ";

cout<<endl;

b[i].putData();

cout<<"Tape Information : \n ";

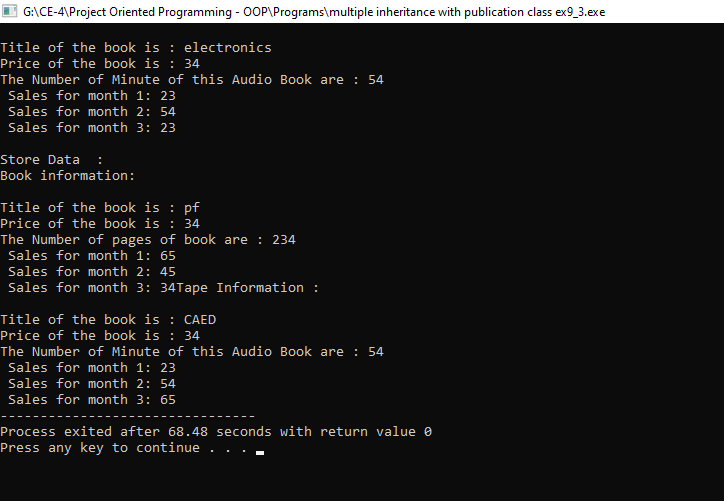
t[i].putData();

}

return 0;

}

* **Output:**



* 1. **Task#02**

Assume that the publisher in task 1 decides to add a third way to distribute  
books: on computer disk, for those who like to do their reading on their laptop. Add a disk class that, like book and tape, is derived from publication. The disk class should incorporate the same member functions as the other classes. The data item unique to this class is the disk type: either CD or DVD. You can use an enum type to store this item. The user could select the appropriate type by typing c or d.

* **Code:**

#include<iostream>

#include<string>

using namespace std;

class publication

{

private:

string title;

float price;

public:

void getData()

{

cin.ignore();

cout<<"\nEnter The Title of the Book : "; getline(cin,title,'\n');

cout<<"\nEnter the Price of the Book : "; cin>>price;

}

void putData()const

{

cout<<"\nTitle of the book is : "<<title;

cout<<"\nPrice of the book is : "<<price;

}

};

class sales

{

private:

enum { MONTHS = 3 };

float salesArr[MONTHS];

public:

void getData();

void putData() const;

};

void sales::getData()

{

cout << " Enter sales for 3 months\n";

for(int j=0; j<MONTHS; j++)

{

cout << " Month " << j+1 << " : ";

cin >> salesArr[j];

}

}

void sales::putData() const

{

for(int j=0; j<MONTHS; j++)

{

cout << "\n Sales for month " << j+1 << ": ";

cout << salesArr[j];

}

}

class book:public publication,public sales

{

private:

int pageCount;

public:

void getData()

{

publication::getData();

cout<<"\nEnter the Number of Pages of the Book : "; cin>>pageCount;

sales::getData();

}

void putData()const

{

publication::putData();

cout<<"\nThe Number of pages of book are : "<<pageCount;

sales::putData();

}

};

class tape:public publication,public sales

{

private:

float minutes;

public:

void getData()

{

publication::getData();

cout<<"\nEnter the Audio Minute of the Book : "; cin>>minutes;

sales::getData();

}

void putData()

{

publication::putData();

cout<<"\nThe Number of Minute of this Audio Book are : "<<minutes;

sales::putData();

}

};

class disk:public publication,public sales

{

private:

enum Disk{cd, dvd};

Disk e;

char ch;

public:

void getData()

{

cout<<"Enter 'c' to select CD or 'd' for DVD : ";

cin>>ch;

if(ch == 'c')

{

e=cd;

}

if(ch == 'd')

{

e=dvd;

}

}

void putData()

{

if(ch == 'c')

{

cout<<"CD";

}

if(ch == 'd')

{

cout<<"DVD";

}

}

};

int main()

{

int n;

cout<<"\nHow many Book's Data You Want To Enter : "; cin>>n;

book b[n];

tape t[n];

disk d[n];

for(int i=0;i<n;i++)

{

cout<<"\nEnter the Books Data : ";

b[i].getData();

cout<<endl;

cout<<"\nEnter the Tape Data : ";

t[i].getData();

cout<<"\nEnter the Disk Data : ";

d[i].getData();

}

system("CLS");

for(int i=0;i<n;i++)

{

cout<<endl;

cout<<"\nStore Data : ";

cout<<"\nBook information: ";

cout<<endl;

b[i].putData();

cout<<"Tape Information : \n ";

t[i].putData();

cout<<"\nDisk Data : ";

