Base Class and Derived Class

In this lesson, we'll be learning about how a base class attributes are available to the derived classes and how to define base and a derived class.

WE'LL COVER THE FOLLOWING

- Vehicle as a Base Class
- Derived Classes
- Modes of Inheritance
- Public Inheritance
  - Explanation

In the last lesson, we have seen that `Vehicle` class attributes are shared by the other two classes(`Cars` and `Ships`).

Vehicle as a Base Class

We can consider the `Vehicle` class as a `base` class as it has common attributes.

Derived Classes

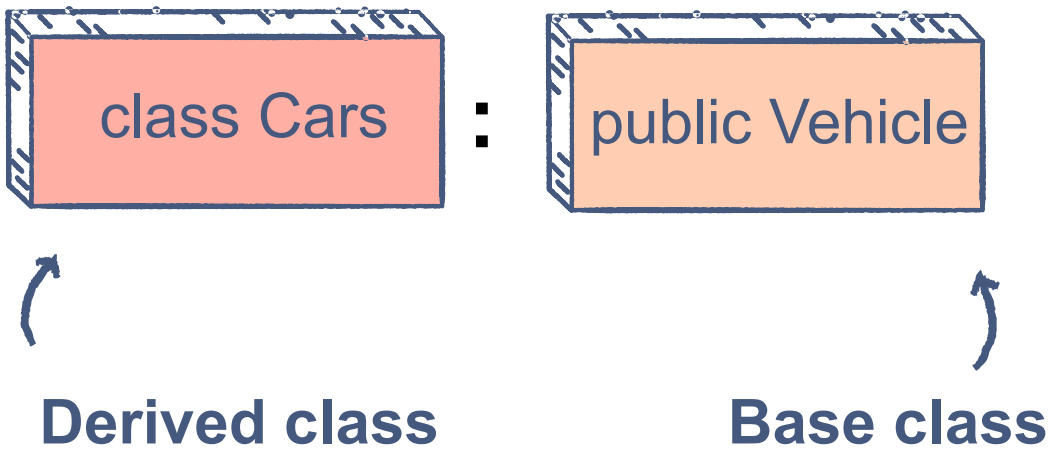
Cars and *Ships* are considered as `derived` classes as they're inheriting properties from `vehicle` class.

Modes of Inheritance

There are three modes of class inheritance: `public`, `private` and `protected`. The basic syntax for inheritance is given below:

```
class derivedClassName : modeOfInheritance baseClassName
```

We use the keyword **public** to implement **public** inheritance.



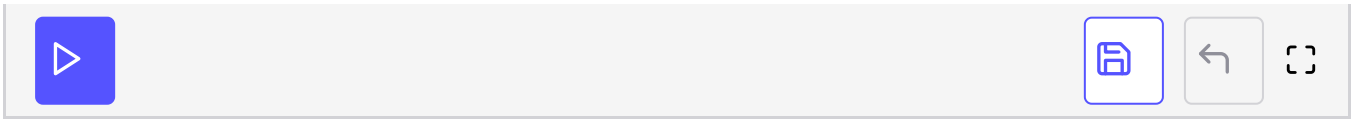
Now, the class `Cars` have access to the public members of a base class `vehicle` and the protected data is inherited as protected data, and the private data is not inherited, but it can be accessed directly by the public member functions of the class.

Public Inheritance

We are updating our `Cars` and `Ships` class so that *Make*, *Color*, *Year*, *Model* and the function `void print_details()` can be inherited from base class `Vehicle`. So, we have removed these variables and function from the derived classes. The following example shows the classes `Cars` and `Ships` that inherits **publicly** from the base class `Vehicle`.

```
class Vehicle{
  protected:
  string Make;
  string Color;
  int Year;
  string Model;

  public:
  Vehicle(){
    Make = "";
    Color = "";
    Year = 0;
    Model = "";
  }

  Vehicle(string mk, string col, int yr, string mdl){
    Make = mk;
    Color = col;
    Year = yr;
    Model = mdl;
  }

  void print_details(){
```

