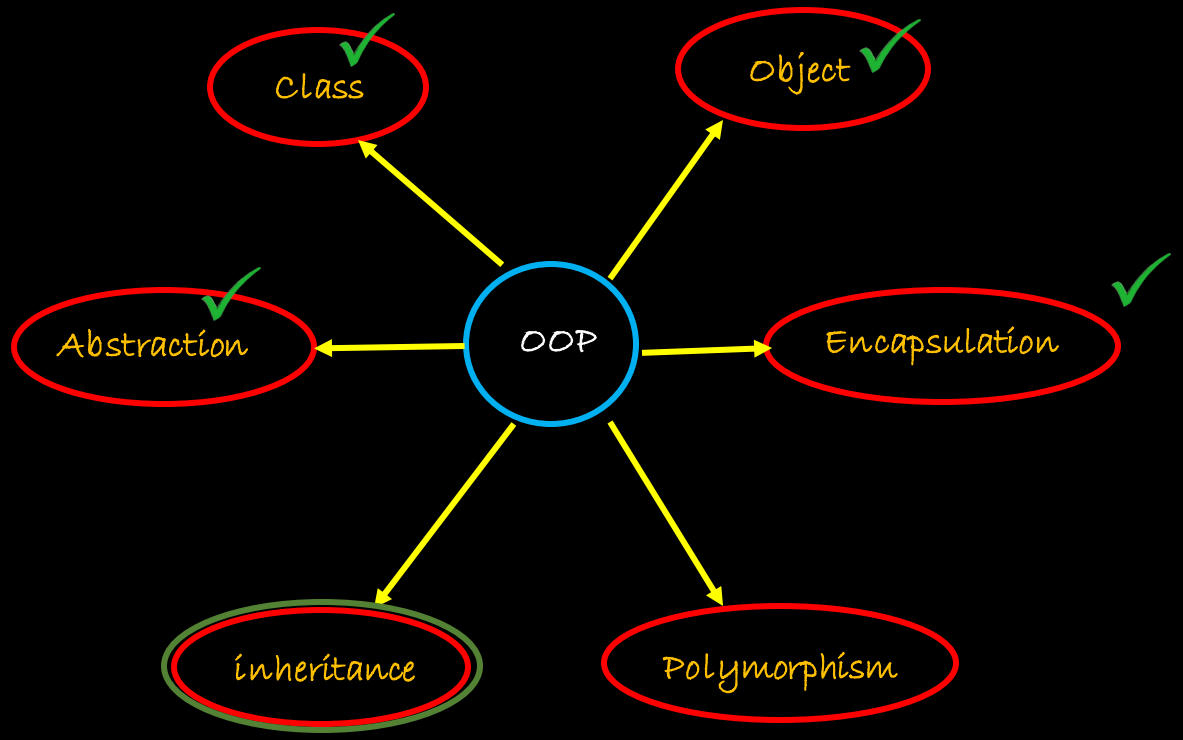
19CSE201 :Advanced Programming

Lecture 9
Inheritance in C++

By
Ritwik M
Assistant Professor(SrGr)
Dept. Of Computer Science & Engg
Amrita Vishwa Vidyapeetham - Coimbatore

A Quick Recap

- OOP Overview
- Abstraction
- Encapsulation
- Examples & Exercises



Inheritance - What is it?

- Definition(s)
  - It is a process in which one object acquires all the properties and behaviors of its parent object automatically.
  - The capability of a class to derive properties and characteristics from another class is called Inheritance.
  - The technique of deriving a new class from an old one is called inheritance.
- Old Class is called : Super class or Base class
  - It is the class that gets inherited
  - Consider it as a parent class
- New class is called: Derived class or sub class
  - It is the class that does the inheriting
  - Consider it as a child class

Why Inheritance?

- Code Reusability
  - Write common properties in base class and extend it to subclasses
- Method Overriding
- Virtual Keyword

Discussed later

Syntax & Rules

- Declaration
  - `class subClassName : accessMode superClassName`
  - Example
  - `class derivedClass : public : baseClass`

- While defining a subclass like this, the super class must be already defined or at least declared before the subclass declaration.

