EXPERIMENT.NO.4

Aim: Identify and implement different classes and member function using inheritance for given statement(Laptop Configuration System).

Objectives:-

1. To understand the concept of inheritance
2. To implement different types of inheritance.

Theory:-

Inheritance

Inheritance is the process by which objects can acquire the properties of objects of other class. In OOP, inheritance provides reusability, like, adding additional features to an existing class without modifying it. This is achieved by deriving a new class from the existing one. The new class will have combined features of both the classes.

Types of Inheritance

1.Single inheritance :

A derived class with only one base class, is called **single inheritance.** A derived class can he defined by specifying its relationship with the base class in addition to its own details. The general form of define a derived class is:

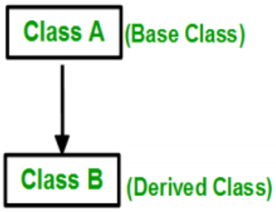
Class derived_class_nae : visibility_mode base_class_name

{

--

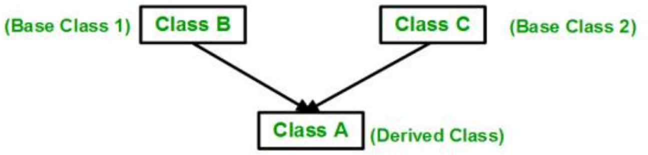
--

};

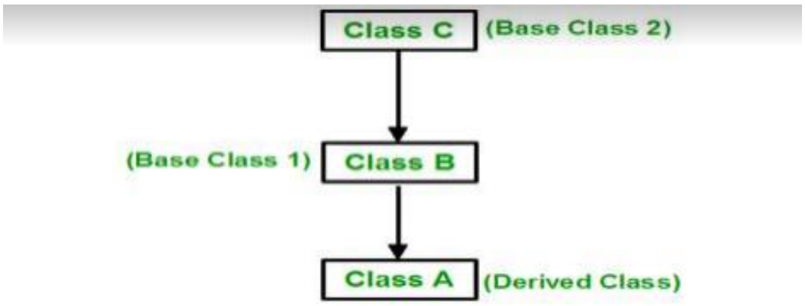


2. Multiple inheritance:

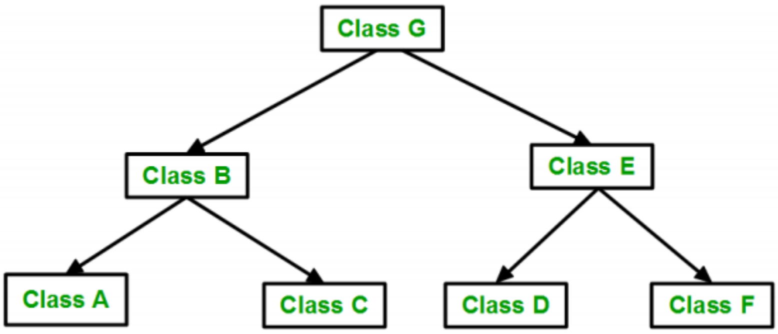
A class can inherit the attributes of two or more classes. This is known as **multiple inheritance**. Multiple inheritance allows us to combine the features of several existing classes as a starting point for defining new classes. It is like a child inheriting the physical features of one parent and the intelligence of another.



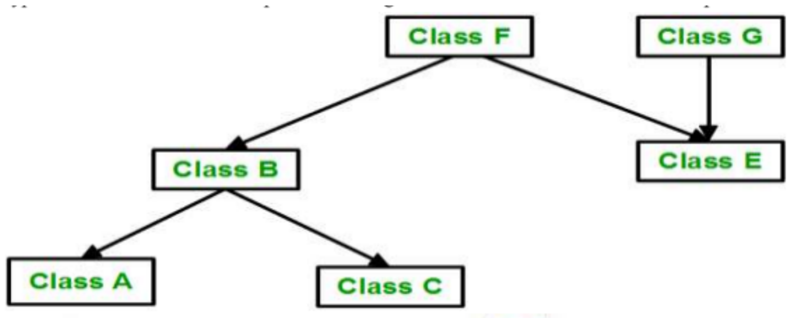
3. Multilevel inheritance: It is not uncommon that a class is derived from another derived class. The class a serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. The class B is known as intermediate base class since it provides a link for the inheritance between A and C. The chain ABC is known as inheritance path.