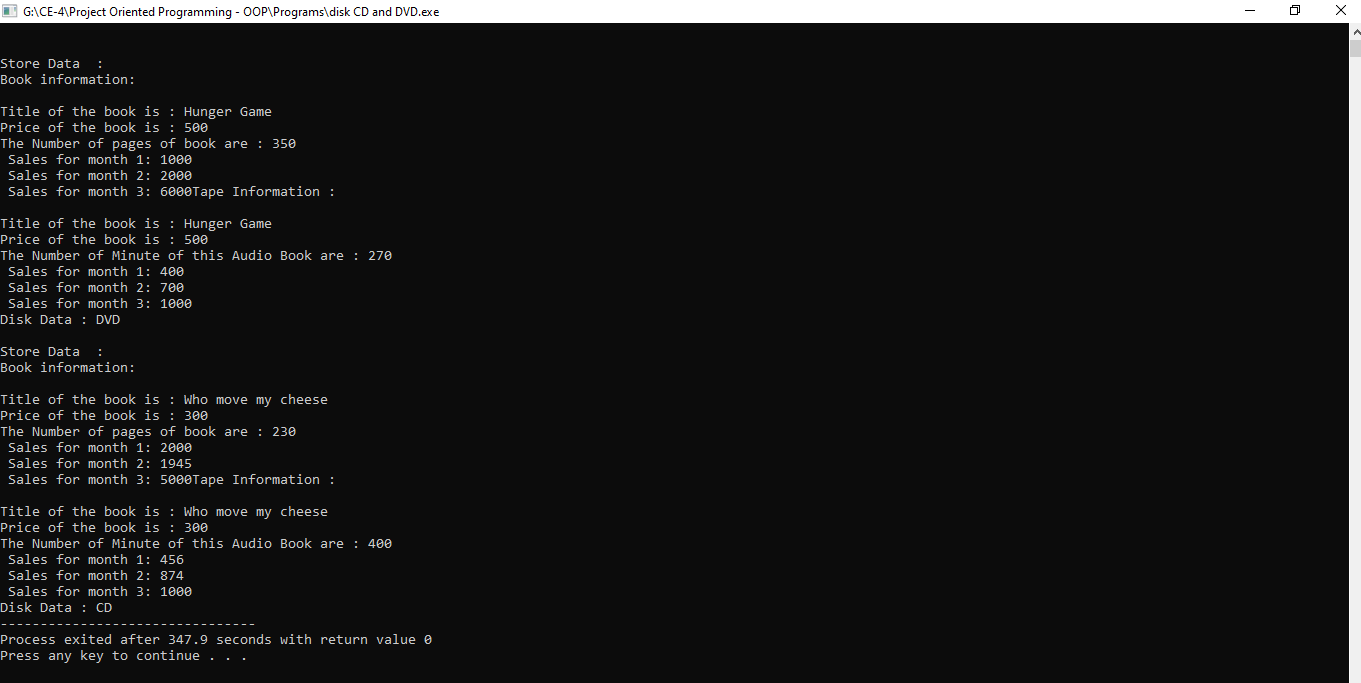
d[i].putData();

}

return 0;

}

* **Output:**

****

* 1. **Task#03**

Design a class namedPersonData with the following member variables:

* lastName
* firstName
* address
* city
* state
* zip
* phone

Write the appropriate accessor and mutator functions for these member variables.

Next, design a class named CustomerData, which is derived from the PersonData

class. The CustomerData class should have the following member variables:

* customerNumber
* mailingList

The customerNumber variable will be used to hold a unique integer for each customer.

The mailingList variable should be a bool. It will be set to true if the customer

wishes to be on a mailing list, or false if the customer does not wish to be on a mailing list. Write appropriate accessor and mutator functions for these member variables.

Demonstrate an object of the CustomerData class in a simple program.

* **Code:**

#include<iostream>

#include<string>

using namespace std;

class PersonData

{

private:

string lastName;

string firstName;

string address;

string city;

string state;

string zip;

string phone;

public:

void getData()

{

cout<<"\nEnter the First Name : "; getline(cin,firstName,'\n');

cout<<"\nEnter the Last Name : "; getline(cin,lastName,'\n');

cin.ignore();

cout<<"\nEnter the Address : "; getline(cin,address,'\n');

cin.ignore();

cout<<"\nEnter the City : "; getline(cin,city,'\n');

cin.ignore();

cout<<"\nEnter the State : "; getline(cin,state,'\n');

cin.ignore();

cout<<"\nEnter the Zip Code : "; getline(cin,zip,'\n');

cin.ignore();

cout<<"\nEnter the Phone Number : "; getline(cin,phone,'\n');

cin.ignore();

}

void putData()