- Access Mode is used to specify, the mode in which the properties of superclass will be inherited into subclass, public, private or protected

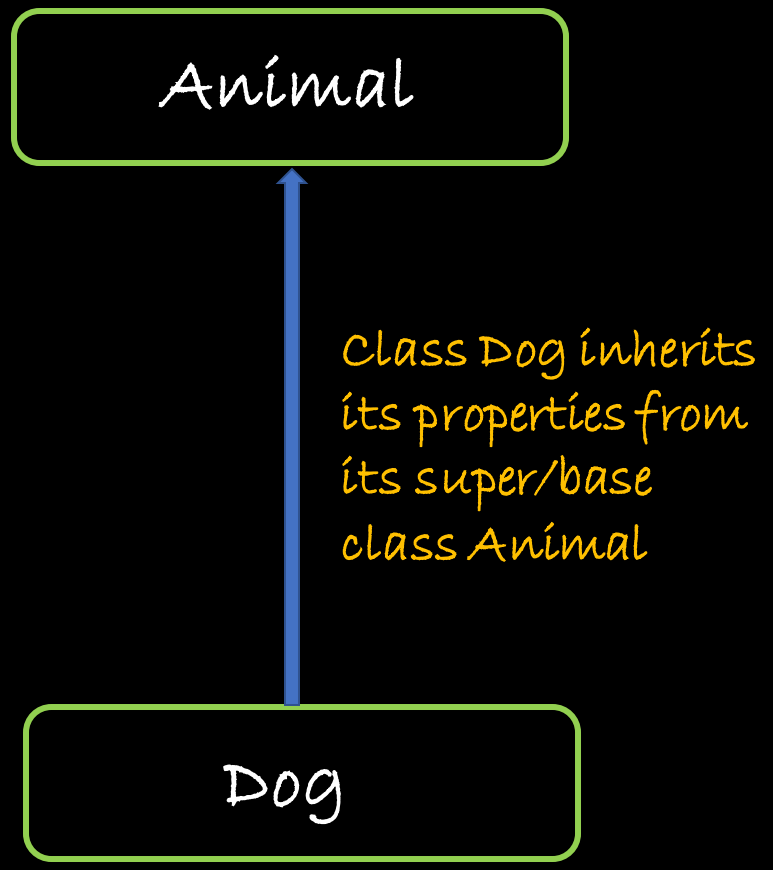
- <u>NOTE:</u> All members of a class except Private, can be inherited

Example

```
class Animal
{
    public:
    int legs = 4;
};

// Dog class inheriting Animal class
class Dog : public Animal
{
    public:
    int tail = 1;
};

int main()
```



Types of Inheritance

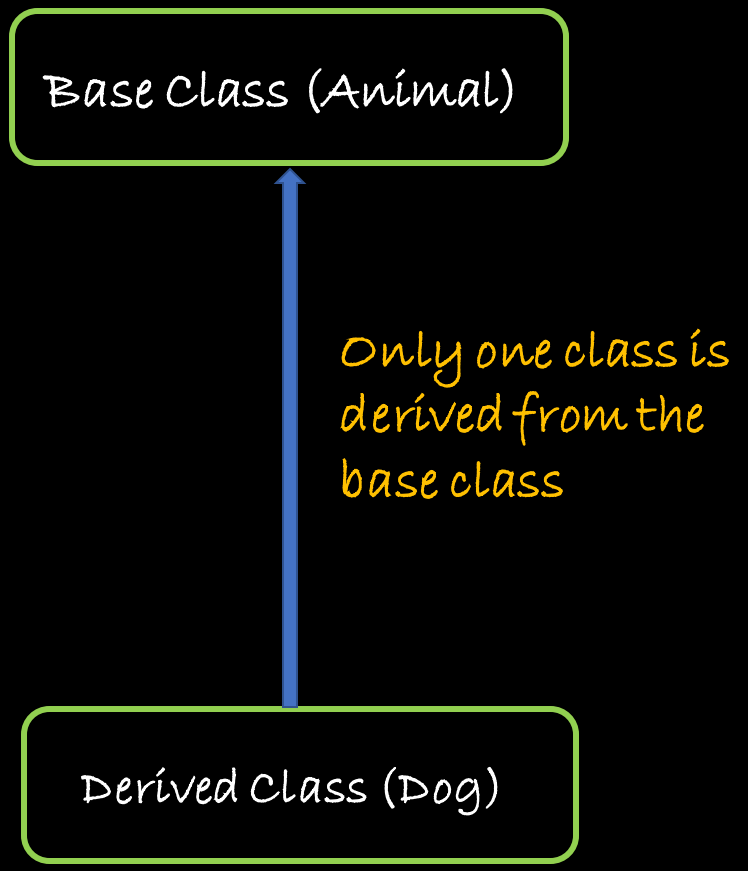
- Single Inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Multilevel Inheritance
- Hybrid Inheritance (also known as Virtual Inheritance)