4. Hierarchical inheritance: In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.



5. Hybrid Inheritance: Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.



There are three types of class inheritance: **public, private** and protected the **protected** access specifier is similar to private. Its only difference occurs in fact with inheritance. When a class inherits from another one, the members of the derived class can access the protected members inherited from the base class, but not its private members.

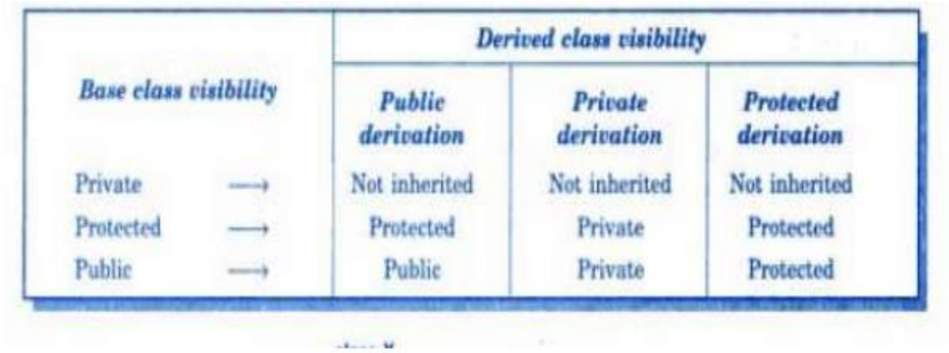
Difference between public, private and protected

• A member (either data member or member function) declared in a private section of a class can only be accessed by member functions and friends of that class

• A member (either data member or member function) declared in a protected section of a class can only be accessed by member functions and friends of that class, and by member functions and friends of derived classes

• A member (either data member or member function) declared in a public section of a class can be accessed by anyone

The following table indicates how the attributes are inherited in the three different types of inheritance:



Base class visibility	*Derived class visibility*		
	Public derivation	*Private derivation*	*Protected derivation*
Private ⟶	Not inherited	Not inherited	Not inherited
Protected ⟶	Protected	Private	Protected
Public ⟶	Public	Private	Protected

What a derived class inherits?

• Every data member defined in the parent class (although such members¤ may not always be accessible in the derived class!)

• Every ordinary member function of the parent class (although such members¤ may not always be accessible in the derived class!)

• The same initial data layout as the base class

What a derived class doesn't inherit?

• The base class's constructors and destructor

• The base class's assignment operator

• The base class's friends

What a derived class can add?

• New data members

• New member functions

• New constructors and destructor

• New friends

Eg suppose that we want to declare a series of classes that describe polygons like our C Rectangle, or like CTriangle. They have certain common properties, such as both can be described by means of only two sides: height and base. This could be represented in the world of classes with a class C Polygon from which we would derive the two other ones: C Rectangle and C Triangle.

The class CPolygon would contain members that are common for both types of polygon. In our case: width and height. And CRectangle and CTriangle would be its derived classes, with specific features that are different from one type of polygon to the other.

Classes that are derived from others inherit all the accessible members of the base class. That means that if a base class includes a member A and we derive it to another class with another member called B, the derived class will contain both members A and B. In order to derive a class from another, we use a colon (:) in the declaration of the derived class using the following format:

```
class derived_class_name: public base_class_name

{ /*...*/ };
```

Where derived_class_name is the name of the derived class and base_class_name is the name of the class on which it is based. The public access specifier may be replaced by any one of the other access specifiers protected and private. This access specifier describes the minimum access level for the members that are inherited from the base class.

```
// derived classes
#include <iostream>
using namespace std;
class CPolygon {
protected:
int width, height;
public:
void set_values (int a, int b)
{ width=a; height=b;}
};
class CRectangle: public CPolygon {
public:
int area ()
{ return (width * height); }
};
class CTriangle: public CPolygon {
public:
int area ()
{ return (width * height / 2); }
};
int main () {
CRectangle rect;
CTriangle trgl;
rect.set_values (4,5);
trgl.set_values (4,5);

cout << rect.area() << endl;

cout << trgl.area() << endl;

return 0;

} 20

10
```

The objects of the classes CRectangle and CTriangle each contain members inherited from CPolygon. These are: width, height and set_values().

Multiple inheritance

Allows you to create a derived class that inherits properties from more than one base class. Because a derived class inherits members from all its base classes, ambiguities can result. For example, if two base classes have a member with the same name, the derived class cannot implicitly differentiate between the two members.

