Object Oriented Programming with C++

11. Inheritance

By: Prof. Pandav Patel

Second Semester, 2020-21 Computer Engineering Department Dharmsinh Desai University

Inheritance: Introduction

- Vehicle has wheels (diff num), has maximum speed, manufacturer etc...
 - Automobile has engine, has gears etc...

Can we resuse code instead?

- PassangeVehicle passanger capacity etc...
- GoodCarrier max load in weight, max load in volume etc...
- Cart Some animal to pull the cart, number of animals required etc...

```
class Automobile {
class Vehicle {
                                                                                   class Cart {
                                           int wheels_count;
         int wheels_count;
                                                                                            int wheels_count;
         float max_speed;
                                          float max_speed;
                                                                                            float max_speed;
         string manufacturer;
                                           string manufacturer;
                                                                                            string manufacturer;
                                                                                            string animal_type;
                                          float engine_cc;
public:
         Vehicle(int, float,
                                                                                            int animal count;
                                           int gear_count;
string);
                                 public:
                                                                                   public:
         void print();
                                          Automobile(int, float, string,
                                                                                            Cart(int, float, string,
};
                                 float, int);
                                                                                   string, int);
                                           void print();
                                                                                            void print();
                                                                                   };

    Is there duplication of code?
```

Inheritance: Introduction

- Inheritance (IS-A relationship)
 - Mechanism of deriving new class from existing class is called Inheritance (or derivation)
 - Existing class used in inheritance is called base class / parent class / super class
 - Newly created class is called derived class / child class / sub class
 - Syntax of inheritance is as follows:

```
class derived-class: visibility-mode base-class

{

// Members of derived class

derived-class
```

- derived-class IS-A base-class.
 - visibility-mode could be private / protected / public
 - Its optional, by default its private
 - Though visibility mode is often public, will look into it first

Inheritance: Introduction

Data Members

Derived class inherits all data members of Base class

Derived class may add data members of its own

Member Functions

Derived class inherits all member functions of Base class

Derived class may override a member function of Base class by redefining it with the same signature

Derived class may overload a member function of Base class by redefining it with the same name; but different signature

Access Specification

Derived class cannot access private members of Base class

Derived class can access protected members of Base class

Construction-Destruction

A constructor of the Derived class must first call a constructor of the Base class to construct the Base class instance of the Derived class

The destructor of the Derived class must call the destructor of the Base class to destruct the Base class instance of the Derived class

```
class Derived:public Base{
#include<iostream>
using std::cout;
using std::endl;
                                                  Other o;
class Base{
                                                  Derived()
Base()
                                                     cout<<"Constructor: Derived"<<endl;</pre>
     cout<<"Constructor: Base"<<endl;</pre>
                                                  ~Derived()
  ~Base()
                                                     cout<<"Destructor: Derived"<<endl;</pre>
     cout<<"Destructor: Base"<<endl;</pre>
                                                int main()
class Other
                                                  Derived derived;
                                                                                            Output ??
                                                  return 0;
Other()
     cout<<"Constructor: Other"<<endl;</pre>
  ~Other()
     cout<<"Destructor: Other"<<endl;</pre>
```

```
#include<iostream>
using std::cout;
using std::endl;
class Base{
Base()
     cout<<"Constructor: Base"<<endl;</pre>
  ~Base()
     cout<<"Destructor: Base"<<endl;</pre>
class Other
Other()
     cout<<"Constructor: Other"<<endl;</pre>
  ~Other()
     cout<<"Destructor: Other"<<endl;</pre>
```

```
class Derived:public Base{
  Other o;
  Derived()
     cout<<"Constructor: Derived"<<endl;</pre>
  ~Derived()
     cout<<"Destructor: Derived"<<endl;</pre>
int main()
  Derived derived;
  return 0;
```

