|  |
| --- |
| Programming Lab 1-8 |
| CS304 – Object Oriented Programming |
| vu logo |
|  |
|  |

|  |
| --- |
| **Department of Computer Science, Virtual University of Pakistan** |

**Lab 01**

Consider the following scenario and identify all objects, their attributes and behaviors.

In Virtual University bookshop, the handouts of all courses are provided at reasonable price. The handouts are made in accordance with the video lectures recorded by famous professors. It is a step to facilitate students to easily digest the course contents.

**Solution:**

|  |  |
| --- | --- |
| **BookShop** | |
| **Characteristics (Attributes)** | **Behavior (Operations)** |
| Name  Address  Phone No | Set Order  Get Order  Confirm Order  Deliver Order  Update Order  Receive Payment |

|  |  |
| --- | --- |
| **Handouts** | |
| **Characteristics (Attributes)** | **Behavior (Operations)** |
| Title  Author Name  Number of Pages  Number of Chapters  Created Date  Modified Date | Download  Upload  Read  Write  Edit |

|  |  |
| --- | --- |
| **Video Lecture** | |
| **Characteristics (Attributes)** | **Behavior (Operations)** |
| Video Lecture No  Video Name  Video Duration  Video URL | Record  Upload  Download  Watch  Stop  Forward  Reverse  Increase Speed  Decrease Speed  Increase Volume  Decrease Volume  Playback |

|  |  |
| --- | --- |
| **Professor** | |
| **Characteristics (Attributes)** | **Behavior (Operations)** |
| Name  Age  Gender  Employee ID  Qualification  Specialization  Experience | Teach  Develop Assignment  Develop Quiz  Develop GDB  Prepare Exam  Take Exam  Mark Activities |

|  |  |
| --- | --- |
| **Student** | |
| **Characteristics (Attributes)** | **Behavior (Operations)** |
| Name  Age  Gender  Student ID  Study Program  Current Semester  CGPA | Study  Give Exam  Attempt Activities |

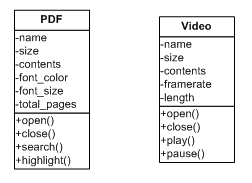
|  |  |
| --- | --- |
| **Course** | |
| **Characteristics (Attributes)** | **Behavior (Operations)** |
| Course Code  Course Title  Credit Hours  Total Marks  Course Instructor | Add Course Information  Publish Course Information  View Course Information  Edit Course Information |

|  |  |
| --- | --- |
| **Course Content** | |
| **Characteristics (Attributes)** | **Behavior (Operations)** |
| Number of Topics  Topic No  Topic Description  Composed by  Published by | Upload  View  Write  Edit |

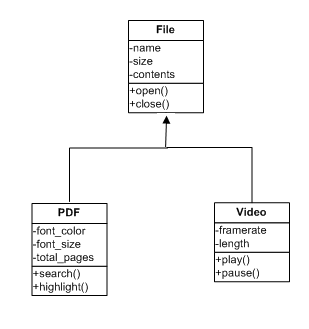
Note: Everyone visualizes real life problems in different ways, so solutions may vary from student to student. Further, Students can also add more Attributes/ Behaviors.

**Lab 02**

In the below diagram, we have two different types of file classes (PDF and Video). You are required to apply the concept of generalization on the given classes and draw class diagram showing this generalization relationship.



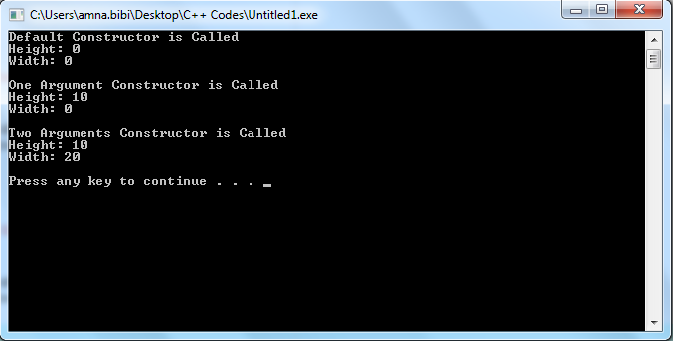
**Solution:**



**Lab 03**

Write a program which consists of a class named **Room** having two data members **Height** and **Width**, the class should also consist of three constructors i.e. Default constructor, one argument constructor and two arguments constructor.