In the following example, classes A, B, and C are direct base classes for the derived class X:

```
class A { /* ... */ };

class B { /* ... */ };

class C { /* ... */ };

class X : public A, private B, public C { /* ... */ };
```

Here in This project **Laptop Configuration System** we have used Single Inheritance.

Base class: User

Member functions:

- fullname();
- display();

Derived class: class Features

Member functions:

- select_os();
- select_processor();
- select_ram();
- display_features();

Source Code for Laptop Customization System using single Inheritance:

```
#include<iostream>

#include<stdio.h>


// Assignment-4


/* Here in this project we have used the single Inheritance

here base class is User and Derived class is Features */


using namespace std;


class User

{

private:

   string name;

   string lastname;

   string fn;



 public:

     User()//default constructor used

     {

         cout<<"\n * Enter the First name : ";

         cin>>name;

         cout<<"\n * Enter the lastname : ";

         cin>>lastname;

     }
     string fullname()
     {
        fn=name+" "+lastname;
        return fn;
     }

     void display()
    {
        cout<<"\n\n **Hello "<<fn<<" Welcome to our Program !!!!"<<endl;
    }

    ~User()//Destructor called
    {
        cout<<"\n\n User Destructed Successfully !!!!\n\n"<<endl;
    }
};


class Features : public User // Single Inheritance Used Class Feature Derived From Class User
{
   int os;
    int processor;
    int ram;
public:

    void select_os();
    void select_processor();

    void select_ram();

    void display_features();

};

void Features::select_os() // member function

{

    cout<<"\n ** Please Select The Operating system you want in your Laptop : \n"<<endl;

    cout<<"\n 1.Windows "<<endl;

    cout<<"\n 2.MAC Os "<<endl;

    cout<<"\n 3.Exit"<<endl;

    cout<<"\n Enter Your Choice : ";

    cin>>os;

    switch(os)

    {

    case 1:

        cout<<"\n * You have Selected Your Os ==> windows Os"<<endl;

        break;


    case 2:

        cout<<"\n * You have Selected Your Os ==> MAC Os"<<endl;

        break;


    case 3:

        cout<<"\n Terminated Successfully !!"<<endl;

        cout<<"\n Thank You Visit Again !!!!\n"<<endl;

        exit(0);

        break;
    default :
        cout<<"\n Enter valid choice ";
        exit(0);
    }
}
void Features::select_processor()
{
if(os==1){
    cout<<"\n\n **Select the Processor : \n"<<endl;
    cout<<"1. Intel i3"<<endl;
    cout<<"2. Intel i5"<<endl;
    cout<<"3. Intel i7"<<endl;
    cout<<"4. Intel i9"<<endl;
    cout<<"5. AMD Ryzen 5"<<endl;