{

cout<<"\nYour First Name : "<<firstName;

cout<<"\nYour Last Name : "<<lastName;

cout<<"\nYour Address : "<<address;

cout<<"\nYour City : "<<city;

cout<<"\nYour State : "<<state;

cout<<"\nYour Zip Code : "<<zip;

cout<<"\nYour Phone Number : "<<phone;

}

};

class CustomerData:public PersonData

{

private:

int customerNumber;

int mailingList;

public:

void getData()

{

PersonData::getData();

cout<<"\n Please enter Customer Nubmer : "; cin>>customerNumber ;

cout<<"\n Mailing List Service Preferacne : ";

cout<<"\n Do you want to get add in mailing list : ";

cout<<"\n if No, Enter any key, if Yes, Enter '1'"; cin>>mailingList;

}

void putData()

{

cout<<"\nYour Customer Service Number : "<<customerNumber;

PersonData::putData();

if(mailingList==1)

{

cout<<"\n\t You Have Subscribed For Mailing Service";

}else

{

cout<<"\n\t You Haven't Subscribed For Mailing Service";

}

}

};

int main()

{

CustomerData c1;

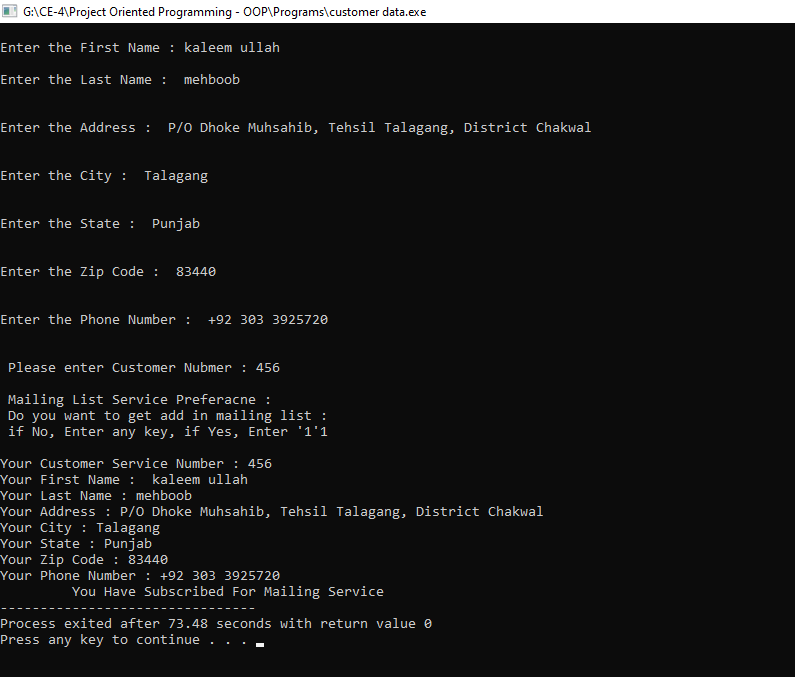
c1.getData();

c1.putData();

return 0;

}

* **Output:**



1. **Post-Lab Tasks:**
   1. **Task#01**

Design a class named Employee. The class should keep the following information in  
•Employee name  
•Employee number  
•Hire date  
Write one or more constructors and the appropriate accessor and mutator functions  
for the class. Next, write a class named ProductionWorker that is derived from the Employee class.  
The ProductionWorker class should have member variables to hold the following  
information:  
•Shift (an integer)  
•Hourly pay rate (a double )  
The workday is divided into two shifts: day and night. The shift variable will hold an  
integer value representing the shift that the employee works. The day shift is shift 1, and  
the night shift is shift 2. Write one or more constructors and the appropriate accessor  
and mutator functions for the class. Demonstrate the classes by writing a program that  
uses a ProductionWorker object.

**Hint:** use **to\_string** function for converting numeric date to string.

* **Code:**

#include<iostream>

#include<string>

using namespace std;

class Employee

{

protected:

string eName;

int eNumber,date,month,year;

public:

void getData()

{

cin.ignore();

cout<<"\nEnter the Employee Name : "; getline(cin,eName,'\n');

cout<<"\nEnter the Employee Number : ";cin>>eNumber;

}

void putData()const

{

cout<<"\nEmployee Name : "<<eName;

cout<<"\nEmployee Number : "<<eNumber;

}

};

class hireDate:public Employee

{

public:

void getData()

{

cout<<"Enter Your Hiring Date >> ";

cout<<"\nEnter the Data of Month : ";cin>>date;

cout<<"\nEnter the Month : ";cin>>month;

cout<<"\nEnter the Year : ";cin>>year;

}

void putData()const

{

cout<<"\nHiring Date is : "<<date<<"\\"<<month<<"\\"<<year;

