Modes of Inheritance

In this lesson, we'll learn about how Public, Private and Protected inheritance is done in C++.

You are already familiar with Access Modifiers from the Classes chapter. By using these specifiers, we limit the access of the data members and member functions to the other *classes* and *main*.

`private` Mode of Inheritance

By using `private` inheritance, the *private* data members and member functions of the base class are inaccessible in the derived class and in `main`. *Protected* and *Public* members of the base class are accessible to the derived class but not in `main`.

Let's look at the implementation using `private` inheritance:

```
class Vehicle{

  string Make;
  string Color;
  int Year;

  protected:
  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;
  }
};

class Cars: private 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;
    cout << "Model: " << Model << endl;  // Model is protected and
    // is accessible in derived class
  }
};

int main(){
  Cars car("Chevrolet", "Black", 2010, "Camaro", "9.1 cubic feet");
  // car.Year = 2000;     // this will give error as Year is private
  // car.Model = "Accord";   // this will give error as Model is protected

  car.car_details();
  //car.print_details();   // public functions of base class are inaccessible in main
}
```

`protected` Mode of Inheritance

By using `protected` inheritance, the *private* members of the base class are inaccessible in the derived class and in `main`. *Protected* and *Public* members of the base class are accessible to the derived class but not in `main`.

Let’s take an example of `protected` inheritance:

```cpp
class Vehicle{

  string Make;
  string Color;
  int Year;

  protected:
  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;
  }
};

class Cars: protected 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;
    cout << "Model: " << Model << endl;  // Model is protected and
    // is accessible in derived class
  }
};

int main(){
  Cars car("Chevrolet", "Black", 2010, "Camaro", "9.1 cubic feet");
  // car.Year = 2000;     // this will give error as Year is private
  // car.Model = "Accord";   // this will give error as Model is protected

  car.car_details();
  //car.print_details();   // public functions of base class are inaccessible in main
}
```

`public` Mode of Inheritance

By using `public` inheritance, the *private* members of the base class are inaccessible in the derived class and in `main`. *Protected* members of the base class are accessible to the derived class but not in `main`. *Public* members of the base class are accessible to the derived class and in `main`.

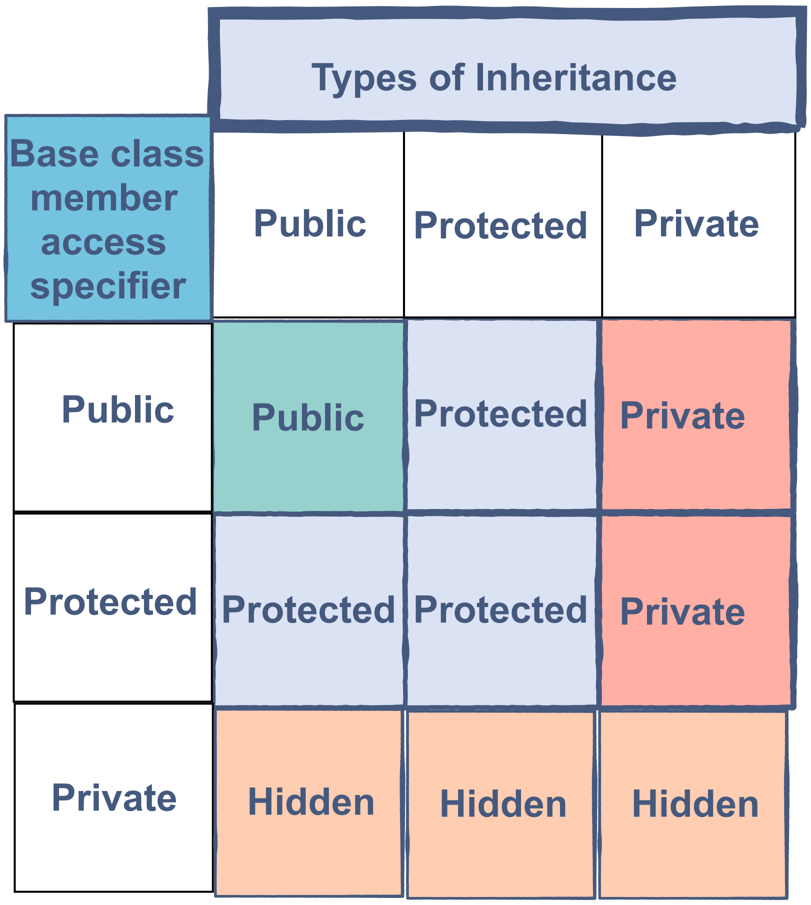
Let's look at the implementation using `public` inheritance:

```
class Vehicle{

  string Make;
  string Color;
  int Year;

  protected:
  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;
  }
};

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(){
    cout << "Trunk size: " << trunk_size << endl;
    cout << "Model: " << Model << endl;  // Model is protected and
    // is accessible in derived class
  }
};

int main(){
  Cars car("Chevrolet", "Black", 2010, "Camaro", "9.1 cubic feet");
  // car.Year = 2000;     // this will give error as Year is private
   //car.Model = "Accord";   // this will give error as Model is protected

  car.car_details();
  car.print_details();   // public functions of base class are accessible in main
}
```

Modes of Inheritance in Base Class

The given table depicts the access of members of our base class when we use specific modifiers and its behavior.



	Types of Inheritance		
Base class member access specifier	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Hidden	Hidden	Hidden

In the next lesson, we'll be learning about multiple inheritance which is a core concept of inheritance.