Object Composition in C++

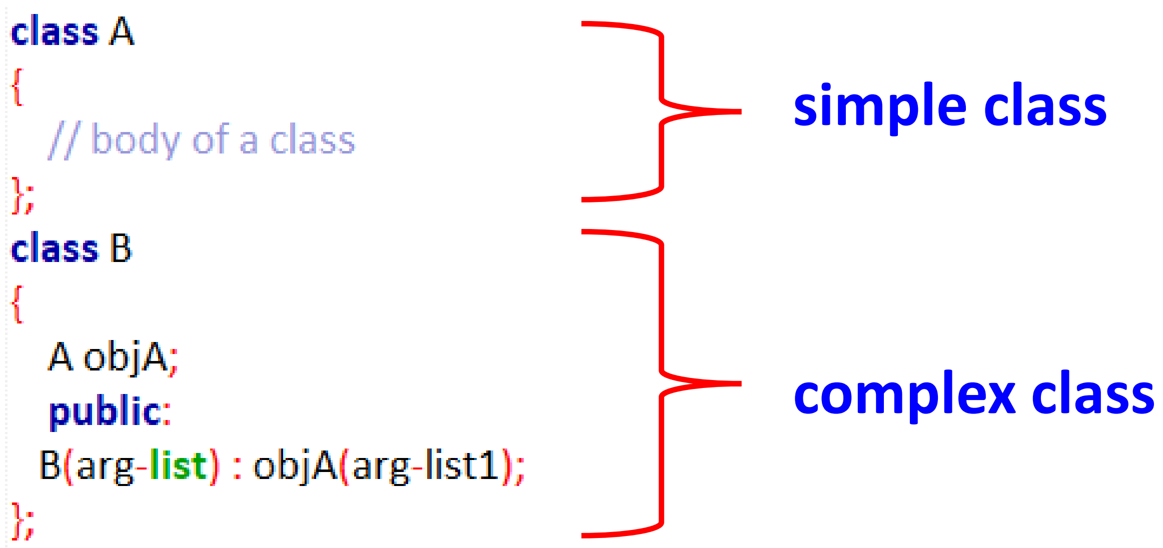
- Composition is referred to building a complex thing with the use of smaller and simple parts.
- For example,
  - A car is built using a metal frame, an engine some tires, a transmission system, a steering wheel, and large number of other parts.
  - A personal computer is built from a CPU, a motherboard, memory unit, input and output units etc.
- Composition is one of the fundamental approaches or concepts used in object-oriented programming.
- This process of building complex objects from simpler ones is called **object composition**.

Object Composition in C++ - Cont’d

- Broadly speaking, object **composition models** a **“has-a” relationship** between two objects.
- A car “has-a” tyre, computer “has-a” CPU etc.
- The complex object is sometimes called the whole, or the parent.
- The simpler object is often called the part, child, or component.
- Object Composition is useful in a C++ context because it allows us to create complex classes by combining simpler, more easily manageable parts.
- This reduces complexity, and allows us to write code faster and with less errors because we can reuse code that has already been written, tested, and verified as working.

Object Composition in C++- Cont’d

Syntax:



```
class A
{
   // body of a class
};
class B
{
   A objA;
   public:
  B(arg-list) : objA(arg-list1);
};
```

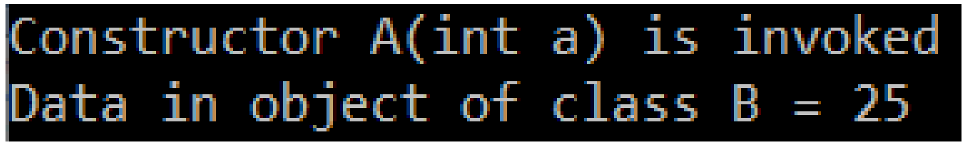
- In the classes given above, B uses objects of class A as its data members.
- Hence, B is a complex class that uses a simple class A.

Object Composition - Example

```cpp
#include <iostream>
using namespace std;
 // Simple class
class A
{
public:
   int x;
   A() { x = 0; }
   A(int a)
   {
      cout << "Constructor A(int a) is invoked\n";
      x = a;
   }
};
```

```cpp
// Complex class
class B
{
   int data;
   A objA;
 public:
   B(int a) : objA(a)
   {
      data = a;
   }
    void display()
   {
      cout << "Data in object of class B = " << data
          << endl;
      cout << "Data in member object of "
          << "class A in class B = " << objA.x;
   }
};
  int main()
  {
     B objb(25);
     objb.display();
     return 0;
  }
```

Output:



```
Constructor A(int a) is invoked
Data in object of class B = 25
```

Object Composition in C++ - Cont’d

- **Types of Object Composition in C++**
  - Object composition is basically of the following subtypes:
    - **Composition**
    - **Aggregation**
    - **Object Delegation**

Object Composition in C++- Cont’d

- **Composition:**
  - **Composition** relationship is also called a **part-whole** relationship in which the part component can only be a part of a single object simultaneously.
  - **In composition relationships, the part component will be created when the object is created, and the part will be destroyed when the object is destroyed.**
  - A person's body and heart is a good example of a part-whole relationship where if a heart is part of a person's body, then it cannot be a part of someone else's body at one time.

Object Composition in C++- Cont’d

- To qualify as a composition, the object and a part must have the following relationship-
  - The part (member) is part of the object (class).
  - The part (member) can only belong to one object (class).
  - The part (member) has its existence managed by the object (class).
  - The part (member) does not know about the existence of the object (class).
- There are some variations on the rule of creating and destroying parts:
  - A composition may avoid creating some parts until they are needed.
  - A composition may opt to use a part that has been given to it as input rather than creates the part itself.
  - A composition may delegate the destruction of its parts to some other object.

Object Composition in C++- Cont’d

- **Aggregation:**
  - The aggregation is also a part-whole relationship but here in aggregation, the parts can belong to more than one object at a time, and the whole object is not responsible for the existence of the parts.
  - To qualify as aggregation, a whole object and its part must have the following relationships:
    - The part (member) is part of the object (class).
    - The part (member) can belong to more than one object (class) at a time.
    - The part (member) does not have its existence managed by the object (class).
    - The part (member) does not know about the existence of the object (class).

Object Composition in C++- Cont'd

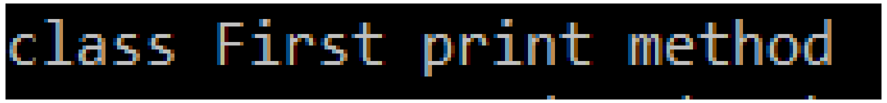
- **Object Delegation :**
    - **Object delegation** is a process in which we use the objects of a class as a member of another class.
    - Object delegation is the passing of work from one object to another.
    - It is an alternative to the process of inheritance.
    - But when the concept of inheritance is used in the program, it shows an **is-a** relationship between two different classes.
    - On the contrary, in object delegation, there is no relationship between different classes.

Object Delegation - Example

```
#include <iostream>
using namespace std;
class First
{
public:
   void print()
   {
      cout << "class First print method";
   }
};
class Second
{
   First fobj;
  public:
   void print()
    {
      fobj.print();
   }
};
```

```
int main()
{
   Second sobj;
   sobj.print();
   return 0;
}
```

Output:



```
class First print method
```