After running your program, the following screen should display.



**Solution:**

#include<iostream>

using namespace std;

class Room {

private:

float width;

float height;

public:

Room(){

width=0.0;

height=0.0;

cout<<"Default Constructor is called"<<endl;

cout<<"Height: "<<height<<endl;

cout<<"Width: "<<width<<endl<<endl;

}

Room(float h){

height=h;

width=0.0;

cout<<"One Argument Constructor is called"<<endl;

cout<<"Height: "<<height<<endl;

cout<<"Width: "<<width<<endl<<endl;

}

Room(float h,float w){

width=w;

height=h;

cout<<"Two Argument Constructor is called"<<endl;

cout<<"Height: "<<height<<endl;

cout<<"Width: "<<width<<endl<<endl;

}

};

int main(){

Room R1, R2(10), R3(10,20);

return 0;

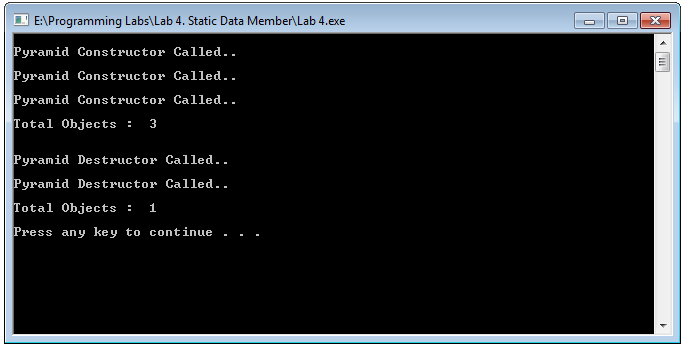
}

**Lab 04**

Write C++ coding program which creates a class Pyramid. The class should contain a static data member “Pcount” to store the total number of Pyramid’s objects.

In the main() function, dynamically create three objects of Pyramid class and print the value of “Pcount”. Afterwards, delete any two objects of Pyramid and print the value of “Pcount” again.

Following is a sample output for above scenario:



**Solution:**

#include<iostream>

using namespace std;

class Pyramid

{

private:

static int Pcount;

public:

static int get\_pcount();

Pyramid();

~Pyramid();

};

main()

{

Pyramid \*p1 = new Pyramid;

Pyramid \*p2 = new Pyramid;

Pyramid \*p3 = new Pyramid;

cout << endl << "Total Objects : " << Pyramid::get\_pcount() << endl << endl;

delete p1;

delete p2;

cout << endl << "Total Objects : " << Pyramid::get\_pcount() << endl << endl;

system("pause");

}

int Pyramid::Pcount = 0;

int ::Pyramid::get\_pcount()

{

return Pcount;

}

Pyramid::Pyramid()

{

cout << endl << "Pyramid Constructor Called.." << endl;

Pcount++;

}

Pyramid::~Pyramid()

{

cout << endl << "Pyramid Destructor Called.." << endl;

Pcount--;

}

**Lab 05**

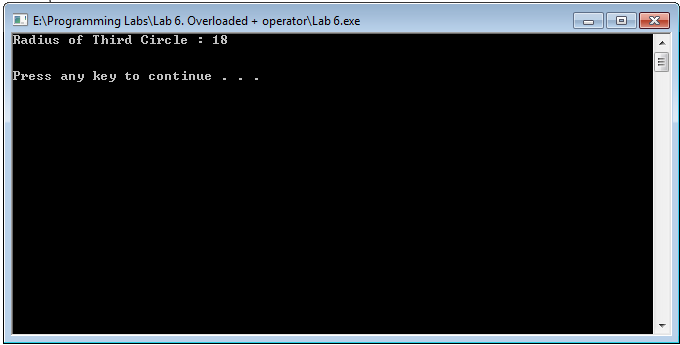
Write a C++ program which consists of a class named **Circle** which has one data member **radius** and **setter and getter functions**. Overload the **+ operator** for this class.