}

};

class ProductionWorker:public Employee

{

private:

int shift;

double hourPerRate;

hireDate hd;

public:

ProductionWorker()

{

shift=0;

hourPerRate=0;

}

void getData()

{

Employee::getData();

hd.getData();

cout<<"\nFor day shift Enter '1'";

cout<<"\nFor night shift Enter '2' ";

cout<<"\nEnter the Shift : ";cin>>shift;

cout<<"\nEnter the Hour Per Rate : ";cin>>hourPerRate;

}

void convertInToText()

{

if (shift == 1)

{

cout<<"\nDay Shift";

}

else if(shift ==2)

{

cout<<"\nNight Shift";

}

else

{

cout<<"\nInvalid Choice For Shift !!";

}

}

void putData()

{

Employee::putData();

hd.putData();

convertInToText();

cout<<"\nHourly pay rate : "<<hourPerRate;

}

};

int main()

{

int n;

cout<<"Enter the Number Of Employee You Want To Enter : ";cin>>n;

ProductionWorker pw[n];

cout<<"Getting Data of Employee >> ";

for(int i=0;i<n;i++)

{

pw[i].getData();

}

cout<<"Stored Data of Employee >> ";

for(int i=0;i<n;i++)

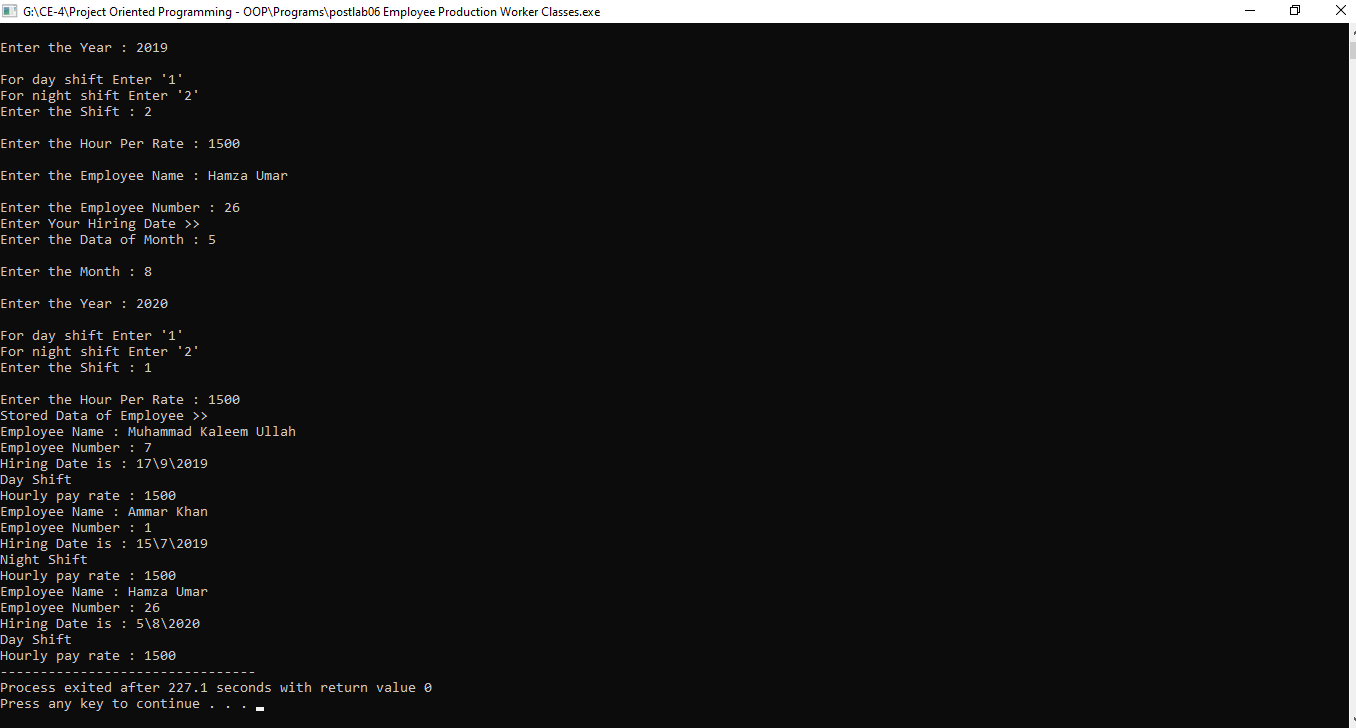
{

pw[i].putData();

}

}

* **Output:**

****

1. **Conclusion:**

After completing lab, we are able to:

* Declare the derived classes along with the access of base class members.
* Know the purpose of protected members and class access as well as working with derived class constructors.