# Tutorial 43 - Ambiguity Resolution in Inheritance in C++

### Concept Overview

Ambiguity in inheritance occurs when a derived class inherits two or more base classes that have functions with the same name. The compiler becomes confused about which function to invoke.
To resolve ambiguity, the **scope resolution operator ( `::` )** is used to explicitly specify the desired base class function.

---

### Example 1: Resolving Ambiguity in Multiple Inheritance

```
class Base1 {
public:
    void greet() {
        cout << "How are you?" << endl;
    }
};
class Base2 {
public:
    void greet() {
        cout << "Kaise ho?" << endl;
    }
};
class Derived : public Base1, public Base2 {
public:
    void greet() {
        Base2::greet(); // Explicitly call Base2's greet function
    }
};

```

**Explanation**:

1. `Base1` and `Base2` both have a `greet()` function.
2. `Derived` inherits from both `Base1` and `Base2`.
3. In `Derived`, ambiguity is resolved by specifying `Base2::greet()`.

---

### Main Function:

```
int main() {
    Base1 base1obj;
    Base2 base2obj;
    Derived d;
    base1obj.greet(); // Calls Base1's greet
    base2obj.greet(); // Calls Base2's greet
    d.greet();        // Calls Base2's greet through Derived
    return 0;
}

```

**Output**:

```
How are you?
Kaise ho?
Kaise ho?

```

---

### Example 2: Method Overriding in Single Inheritance

```
class B {
public:
    void say() {
        cout << "Hello world" << endl;
    }
};
class D : public B {
public:
    void say() {
        cout << "Hello my beautiful people" << endl;
    }
};

```

**Explanation**:

1. `B` has a function `say()`.
2. `D` inherits from `B` and overrides `say()`.
3. If `D` does not have its own `say()` function, it would call `B::say()`.

---

**Main Function:**

```
int main() {
    B b;
    D d;
    b.say(); // Calls B's say
    d.say(); // Calls D's overridden say
    return 0;
}

```

**Output**:

```
Hello world
Hello my beautiful people

```

---

### Short Notes for Notebook

1. **Ambiguity in Inheritance**:
   - Occurs when a derived class inherits multiple base classes with functions having the same name.
   - Resolved using the **scope resolution operator (`::`)**.
2. **Example 1**: Multiple Inheritance
   - `Base1` and `Base2` have `greet()` functions.
   - `Derived` resolves ambiguity by calling `Base2::greet()` explicitly.
3. **Example 2**: Method Overriding in Single Inheritance
   - A derived class overrides a base class function by redefining it.

    - If the function is not overridden, the base class function is used.
4. **Key Points**:
    - Scope resolution operator specifies which base class function to call.
    - Method overriding happens by default when a derived class has the same function as its base class.

---

### Comparison

| **Scenario** | **Resolution Method** |
|---|---|
| Multiple Inheritance | Use `BaseClassName::Function()` |
| Single Inheritance (Override) | Derived class function overrides base class automatically. |