Single Inheritance - Example

```
class Animal
{
    public:
    int legs = 4;
};

// Dog class inheriting Animal class
class Dog : public Animal
{
    public:
    int tail = 1;
};

int main()
```



Importance of Access Specifiers

- Public
  - Syntax:
    - `class Subclass : public Superclass`
  - This is the most used inheritance mode.
  - In this the protected member of super class becomes protected members of sub class and public becomes public.
- Private
  - Syntax:
    - `class Subclass : Superclass  // No need for specifier - default is private`
  - Here the protected and public members of super class become private members of derived class.
- Protected
  - Syntax:
    - `class subclass : protected Superclass`
  - In protected mode, the public and protected members of Super class becomes protected members of Sub class.

Example – Access specifiers in Inheritance

```
//PUBLIC
class A {

public:
int x;

protected:
int y;

private:
int z;
};

class B : public A {
 // x stays public
 // y stays protected
// z is not accessible from
B
};
```

```
//PRIVATE
class A {

public:
int x;

protected:
int y;

private:
int z;
};

// 'private' is default for
classes
class D : private A {
 // x becomes private
 // y becomes private
 // z is not accessible from D
}
```

```
//PROTECTED
class A {

public:
int x;

protected:
int y;

private:
int z;
};

class C : protected A {
 // x becomes protected
 // y stays protected
 // z is not accessible from C
};
```

Exercise – 1: Predict the output

```
class A {

public:
int x;

protected:
int y;

private:
int z;
};

class B : public A {
 // x stays public
 // y stays protected
 // z is not accessible from B
};
```

```
int main()
{
    B b;
    b.x=5;
    b.y=10;
    b.z=15
}
```

Exercise – 2: Predict the output

```
class A {

public:
int x;

protected:
int y;

private:
int z;
};

class B : public A {
 // x stays public
 // y stays protected
 // z is not accessible from B
};
```

```
int main()
{
    B b;
    b.x=5;
}
```

Exercise – 3: Predict the output

```
class A {

public:
int x;

protected:
int y;

private:
int z;
};
```

```
class B : public A
{
    public:
    void display()
    {
      y=20;
      z=25;
      cout<<"x:"<<x<<"y:"<<y<<endl;
      cout<<"z: " <<z<<endl;
    }
};
```

```
int main()
{
    B b;
    b.x=5;
    b.display();
}
```

Exercise – 4: Predict the output

```
class A {

public:
int x;

protected:
int y;

private:
int z;
};
```

```
class C : protected A {
 // x becomes protected
 // y stays protected
 // z is not accessible from C
};

int main()
{
    C c;
    c.x=5;
    c.y=10;
    c.z=15;
}
```

Exercise – 5: Predict the output

```
class A {

public:
int x;

protected:
int y;

private:
int z;
};
```

```
class C : protected A
{
    public:
    void display()
    {
     y=20;
     x=10;
     cout<<"x:"<<x<<"y:"<<y<<endl;
    }
};
```

```
int main()
{
    C c;
    c.display();
}
```

Exercise – 6: Predict the output

```
class A {

public:
int x;

protected:
int y;

private:
int z;
};
```

```
class C : protected A
{
    public:
    void display()
    {
     y=20;
     x=10;
     z=25;
     cout<<"x:"<<x<<"y:"<<y<<endl;
     cout<<"z: " <<z<<endl;
    }
};
```

```
int main()
{
    C c;
    c.display();
}
```

Exercise – 7: Predict the output

```
class A {

public:
int x;

protected:
int y;

private:
int z;
};
```

```
class D : private A {
 // x becomes private
 // y becomes private
 // z is not accessible from D
};

int main()
{
    D d;
    d.x=5;
    d.y=10;
    d.z=15;
}
```

Exercise – 8: Predict the output

```
class A {

public:
int x;

protected:
int y;

private:
int z;
};
```

```
class D : private A
{
    public:
    void display()
    {
     y=20;
     x=10;
     cout<<"x:"<<x<<"y:"<<y<<endl;
     }
};
```

```
int main()
{
    D d;
    d.display();
}
```

Exercise – 9: Predict the output

```
class A {

public:
int x;

protected:
int y;

private:
int z;
};
```

```
class D : private A
{
    public:
    void display()
    {
     y=20;
     x=10;
     z=25
     cout<<"x:"<<x<<"y:"<<y<<endl;
     cout<<"z: " <<z<<endl;
     }
};
```

```
int main()
{
    D d;
    d.display();
}
```