In main () function, create three objects c1, c2 and c3. Call setter function for objects c1 and c2. The program should be able to execute the following statement;

c3=c1+c2;

Call the getter function for c3.

After running your program, the following screen should display.



**Solution:**

#include <iostream>

using namespace std;

class Circle {

public:

void setRadius( double r ) {

radius = r;

}

double getRadius( ) {

return radius;

}

// Overload + operator to add two Box objects.

Circle operator+(const Circle& c) {

Circle cir;

cir.radius = this->radius + c.radius;

return cir;

}

private:

double radius; // radius of circle

};

// Main function for the program

int main( ) {

Circle c1,c2,c3;

double r;

c1.setRadius(6.0);

c2.setRadius(12.0);

c3 = c1 + c2;

r = c3.getRadius();

cout << "Radius of Third Circle : " << r <<endl<<endl<<endl;

system("pause");

return 0;

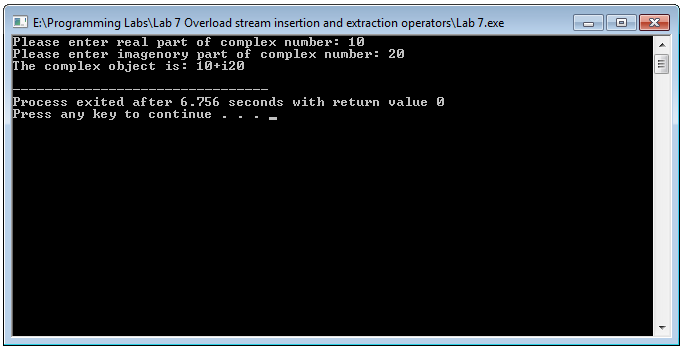
}

**Lab 06**

Write a C++ program which consists of a class named **Complex** which has data members **real** and **imaginary**. Overload the **stream insertion operator (<<) operator** and **stream extraction operator (>>)** for this class.

In main () function, create an object com1 and take input using stream extraction operator and print output using stream insertion operator

After running your program, the following screen should display.



**Solution:**

using namespace std;

#include<iostream>

#include<string.h>

#include <assert.h>

#include <iostream>

class Complex

{

private:

int real, imag;

public:

Complex(int r = 0, int i =0)

{ real = r; imag = i; }

friend ostream & operator << (ostream &outObj, const Complex &com); //stream insertion operator

friend istream & operator >> (istream &inObj, Complex &com); //stream extraction operator

};

ostream & operator << (ostream &outObj, const Complex &com)

{

outObj << com.real;

outObj << "+i" << com.imag << endl;

return outObj;

}

istream & operator >> (istream &inObj, Complex &com)

{

cout << "Please enter real part of complex number: ";

inObj >> com.real;

cout << "Please enter imagenory part of complex number: ";

inObj >> com.imag;

return inObj;

}

int main()

{

Complex com1;

cin >> com1;

cout << "The complex object is: ";

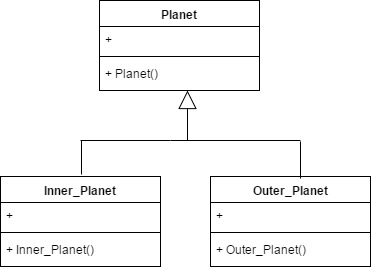
cout << com1;

return 0;

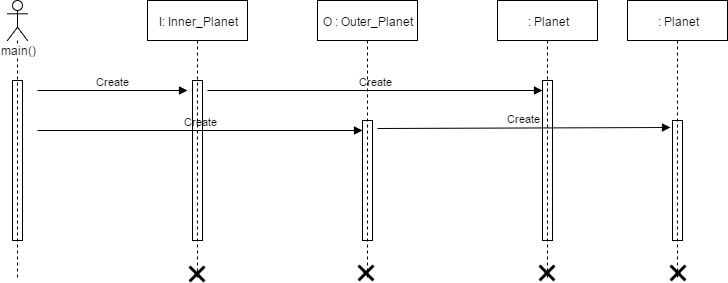
}

**Lab 07**

Write a C++ program which consists of a class named **Planet** which has no data member and only contains a default constructor. Your program should contain two more classes **Inner\_Planet** and **Outer\_Planet**. Both of these classes should also contain default constructor. Your program should implement the concept of inheritance between **“Planet class, Inner\_Planet class and Outer\_Planet class”.**