Output

Error: Base::Base() is private

Error Other::Other() is private

```
#include<iostream>
using std::cout;
using std::endl;
class Base{
public:
  Base()
     cout<<"Constructor: Base"<<endl;</pre>
  ~Base()
     cout<<"Destructor: Base"<<endl;</pre>
class Other
public:
  Other()
     cout<<"Constructor: Other"<<endl;
  ~Other()
     cout<<"Destructor: Other"<<endl;</pre>
```

```
class Derived:public Base{
public:
  Other o;
  Derived()
     cout<<"Constructor: Derived"<<endl;</pre>
  ~Derived()
     cout<<"Destructor: Derived"<<endl;</pre>
int main()
  Derived derived;
  return 0;
```

Output ??

```
#include<iostream>
using std::cout;
using std::endl;
class Base{
public:
  Base()
     cout<<"Constructor: Base"<<endl;
  ~Base()
     cout<<"Destructor: Base"<<endl;</pre>
class Other
public:
  Other()
     cout<<"Constructor: Other"<<endl;
  ~Other()
     cout<<"Destructor: Other"<<endl;</pre>
```

```
class Derived:public Base{
public:
  Other o:
  Derived()
     cout<<"Constructor: Derived"<<endl;
  ~Derived()
     cout<<"Destructor: Derived"<<endl;</pre>
int main()
  Derived derived;
  return 0;
```

Output:

