

**OBJECT ORIENTED PROGRAMMING FINAL EXAM**

**SUBMITTED BY:**

* **NAME: MUHAMMAD FARHAT TAYYAB**
* **ROLL NO. CSC-19F-088**
* **SECTION: CS-2B**
* **COURS: OBJECT ORIENTED PROGRAMMING**

**SUBMITTED TO:**

* **SIR ZUBAIR UDDIN SHAIKH**

**POLYMORPHISM:**

Polymorphism in C++ is a method of tackling various tasks in the identical manner. The best examples of this are function overloading and operator overloading as we an operator and it has so many tasks. Another good example below.

Basically, Polymorphism basically consisting of two words in simple poly means so many and morphism means form in simpler words we can say that a thing that plays more than one rule means multiple forms of one name.

**EXAMPLE:**

This is a very simple example to understand the Polymorphism.

#include <iostream>

using namespace std;

class Add

{

Public:

Int sum(int num 1, int num2,)

{

Return num1+num2;

}

Int sum ( int num1, int num2, int num3)

{

return num1+num2+num3+

}

};

int main()

{

Add obj;

cout<<”Output: “<<obj.sum(10, 20) <<endl;

cout <<”Output: “obj.sum(11,12,13);

return 0; }

**THE DIAMOND PROBLEM:**

The Diamond problem is basically a problem we call it diamond in C++ programming because its shape is like a diamond which is why we call it diamond. In C++ programming diamond shape malfunctioning equipment because when two classes A and B inherit from the first class means class A and the last class means class D inheritance from class B and C overridden means the objects of class A found in class B and C but class D is confused from which class he chooses the objects in which version of the method does class D inherit and in order to solve this problem.

**TEMPLATE:**

A template is a C++ tool which is simple and yet very powerful. The simple idea is to pass the data type as a parameter so we do not have to write the same code for different types of data. For example, sort) (may be required by a technology company for various data types. We can write one sort) (and pass the type of data as a parameter rather than writing and maintaining the multiple codes.

To support templates, C++ adds two new keywords: 'template' and 'typename.' You could always substitute the second keyword with the keyword 'class.'

**EXAMPLE:**

#include <iostream>

using namespace std;

template <typename Name>

Name myTem(Name x, Name y)

{

   return (x > y)? x: y;

}

int main()

{

  cout << myTem <int>(5, 8) << endl;

  cout << myTem <double>(3.0, 7.0) << endl;

  cout << myTem <char>('T', 'n') << endl;

  return 0;

}

**FRIEND CLASS:**

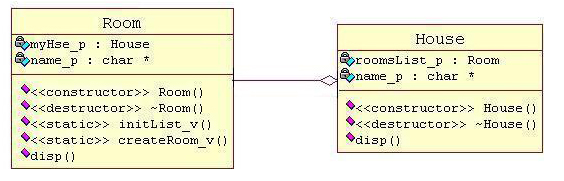
A friend class is a class that can access a class in which it is declared as friend to the private and protected members. This is required if we want to allow the private and protected members of a class to access a particular class**.**

**COMPOSITION:**

Composition is Aggregation form of specialisation again. It is a powerful form of aggregation. Here the objects of parent and child have lifetime coincidences. Child object dose does not have its own lifecycle and if the parent object is deleted then all child objects will be deleted as well.

* We normally use normal member variables
* Use the pointer values if the class of composition handles them automatically Assignment / Deallocation
* Taking responsibility for Subclass creation / destruction

**EXAMPLE:**



House may retain multiple rooms; room life is not independent and any room cannot belong to two different houses. Also, the house room will be deleted automatically if we delete it.