C++ Inheritance

Inheritance

In C++, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

- **derived class** (child) - the class that inherits from another class
- **base class** (parent) - the class being inherited from

To inherit from a class, use the : symbol.

In the example below, the Car class (child) inherits the attributes and methods from the Vehicle class (parent):

Example

```
// Base class
class Vehicle {
public:
string brand = "Ford";
void honk() {
cout << "Tuut, tuut! \n" ;
}
};

// Derived class
class Car: public Vehicle {
public:
string model = "Mustang";
};

int main() {
Car myCar;
myCar.honk();
cout << myCar.brand + " " + myCar.model;
return 0;
}
```

Why And When To Use "Inheritance"?

- It is useful for code reusability: reuse attributes and methods of an existing class when you create a new class.

C++ Multilevel Inheritance

Multilevel Inheritance

A class can also be derived from one class, which is already derived from another class.

In the following example, MyGrandChild is derived from class MyChild (which is derived from MyClass).

Example

```
// Base class (parent)
class MyClass {
public:
void myFunction() {
cout << "Some content in parent class." ;
}
};

// Derived class (child)
class MyChild: public MyClass {
};

// Derived class (grandchild)
class MyGrandChild: public MyChild {
};

int main() {
 MyGrandChild myObj;
 myObj.myFunction();
 return 0;
}
```

C++ Multiple Inheritance

Multiple Inheritance

A class can also be derived from more than one base class, using a **comma-separated list:**

Example

```
// Base class
class MyClass {
public:
void myFunction() {
cout << "Some content in parent class." ;
}
};

// Another base class
class MyOtherClass {
public:
void myOtherFunction() {
cout << "Some content in another class." ;
}
};

// Derived class
class MyChildClass: public MyClass, public MyOtherClass {
};

int main() {
 MyChildClass myObj;
 myObj.myFunction();
 myObj.myOtherFunction();
 return 0;
}
```

C++ Inheritance Access

Access Specifiers

You learned from the Access Specifiers chapter that there are three specifiers available in C++. Until now, we have only used public (members of a class are accessible from outside the class) and private (members can only be accessed within the class). The third specifier, protected, is similar to private, but it can also be accessed in the **inherited** class:

Example

```
// Base class
class Employee {
protected: // Protected access specifier
 int salary;
};

// Derived class
class Programmer: public Employee {
 public:
 int bonus;
 void setSalary(int s) {
 salary = s;
 }
 int getSalary() {
 return salary;
 }
};

int main() {
 Programmer myObj;
 myObj.setSalary(50000);
 myObj.bonus = 15000;
 cout << "Salary: " << myObj.getSalary() << "\n";
 cout << "Bonus: " << myObj.bonus << "\n";
 return 0;
}
```