Exercise – 10: Predict the output

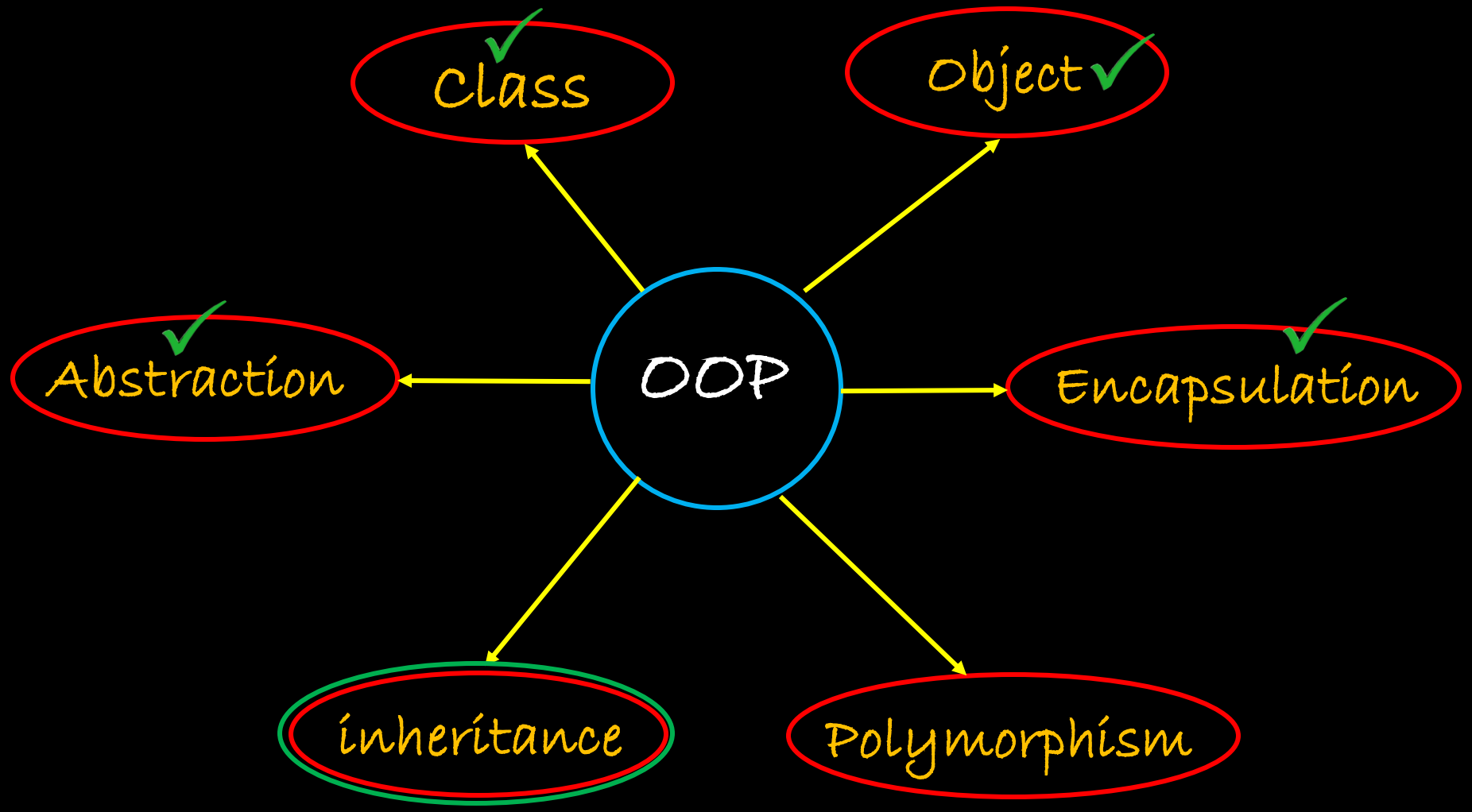
```
class base {
    int arr[10];
};
class b1: public base { };
class b2: public base { };
class derived: public b1, public b2 {};
```

```
int main()
{
  cout<<sizeof(derived);
  return 0;
}
```

Assume that an integer takes 4 bytes and write down the answer in your notebook

What if you are running this code on an x64 computer? Does your answer change?

OOP overview



Quick Summary

- Inheritance
- Why Inheritance
- Single inheritance
- Access specifiers and inheritance
- Examples and Exercises

Up Next

More on Inheritance in C++