**Name: Kaleem Ullah Reg. Number: FA19-BCE-007**

1. **In-Lab Tasks:**
   1. Write a program that declares two classes. The parent class is called Simple that has two data members num1 and num2to store two numbers. It also has four member functions.

* The add() function adds two numbers and displays the result.
* The sub() function subtracts two numbers and displays the result.
* The mul() function multiplies two numbers and displays the result.
* The div() function divides two numbers and displays the result.

The child class is called Complex that overrides all four functions. Each function in the child class checks the value of data members. It calls the corresponding member function in the parent class if the values are greater than 0. Otherwise, it displays error message.

**Solution:**

* **Code:**

#include<iostream>

using namespace std;

class Simple

{

protected:

int num1,num2;

int addition,subtraction,multiplication,division;

public:

void getData()

{

cout<<"\nEnter the Number 1 : ";cin>>num1;

cout<<"\nEnter the Number 2 : ";cin>>num2;

}

void add()

{

addition=num1+num2;

cout<<"The Addition i.e. Num1 + Num2 = "<<addition;

}

void sub()

{

subtraction=num1-num2;

cout<<"\nThe Subtraction i.e. Num1-Num2 = "<<subtraction;

}

void mul()

{

multiplication=num1\*num2;

cout<<"\nThe Multiplication i.e. Num1\*Num2 = "<<multiplication;

}

void div()

{

division=num1/num2;

cout<<"\nThe Division i.e. Num1/Num2 ="<<division;

}

};

class Complex:public Simple

{

public:

void add()

{

Simple::add();

}

void sub()

{

Simple::sub();

}

void mul()

{

if(num1==0||num2==0)

{

cout<<"\nerror !!! ";

}

else

{

Simple::mul();

}

}

void div()

{

if(num1==0||num2==0)

{

cout<<"\nerror !!! ";

}

else

{

Simple::div();

}

}

};

int main()

{

Complex c;

char n,choice;

cout<<"\nFor Addition Enter '+'";

cout<<"\nFor Subtraction Enter '-'";

cout<<"\nFor Multiplication Enter '\*'";

cout<<"\nFor Division Enter '/'";

do

{

c.getData();

cout<<"\nEnter the Operation You Want to Try : ";

cin>>n;

switch (n)

{

case '+':

{

c.add();

break;

}

case '-':

{

c.sub();

break;

}

case '\*':

{

c.mul();

break;

}

case '/':

{

c.div();

break;

}

}

cout<<"\nDo you wanna try this again ? ";

cout<<"\nEnter 'y' for yes and 'n' for no >> ";

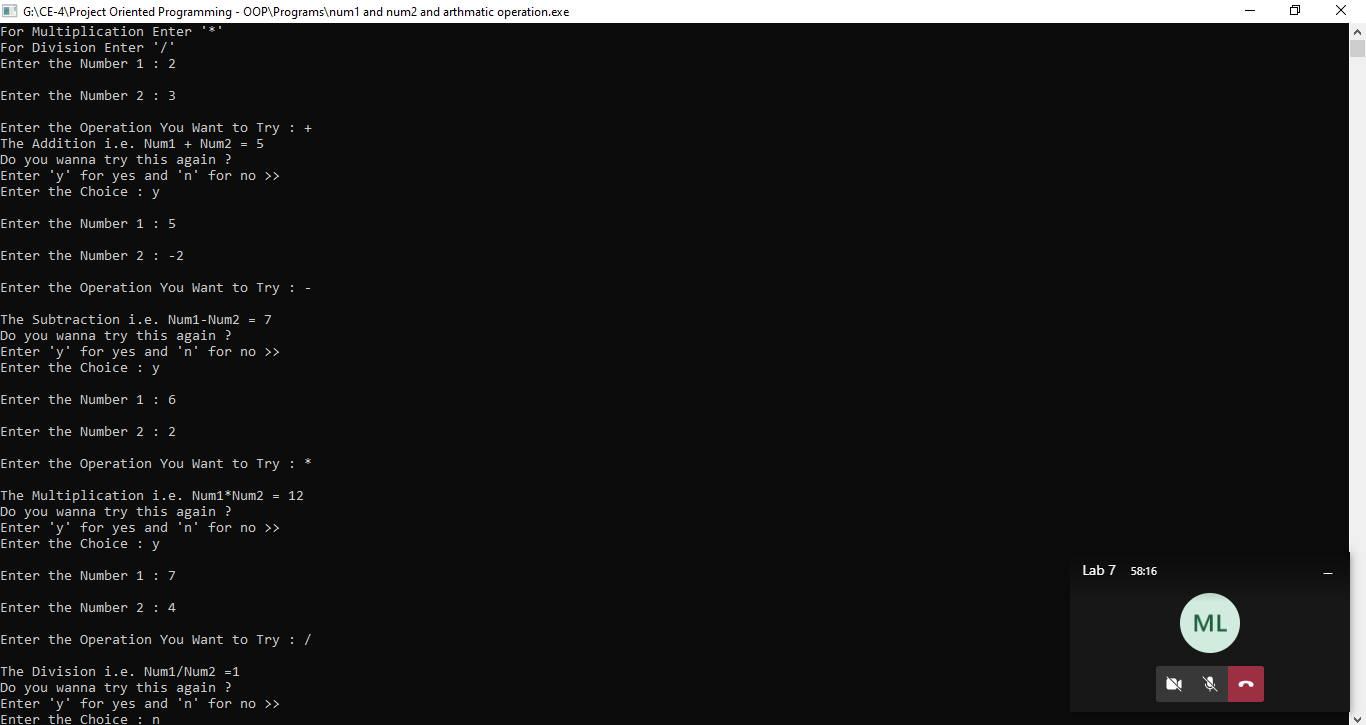
cout<<"\nEnter the Choice : ";cin>>choice;