Constructor: Base
Constructor: Other
Constructor: Derived
Destructor: Derived
Destructor: Other
Destructor: Base

```
class B {
public:
int data_;
B(int d = 0) : data_(d) 
  cout << "B::B(int): " << data_ << endl; }
~B() {
  cout << "B::~B(): " << data_ << endl; }
class D: public B {
int info_;
public:
D(int d, int i): B(d), info_(i) // ctor-1: Explicit construction of Base
  cout << "D::D(int, int): " << data_ << ", " << info_ << endl; }
D(int i): info_(i) // ctor-2: Default construction of Base
  cout << "D::D(int): " << data_ << ", " << info_ << endl; }
~D() { cout << "D::~D(): " << data_ << ", " << info_ << endl; }
```

```
Object b Object d1 Object d2

5
2
3
```

```
Int main()
{
    B b(5);

// ctor-1: Explicit construction of Base D d1(1, 2);

// ctor-2: Default construction of Base //If base class constructor do not have default value → Error D d2(3);
return 0;
}
```

```
class B {
public:
int data_;
B(int d = 0) : data_(d) 
  cout << "B::B(int): " << data_ << endl; }
~B() {
  cout << "B::~B(): " << data_ << endl; }
class D: public B {
int info_;
public:
D(int d, int i)
B(d);
Info_= i ;
cout << "D::D(int, int): " << data_ << ", " << info_ << endl;
D(int i): info_(i) // ctor-2: Default construction of Base
{ cout << "D::D(int): " << data_ << ", " << info_ << endl; }
~D() { cout << "D::~D(): " << data_ << ", " << info_ << endl; }
```

```
Int main()
 B b(5);
D d1(1, 2);
Error: declaration of 'B d'
shadows a parameter
as we must have base object prior
to call to constructor of D. We
haven't supplied it so compiler call
default constructor of B and set
data_ =0 (default).
Now, code B(d) inside constructor
try to create new object of B but as
we already have an object of B
created by compiler using default
constructor of base class
return 0;
```

```
class B {
public:
int data_;
B(int d = 0) : data_(d) 
   cout << "B::B(int): " << data_ << endl; }
~B() {
   cout << "B::~B(): " << data_ << endl; }
class D: public B {
int info_;
public:
D(int d, int i)
//B(d);
Info_= i ;
cout << "D::D(int, int): " << data_ << ", " << info_ << endl;
D(int i): info_(i) // ctor-2: Default construction of Base
{ cout << "D::D(int): " << data_ << ", " << info_ << endl; }
~D() { cout << "D::~D(): " << data_ << ", " << info_ << endl; }
```

```
Int main()
 B b(5);
D d1(1, 2);
Now the code will work fine, the
only issue is compiler will set data_
variable to 0 (default) rather setting
it to 1 (user's input)
return 0;
```

```
#include<iostream>
                                                                    int main() {
#include<string>
                                                                      Vehicle vehicle(2, 7.1, "Ambica carts");
using std::cout;
                                                                      vehicle.vehicle_print();
using std::endl;
using std::string;
                                                                      return 0;
class Vehicle {
  int wheels_count;
  float max_speed;
  string manufacturer;
public:
  Vehicle(int, float, string);
  void vehicle_print();
Vehicle::Vehicle(int wheels_count, float max_speed, string manufacturer) {
  this->wheels_count = wheels_count;
  this->max_speed = max_speed;
  this->manufacturer = manufacturer;
                                                                      Wheel Count: 2
void Vehicle::vehicle_print() {
                                                                      Max Speed: 7.1
  cout << "Wheel Count: " << wheels count << endl;
                                                                      Manufacturer: Ambica carts
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl;
```

```
class Vehicle {
                                                           Automobile(int wheels_count, float max_speed,
                                                                             string manufacturer,
  int wheels_count;
                                                                             float engine_cc, int gear_count):
  float max_speed;
                                                            engine_cc(engine_cc),
  string manufacturer;
                                                            gear_count(gear_count),
public:
                                                            // error: no matching function for call to 'Vehicle::Vehicle()'
  Vehicle(int, float, string);
  void vehicle_print();
                                                            void print() {
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                              cout << "Wheel Count: " << wheels_count << endl;</pre>
                                     string manufacturer):
                                                              cout << "Max Speed: " << max_speed << endl;</pre>
wheels_count(wheels_count),
                                                              cout << "Manufacturer: " << manufacturer << endl;</pre>
max_speed(max_speed),
                                                              cout << "Engine Capacity: " << engine_cc << endl;</pre>
manufacturer(manufacturer) {
                                                              cout << "Gear Count: " << gear_count << endl;</pre>
  cout << "Vehicle constructor called\n";</pre>
                                                         };
void Vehicle::vehicle_print() {
  cout << "Wheel Count: " << wheels_count << endl;
                                                         int main() {
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                           Automobile automobile(2, 100, "Honda", 125, 4);
  cout << "Manufacturer: " << manufacturer << endl;
                                                            return 0:
class Automobile : public Vehicle{
  float engine_cc;
  int gear_count;
public:
```

```
class Vehicle {
                                                          Automobile(int wheels_count, float max_speed,
                                                                           string manufacturer,
  int wheels_count;
                                                                  float engine_cc, int gear_count):
  float max_speed;
                                                          engine_cc(engine_cc),
  string manufacturer;
                                                          gear_count(gear_count),
public:
                                                          Vehicle(wheels_count, max_speed, manufacturer) {
  Vehicle(int, float, string);
                                                             cout << "Automobile constructor called\n";
  void vehicle_print();
                                                          void print() {
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                             // error: 'int Vehicle::wheels_count' is private
                                    string manufacturer):
                                                             cout << "Wheel Count: " << wheels_count << endl;
wheels_count(wheels_count),
                                                             // similar error
max_speed(max_speed),
                                                             cout << "Max Speed: " << max_speed << endl;</pre>
manufacturer(manufacturer) {
                                                             // similar error
  cout << "Vehicle constructor called\n";</pre>
                                                             cout << "Manufacturer: " << manufacturer << endl:
                                                             cout << "Engine Capacity: " << engine_cc << endl;
void Vehicle::vehicle_print() {
                                                             cout << "Gear Count: " << gear_count << endl;</pre>
  cout << "Wheel Count: " << wheels_count << endl;</pre>
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                        };
  cout << "Manufacturer: " << manufacturer << endl;
                                                        int main() {
                                                          Automobile automobile(2, 100, "Honda", 125, 4);
                                                          automobile.print();