    cout<<"6. AMD Ryzen 7"<<endl;
    cout<<"7. Exit"<<endl;
    cout<<"\n Enter Your Choice : ";
    cin>>processor;
    switch(processor)
    {
    case 1:

        cout<<"\n * You have Selected Your Processor ==> Intel i3 "<<endl;
        break;
    case 2:
```

```
    cout<<"\n * You have Selected Your Processor ==> Intel i5 "<<endl;
    break;
case 3:

    cout<<"\n * You have Selected Your Processor ==> Intel i7 "<<endl;
    break;
case 4:

    cout<<"\n * You have Selected Your Processor ==> Intel i9 "<<endl;
    break;
case 5:

    cout<<"\n * You have Selected Your Processor ==> AMD Ryzen 5 "<<endl;
    break;
case 6:

    cout<<"\n * You have Selected Your Processor ==> AMD Ryzen 7 "<<endl;
    break;
case 7:
    cout<<"\n Terminated Successfully !!"<<endl;
    cout<<"\n Thank You Visit Again !!!!\n"<<endl;
    exit(0);
    break;

default :
    cout<<"\n Enter valid choice ";
    exit(0);
```

```
}
}

else if(os==2)
{
cout<<"\n\n **Select the Processor : \n"<<endl;
cout<<"1. Intel i3"<<endl;
cout<<"2. Intel i5"<<endl;
cout<<"3. Intel i7"<<endl;
cout<<"4. Intel i9"<<endl;
cout<<"5. Exit"<<endl;
cout<<"\n Enter Your Choice : ";
cin>>processor;
switch(processor)
{
case 1:
   cout<<"\n * You have Selected Your Processor ==> Intel i3 "<<endl;
   break;

case 2:
   cout<<"\n * You have Selected Your Processor ==> Intel i5 "<<endl;
   break;

case 3:
   cout<<"\n * You have Selected Your Processor ==> Intel i7 "<<endl;
   break;
```

```
    case 4:
        cout<<"\n * You have Selected Your Processor ==> Intel i9 "<<endl;
        break;

    case 5:
        cout<<"\n Terminated Successfully !!"<<endl;
        cout<<"\n Thank You Visit Again !!!!\n"<<endl;
        exit(0);
        break;

    default :
        cout<<"\n Enter valid choice ";
        exit(0);
    }
    }
}

void Features::select_ram()
{

if(os==1){
    cout<<"\n\n **Select the RAM : \n"<<endl;
    cout<<"1. 4 GB"<<endl;
    cout<<"2. 8 GB"<<endl;
    cout<<"3. 16 GB"<<endl;
    cout<<"4. 32 GB"<<endl;
    cout<<"5. Exit"<<endl;
```

```
cout<<"\n Enter Your Choice : ";

cin>>ram;

switch(ram)

{

case 1:


    cout<<"\n * You have Selected Your RAM ==> 4 GB "<<endl;

    break;

case 2:


   cout<<"\n * You have Selected Your RAM ==> 8 GB "<<endl;

    break;

case 3:


    cout<<"\n * You have Selected Your RAM ==> 16 GB "<<endl;

    break;

case 4:


    cout<<"\n * You have Selected Your RAM ==> 32 GB "<<endl;

    break;

case 5:

    cout<<"\n Terminated Successfully !!"<<endl;

    cout<<"\n Thank You Visit Again !!!!\n"<<endl;

    exit(0);

    break;

default :

    cout<<"\n Enter valid choice ";
```