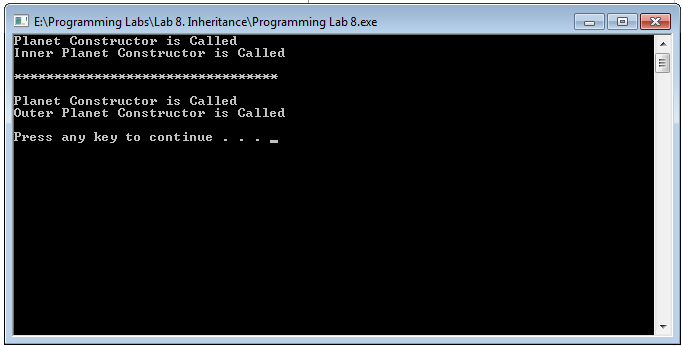


**Sequence Diagram:**



**Sample Output:**

After running your program, the following output screen should display.



**Solution:**

#include<iostream>

#include<conio.h>

using namespace std;

class Planet

{

private:

public:

Planet()

{

cout<<"Planet Constructor is Called"<<endl;

}

};

class Inner\_Planet:Planet

{

private:

float distance;

public:

Inner\_Planet()

{

cout<<"Inner Planet Constructor is Called"<<endl;

}

};

class Outer\_Planet:Planet

{

private:

float distance;

public:

Outer\_Planet()

{

cout<<"Outer Planet Constructor is Called"<<endl;

}

};

main()

{

Inner\_Planet I;

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

Outer\_Planet O;

cout<<"\n";

system("pause");

}

**Lab 08**

Write a C++ program which should consist of following four classes.

1. Base
2. Derived\_Private
3. Derived\_Protected
4. Derived\_Public

“Base” class should consist of following data members along with following type.

|  |  |
| --- | --- |
| **Data member** | **Access Specifier** |
| secret | private |
| protect | protected |
| access | public |

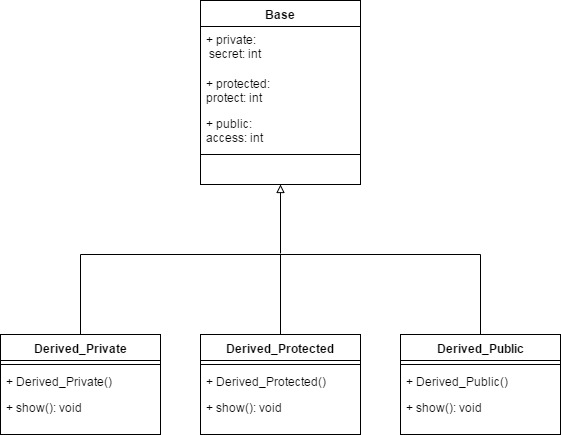
Further, “Base” class should contain default constructor to initialize its data members.

Other three classes should be inherited from “Base” class with respect to following type of inheritance.

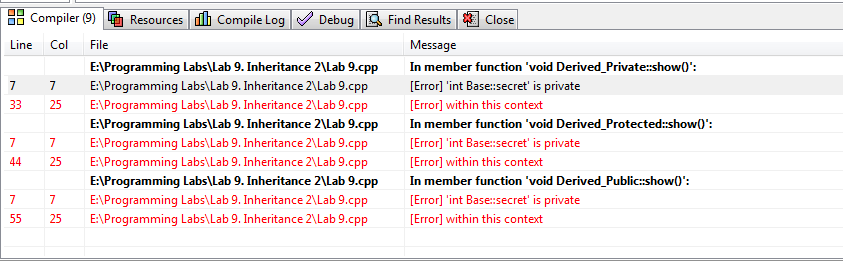
|  |  |
| --- | --- |
| **Class** | **Inheritance type** |
| Derived\_Private | Private |
| Derived\_Protected | Protected |
| Derived\_Public | Public |

All three derived classes should contain **show() function**, which should display the values of **secret, protect** and **access** data member of base class. On running this program, your compiler will generate compile time errors. Carefully observe for which type of base class data member, compiler is generating error.