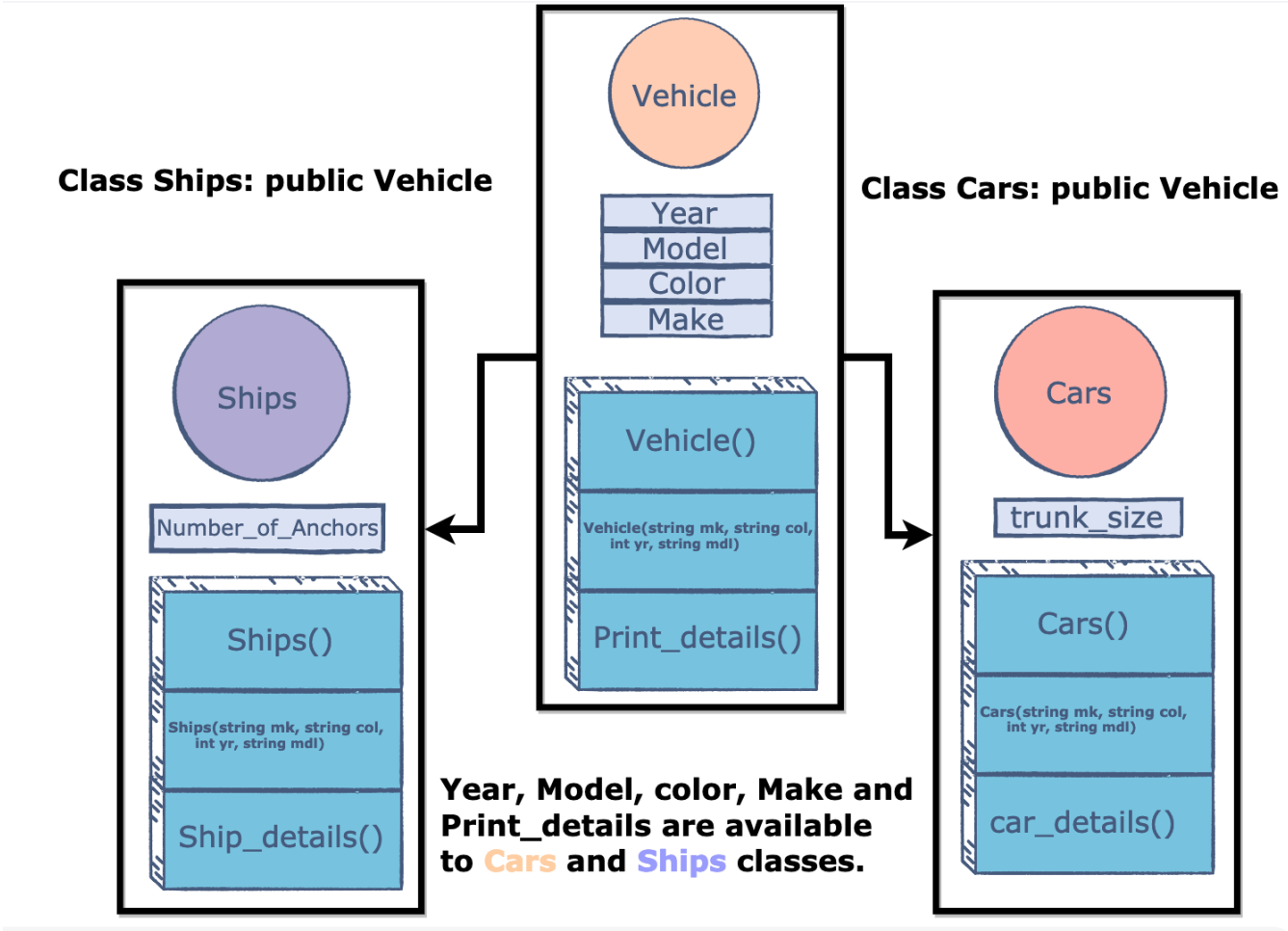
```
    cout << "Manufacturer: " << Make << endl;
    cout << "Color: " << Color << endl;
    cout << "Year: " << Year << endl;
    cout << "Model: " << Model << endl;
  }
};

class Cars: public Vehicle{
  string trunk_size;

  public:
  Cars(){
    trunk_size = "";
  }

  Cars(string mk, string col, int yr, string mdl, string ts)
    :Vehicle(mk, col, yr, mdl){
    trunk_size = ts;
  }

  void car_details(){
    print_details();
    cout << "Trunk size: " << trunk_size << endl;
  }
};

class Ships: public Vehicle{
  int Number_of_Anchors;

  public:
  Ships(){
    Number_of_Anchors = 0;
  }

  Ships(string mk, string col, int yr, string mdl, int na)
  :Vehicle(mk, col, yr, mdl){
    Number_of_Anchors = na;
  }

  void Ship_details(){
    print_details();
    cout << "Number of Anchors: " << Number_of_Anchors << endl;
  }
};

int main(){
  Cars car("Chevrolet", "Black", 2010, "Camaro", "9.1 cubic feet");
  car.car_details();

  cout << endl;

  Ships ship("Harland and Wolff, Belfast", "Black and whilte",
             1912, "RMS Titanic", 3);
  ship.Ship_details();
}
```



The highlighted lines in the above code indicate how to achieve inheritance in

C++ by using `:` and mentioning the mode of inheritance.

The following illustration explains the concept of inheritance in the above code:



Explanation

Now the `Ships` and `Cars` classes have access to public member functions of the base class `Vehicle` as shown in the above illustration. `Protected` and `public` data members are accessible to derived classes.

Now that we have learned about the base and derived classes. So, let's move to the next lesson in which we'll learn about the base class constructors and destructors.