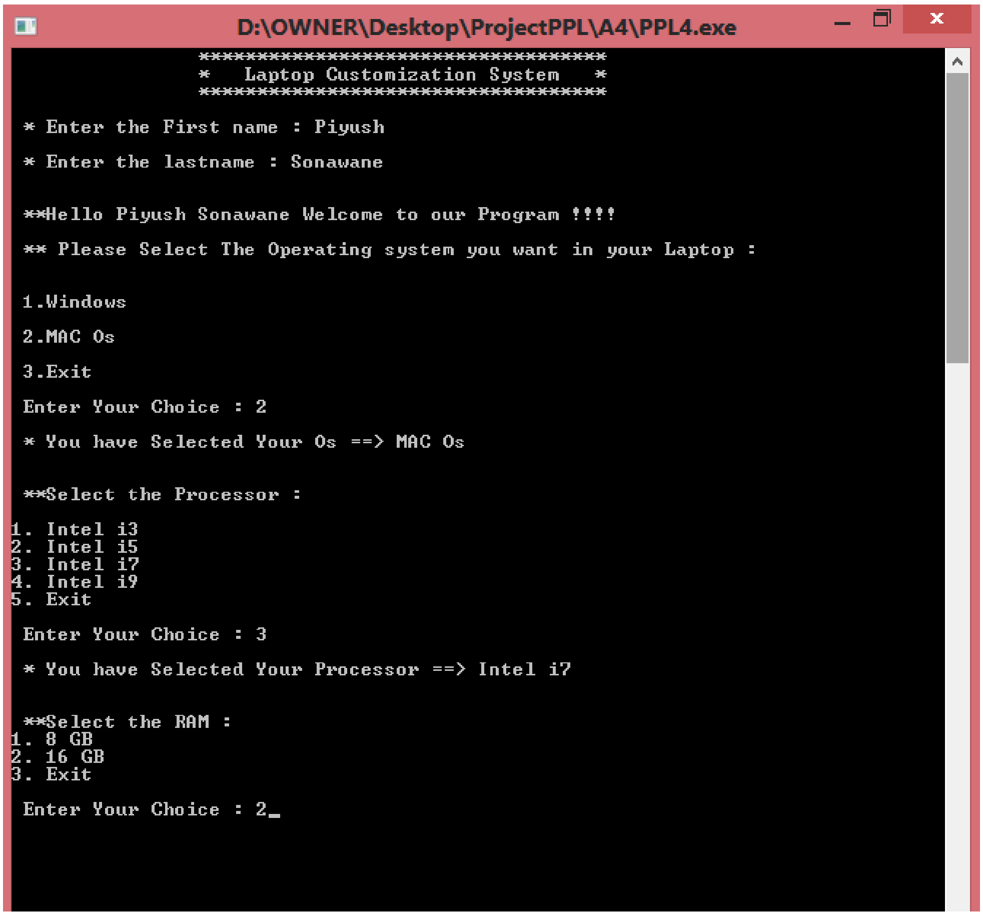
```
         exit(0);
    }
}

if(os==2)
{
    cout<<"\n\n **Select the RAM : "<<endl;
    cout<<"1. 8 GB"<<endl;
    cout<<"2. 16 GB"<<endl;
    cout<<"3. Exit"<<endl;
    cout<<"\n Enter Your Choice : ";
    cin>>ram;
    switch(ram)
    {
    case 1:

       cout<<"\n * You have Selected Your RAM ==> 8 GB "<<endl;
        break;
    case 2:

        cout<<"\n * You have Selected Your RAM ==> 16 GB "<<endl;
        break;

    case 3:
        cout<<"\n Terminated Successfully !!"<<endl;
        cout<<"\n Thank You Visit Again !!!!\n"<<endl;
        exit(0);
        break;
    default :
        cout<<"\n Enter valid choice ";
         exit(0);
    }

}
}



void Features::display_features()
{
    cout<<"\n\n";
    cout<<"\n * Your Selected Specifications are ==> \n"<<endl;

 //OS
    if(os==1){
        cout<<"\n Operating System ==> Windows Os"<<endl;



    //Processor
    if(processor==1)
        cout<<"\n Processor ==> Intel i3"<<endl;
    else if(processor==2)
        cout<<"\n Processor ==> Intel i5"<<endl;
    else if(processor==3)
        cout<<"\n Processor ==> Intel i7"<<endl;
    else if(processor==4)
        cout<<"\n Processor ==> Intel i9"<<endl;
    else if(processor==5)
        cout<<"\n Processor ==> Ryzen 5"<<endl;
    else if(processor==6)
        cout<<"\n Processor ==> Ryzen 7"<<endl;

     //RAM
    if(ram==1)
        cout<<"\n RAM ==> 4 GB"<<endl;
    else if(ram==2)
        cout<<"\n RAM ==> 8 GB"<<endl;
    else if(ram==3)
        cout<<"\n RAM ==> 16 GB"<<endl;
    else if(ram==4)
        cout<<"\n RAM ==> 32 GB"<<endl;
}
    //for mac
    else if(os==2){
        cout<<"\n Operating System ==> MAC Os "<<endl;
    if(processor==1)
        cout<<"\n Processor ==> Intel i3"<<endl;
     else if(processor==2)
        cout<<"\n Processor ==> Intel i5"<<endl;
     else if(processor==3)
        cout<<"\n Processor ==> Intel i7"<<endl;
     else if(processor==4)
```

```
        cout<<"\n Processor ==> Intel i9"<<endl;

     //ram mac
    if(ram==1)
      cout<<"\n RAM ==> 8 GB"<<endl;
    else if(ram==2)
      cout<<"\n RAM ==> 16 GB"<<endl;
    }
}
int main()
{
    cout<<"\t\t***********************************"<<endl;
    cout<<"\t\t*   Laptop Customization System   *"<<endl;
    cout<<"\t\t***********************************"<<endl;
    Features f;//object created
    f.fullname();
    f.display();
    f.select_os();
    f.select_processor();
    f.select_ram();
    system("cls");
    f.display_features();
    cout<<"\n\n * Thank you Visit Again :) !!!! "<<endl;
}
```