**Class Diagram:**



**Compilation Results:**



**Solution:**

#include<iostream>

using namespace std;

class Base

{

private:

int secret;

protected:

int protect;

public:

int access;

public:

Base()

{

access=0;

protect=0;

secret=0;

}

};

class Derived\_Private: private Base

{

public:

void show()

{

cout<<access; // access is public, so it will be accessed from Derived\_Public class.

cout<<protect; // protect is proteced and will be accessed from Derived\_Public class.

cout<<secret; // Error. Secret can't be accessed from Derived\_Public class because it is private.

}

};

class Derived\_Protected: protected Base

{

public:

void show()

{

cout<<access; // access is public, so it will be accessed from Derived\_Public class.

cout<<protect; // protect is proteced and will be accessed from Derived\_Public class.

cout<<secret; // Error. Secret can't be accessed from Derived\_Public class because it is private.

}

};

class Derived\_Public: public Base

{

public:

void show()

{

cout<<access; // access is public, so it will be accessed from Derived\_Public class.

cout<<protect; // protect is proteced and will be accessed from Derived\_Public class.

cout<<secret; // Error. Secret can't be accessed from Derived\_Public class because it is private.

}

};

int main()

{/\*

Derived\_Public child1;

cout<<"This is accessability of base data members in derived class in case of Public Inheritance"<<endl;

child1.show();

cout<<endl<<endl;

Derived\_Protected child2;

cout<<"This is accessability of base data members in derived class in case of Protected Inheritance"<<endl;

child2.show();

cout<<endl<<endl;

Derived\_Private child3;

cout<<"This is accessability of base data members in derived class in case of Private Inheritance"<<endl;

child3.show();

cout<<endl<<endl;

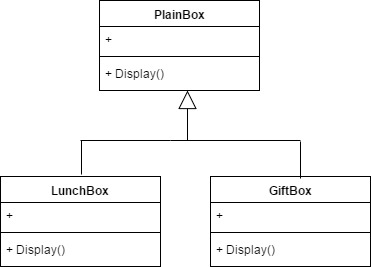
\*/

return 0;

}

**Lab 09**

**Class Diagram:**

****

**Part (1):**

Write C++ program which should contain three classes as follows:

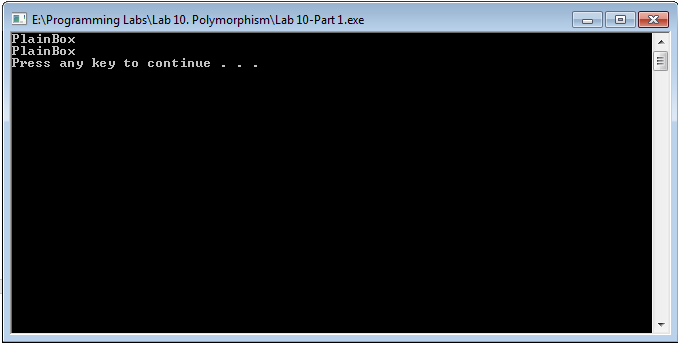
1. PlainBox
2. LunchBox
3. GiftBox

“LunchBox” and “GiftBox” classes should be publically inherited from PlainBox Class.

All three classes should contain Display() function, which should display respective class name.

In main() function, create objects of LunchBox and GiftBox classes dynamically. Static type of these objects should be “PlianBox”. Using object of PlianBox, call Display() function.

Following is a sample output for the above scenario:



**Part (2):**

Make Display() function of “PlainBox” class virtual and then run the program. Your output should be as follows:

