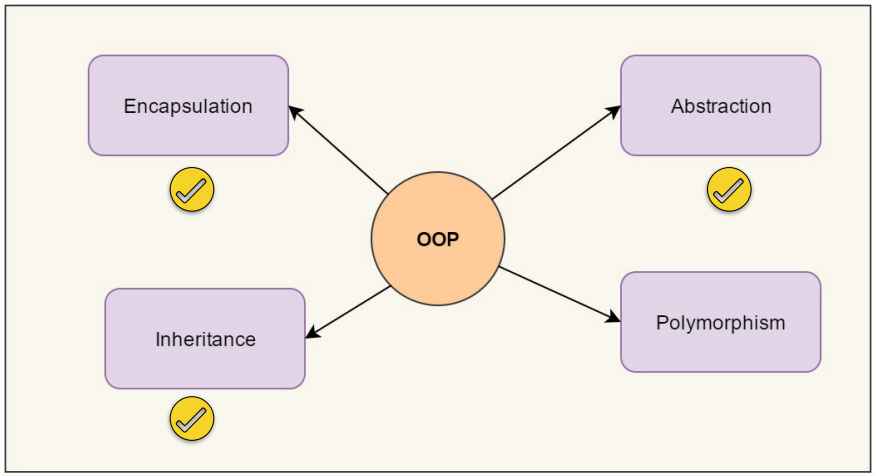
CSC 211: Object Oriented Programming
Class Inheritance

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Spring 2020

THINK BIG WE DO℠



Four Pillars of Object Oriented Programming

Inheritance in C++

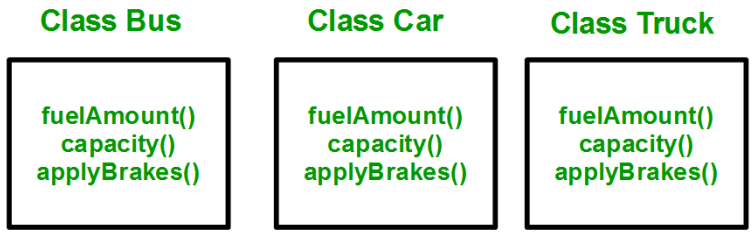
- The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important feature of Object Oriented Programming.
- **Derived Class:** The class that inherits properties from another class is called Sub class or Derived Class.
- **Base Class:**The class whose properties are inherited by sub class is called Base Class or Super class.
- Derived class is a *superset* of the base class.

https://www.geeksforgeeks.org/inheritance-in-c/

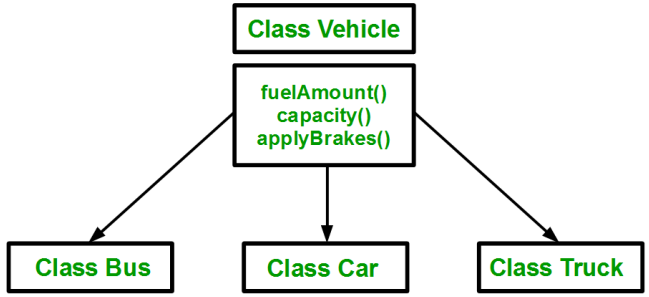
Inheritance in C++

- What if we create a stand alone function that accepts an object from the base class as an argument. Could we pass in a derived class object instead?
- Yes! The derived class object has everything a base class object would have (and maybe more)!

Why and When to use Inheritance?



Note: duplication of same code 3 times



Note: by using inheritance, we can avoid the duplication of data and increase re-usability of code

Note: by using inheritance, we can avoid the duplication of data and increase re-usability of code

Implementing Inheritance

Syntax

```
class subclass_name : access_mode base_class_name
{
  //body of subclass
};
```

https://www.geeksforgeeks.org/inheritance-in-c/

Modes of inheritance

Modes of inheritance

- **Public mode**: If we derive a sub class from a public base class then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.
- **Protected mode**: If we derive a sub class from a Protected base class then both public member and protected members of the base class will become protected in derived class.
- **Private mode**: If we derive a sub class from a Private base class then both public member and protected members of the base class will become Private in derived class.

https://www.geeksforgeeks.org/inheritance-in-c/

Example

```
// C++ Implementation to show that a derived class
// doesn't inherit access to private data members.
// However, it does inherit a full parent object
class A {

public:
    int x;

protected:
    int y;

private:
    int z;

};

class B : public A {

    // x is public
    // y is protected
    // z is not accessible from B
};

class C : protected A {

    // x is protected
    // y is protected
    // z is not accessible from C
};

class D : private A {   // 'private' is default for classes

    // x is private
    // y is private
    // z is not accessible from D
};
```

https://www.geeksforgeeks.org/inheritance-in-c/

Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

https://www.geeksforgeeks.org/inheritance-in-c/

Order of Constructor Call with Inheritance

Order of Constructor Call

- Base class constructors are **always called** in the derived class constructors.
- Whenever you create a derived class object, **first** the base class default constructor is executed and **then** the derived class's constructor finishes execution.

Order of Constructor Call

```
class Base
{
    int x;
    public:
    // default constructor
    Base()
    {
        std::cout << "Base default constructor\n";
    }
};


class Derived : public Base
{
    int y;
    public:
    // default constructor
    Derived()
    {
        std::cout << "Derived default constructor\n";
    }
    // parameterized constructor
    Derived(int i)
    {
        std::cout << "Derived parameterized constructor\n";
    }
};
```


Order of Constructor Call

```
int main()
{
    Base b;
    Derived d1;
    Derived d2(10);
}
```

OUTPUT

Base default constructor
Base default constructor
Derived default constructor
Base default constructor
Derived parameterized constructor

Inheritance Example

Example

```
class Entity{
    public:
        float x, y;

        void move(float xa, float ya){
            x += xa;
            y += ya;
        }

        void printLoc(){
            std::cout << "x = " << x << std::endl;
            std::cout << "y = " << y << std::endl;
        }
};
```

Example

```
class Player{

    public:
       const char* name;
       float x, y;

    void move(float xa, float ya){
           x += xa;
           y += ya;
       }

    void printLoc(){
           std::cout << "x = " << x << std::endl;
           std::cout << "y = " << y << std::endl;
       }

    void printName(){

        std::cout << name << std::endl;
    }

};
```


Example

```
class Player : public Entity{

    public:
       const char* name;

    void printName(){

        std::cout << name << std::endl;
    }
};
```


Practice

- Write a base Person class with following properties and methods
- Person:
  Member Variables: name, age, favorite color, birthday
  Methods: ‘Getters’ and ‘Setters’ for all member Variables
- Derive a Student and Employee class that extends the Person class with the additional attributes
- Student: GPA, Major, Year, StudentID
- Employee: Job Title, Salary, Years Employed

https://www.geeksforgeeks.org/inheritance-in-c/