}while(choice=='y');

return 0;

}

* **Output:**

****

* 1. Suppose we are developing a program that a car dealership can use to manage its inventory of used cars. The dealership’s inventory includes three types of automobiles: cars, pickup trucks, and sport-utility vehicles (SUVs). Regardless of the type, the dealership keeps the following data about each automobile:

• Make

• Year model

• Mileage

• Price

Each type of vehicle that is kept in inventory has these general characteristics, plus its own specialized characteristics. For cars, the dealership keeps the following additional data:

• Number of doors (2 or 4)

For pickup trucks, the dealership keeps the following additional data

• Drive type (two-wheel drive or four-wheel drive)

And, for SUVs, the dealership keeps the following additional data:

• Passenger capacity

Write a program which has a single base class to keep the record of general information regarding all automobiles further derive three classes for each type of automobile each having its specific characteristic i.e. Number of doors, Drive type and Passenger capacity.

Define appropriate accessor and mutator functions in the base class to get data from user and display the results, override these functions in respective derived classes. In main function, initialize any two of objects from user and one through argument constructor.

Your Output should look like this:

**We have the following car in inventory:**

**2007 BMW with mileage of 50000 miles having 4 doors.**

**Price: $15000.00**

**We have the following truck in inventory:**

**2006 Toyota with mileage of 40000 miles and 4WD drive type.**

**Price: $12000.00**

**We have the following SUV in inventory:**

**2005 Volvo with mileage of 30000 miles and 5 passenger capacity.**

**Price: $18000.00**

**Solution:**

* **Code:**

#include<iostream>

#include<string>

using namespace std;

class baseData

{

protected:

string companyName;

int yearModal,price,mileAge;

public:

daseData()

{

companyName=" ";

yearModal=0;

mileAge=0;

price=0;

}

void getData()

{

cin.ignore();

cout<<"\nEnter the Company Name : ";

getline(cin,companyName,'\n');

cout<<"\nEnter The Model Year : ";cin>>yearModal;

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

cout<<"\nEnter the mileAge";cin>>mileAge;

}

void putData()const

{

cout<<endl<<yearModal<<" "<<companyName<<" with mileage of "<<mileAge<<" miles ";

}

};

class car:public baseData

{

private:

int door;

public:

void getData()

{

cout<<"\nEnter the Car Data : \n";

baseData::getData();

cout<<"Number of doors (2 or 4) >> ";cin>>door;

}

void putData()

{

cout<<"\nWe have the following car in inventory:";

baseData::putData();

cout<<"having "<<door<<"doors.";

cout<<endl<<"Price: $"<<price;

}

};

class pickupTruck:public baseData

{

private:

int drive;

public:

void getData()

{

cout<<"\nEnter the Pickup Truck Data : ";

baseData::getData();

cout<<"Drive type (two-wheel drive or four-wheel drive) >>";

cin>>drive;

}

void putData()

{

cout<<"We have the following truck in inventory:";

cout<<endl;

baseData::putData();

cout<<"and "<<drive<<"WD drive type.";

cout<<endl<<"Price: $"<<price;

}

};

class SUVs:public baseData

{

private:

int pcapacity;

public:

void getData()

{

cout<<"\nEnter the sport-utility vehicles (SUVs) ";

baseData::getData();

cout<<"\nPassenger Capacity >> ";cin>>pcapacity;

}

void putData()

{

cout<<"\nWe have the following SUV in inventory:";

baseData::putData();

cout<<"and "<<pcapacity<<" passenger capacity.";

cout<<endl<<"Price: $"<<price;

}

};

int main()

{

car c;

pickupTruck pt;

SUVs s;

c.getData();

pt.getData();

s.getData();

system("CLS");

c.putData();

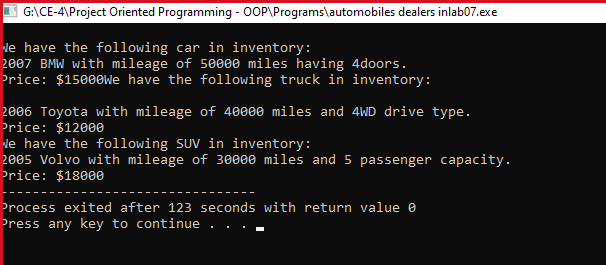
pt.putData();

s.putData();

return 0;

}

* **Output:**

****