class Automobile : public Vehicle{
                                                          return 0;
  float engine_cc;
  int gear_count;
public:
```

```
Vehicle constructor called
class Vehicle {
                          Automobile constructor called
                                                           Automobile(int wheels_count, float max_speed,
public:
                          Wheel Count: 2
                                                                             string manufacturer,
                          Max Speed: 100
  int wheels_count;
                                                                   float engine_cc, int gear_count):
                          Manufacturer: Honda
  float max_speed;
                                                           engine_cc(engine_cc),
                          Engine Capacity: 125
  string manufacturer;
                                                           gear_count(gear_count),
                          Gear Count: 4
                                                            Vehicle(wheels_count, max_speed, manufacturer) {
  Vehicle(int, float, string);
                                                              cout << "Automobile constructor called\n";</pre>
  void vehicle_print();
                                                           void print() {
Vehicle::Vehicle(int wheels_count, float max_speed,
                                     string manufacturer):
                                                              cout << "Wheel Count: " << wheels_count << endl;</pre>
wheels_count(wheels_count),
max_speed(max_speed),
                                                              cout << "Max Speed: " << max_speed << endl;</pre>
manufacturer(manufacturer) {
  cout << "Vehicle constructor called\n";</pre>
                                                              cout << "Manufacturer: " << manufacturer << endl;
                                                              cout << "Engine Capacity: " << engine_cc << endl;</pre>
void Vehicle::vehicle_print() {
                                                              cout << "Gear Count: " << gear_count << endl;</pre>
  cout << "Wheel Count: " << wheels_count << endl;</pre>
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl;
                                                         int main() {
                                                           Automobile automobile(2, 100, "Honda", 125, 4);
                                                           automobile.print();
class Automobile : public Vehicle{
                                                           return 0;
  float engine_cc;
  int gear_count;
public:
```

```
Vehicle constructor called
class Vehicle {
                          Automobile constructor called
                          Wheel Count: 2
                          Max Speed: 100
  int wheels_count;
                          Manufacturer: Honda
  float max_speed;
                           Engine Capacity: 125
  string manufacturer;
                          Gear Count: 4
public:
  Vehicle(int, float, string);
  void vehicle_print();
Vehicle::Vehicle(int wheels_count, float max_speed,
                                     string manufacturer):
wheels_count(wheels_count),
max_speed(max_speed),
manufacturer(manufacturer) {
  cout << "Vehicle constructor called\n";</pre>
void Vehicle::vehicle_print() {
  cout << "Wheel Count: " << wheels_count << endl;</pre>
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl;
class Automobile : public Vehicle{
  float engine_cc;
  int gear_count;
public:
```

```
Automobile(int wheels_count, float max_speed,
                   string manufacturer,
          float engine_cc, int gear_count):
  engine_cc(engine_cc),
  gear_count(gear_count),
  Vehicle(wheels_count, max_speed, manufacturer) {
     cout << "Automobile constructor called\n";
  void print() {
         vehicle_print();
     cout << "Engine Capacity: " << engine_cc << endl;</pre>
     cout << "Gear Count: " << gear_count << endl;</pre>
int main() {
  Automobile automobile(2, 100, "Honda", 125, 4);
  automobile.print();
  return 0;
```

```
Vehicle constructor called
class Vehicle {
                          Automobile constructor called
protected:
                          Wheel Count: 2
                          Max Speed: 100
  int wheels_count;
                          Manufacturer: Honda
  float max_speed;
                          Engine Capacity: 125
  string manufacturer;
                          Gear Count: 4
  Vehicle(int, float, string);
  void vehicle_print();
Vehicle::Vehicle(int wheels_count, float max_speed,
                                     string manufacturer):
wheels_count(wheels_count),
max_speed(max_speed),
manufacturer(manufacturer) {
  cout << "Vehicle constructor called\n";</pre>
void Vehicle::vehicle_print() {
  cout << "Wheel Count: " << wheels_count << endl;</pre>
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl;
class Automobile : public Vehicle{
  float engine_cc;
  int gear_count;
public:
```

```
Automobile(int wheels_count, float max_speed,
                   string manufacturer,
          float engine_cc, int gear_count):
  engine_cc(engine_cc),
  gear_count(gear_count),
  Vehicle(wheels_count, max_speed, manufacturer) {
     cout << "Automobile constructor called\n";
  void print() {
     cout << "Wheel Count: " << wheels_count << endl;
     cout << "Max Speed: " << max_speed << endl;</pre>
     cout << "Manufacturer: " << manufacturer << endl;</pre>
     cout << "Engine Capacity: " << engine_cc << endl;</pre>
     cout << "Gear Count: " << gear_count << endl;</pre>
int main() {
  Automobile automobile(2, 100, "Honda", 125, 4);
  automobile.print();
  return 0:
```

```
class Vehicle {
                                                           Automobile(int wheels_count, float max_speed,
protected:
                                                                            string manufacturer,
  int wheels_count;
                                                                  float engine_cc, int gear_count):
  float max_speed;
                                                           engine_cc(engine_cc),
  string manufacturer;
                                                           gear_count(gear_count),
                                                           Vehicle(wheels_count, max_speed, manufacturer) {
  Vehicle(int, float, string);
                                                             cout << "Automobile constructor called\n";
  void vehicle_print();
                                                           void print() {
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                             cout << "Wheel Count: " << wheels_count << endl;
                                     string manufacturer):
                                                             cout << "Max Speed: " << max_speed << endl;</pre>
wheels_count(wheels_count),
                                                             cout << "Manufacturer: " << manufacturer << endl;</pre>
max_speed(max_speed),
                                                             cout << "Engine Capacity: " << engine_cc << endl;</pre>
manufacturer(manufacturer) {
                                                             cout << "Gear Count: " << gear_count << endl;</pre>
  cout << "Vehicle constructor called\n";</pre>
void Vehicle::vehicle_print() {
                                                        int main() {
  cout << "Wheel Count: " << wheels_count << endl;</pre>
                                                           Automobile automobile(2, 100, "Honda", 125, 4);
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                           // error: 'void Vehicle::vehicle_print()' is protected
  cout << "Manufacturer: " << manufacturer << endl;
                                                           automobile.vehicle_print();
                                                           return 0;
class Automobile : public Vehicle{
  float engine_cc;
  int gear_count;
public:
```