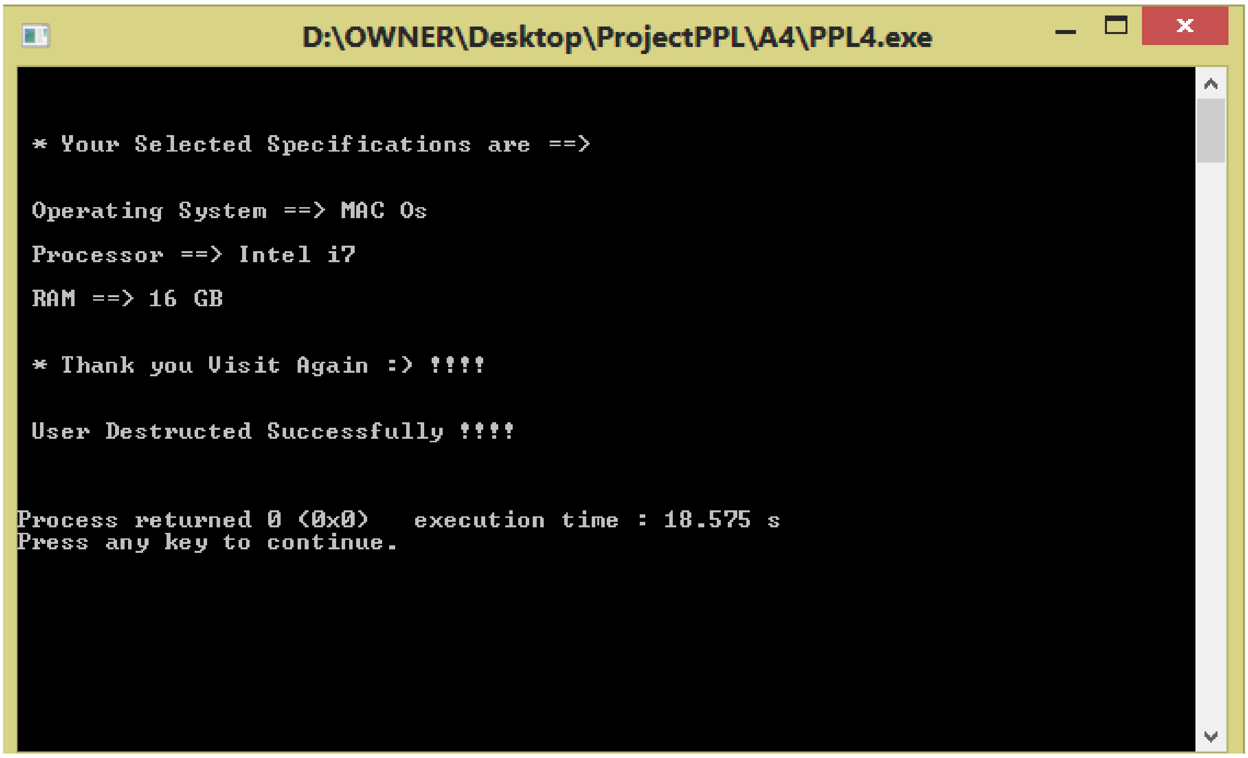
Output:

```
               ***********************************
               *       Laptop Customization System          *
               ***********************************


 * Enter the First name : Piyush
 * Enter the lastname : Sonawane
 **Hello Piyush Sonawane Welcome to our Program !!!!
 ** Please Select The Operating system you want in your Laptop :
 1.Windows
 2.MAC Os
 3.Exit
 Enter Your Choice : 2
 * You have Selected Your Os ==> MAC Os
 **Select the Processor :
1. Intel i3
2. Intel i5
3. Intel i7
4. Intel i9
5. Exit
 Enter Your Choice : 3
 * You have Selected Your Processor ==> Intel i7
 **Select the RAM :
1. 8 GB
2. 16 GB
3. Exit
 Enter Your Choice : 2
 * Your Selected Specifications are ==>

 Operating System ==> MAC Os

 Processor ==> Intel i7

 RAM ==> 16 GB

 * Thank you Visit Again :) !!!!


 User Destructed Successfully !!!!


Process returned 0 (0x0)   execution time : 145.025 s

Press any key to continue.
```

Screenshots:



```
                *************************************
                *    Laptop Customization System    *
                *************************************

 * Enter the First name : Piyush

 * Enter the lastname : Sonawane

 **Hello Piyush Sonawane Welcome to our Program !!!!

 ** Please Select The Operating system you want in your Laptop :

 1.Windows

 2.MAC Os

 3.Exit

 Enter Your Choice : 2

 * You have Selected Your Os ==> MAC Os

 **Select the Processor :

1. Intel i3
2. Intel i5
3. Intel i7
4. Intel i9
5. Exit

 Enter Your Choice : 3

 * You have Selected Your Processor ==> Intel i7

 **Select the RAM :
1. 8 GB
2. 16 GB
3. Exit

 Enter Your Choice : 2_
```



```
 * Your Selected Specifications are ==>

 Operating System ==> MAC Os

 Processor ==> Intel i7

 RAM ==> 16 GB

 * Thank you Visit Again :) !!!!

 User Destructed Successfully !!!!


Process returned 0 (0x0)   execution time : 18.575 s
Press any key to continue.
```

Conclusion: Thus studied and applied inheritance for Laptop Configuration system