Inheritance: access specifiers

Access Specifier	Access from methods of class itself	Access from methods of derived class	Acess from methods of other classes and independent functions
public	Allowed	Allowed	Allowed
protected	Allowed	Allowed	Not Allowed
private	Allowed	Not Allowed	Not Allowed

```
class Vehicle {
                                                                 Automobile(int wheels_count, float max_speed,
  int wheels count;
                                                                                    string manufacturer,
protected:
                                                                        float engine_cc, int gear_count):
  float max_speed;
                                                                 engine_cc(engine_cc),
public:
                                                                 gear_count(gear_count),
  string manufacturer;
                                                                 Vehicle(wheels_count, max_speed, manufacturer) {
  Vehicle(int, float, string);
                                                                   cout << "Automobile constructor called\n";
  void vehicle_print();
                                                                 void print() {
Vehicle::Vehicle(int wheels count, float max speed
                                                                   // error: 'int Vehicle::wheels_count' is private
                                             string manufacturer):
                                                                   cout << "Wheel Count: " << wheels count << endl:
wheels count(wheels count),
                                                                   cout << "Max Speed: " << max_speed << endl;</pre>
max_speed(max_speed),
                                                                   cout << "Manufacturer: " << manufacturer << endl:
manufacturer(manufacturer){
                                                                   cout << "Engine Capacity: " << engine_cc << endl;</pre>
  cout << "Vehicle constructor called\n";
                                                                   cout << "Gear Count: " << gear_count << endl;</pre>
void Vehicle::vehicle print() {
  cout << "Wheel Count: " << wheels_count << endl;
                                                              int main() {
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                                 Automobile automobile(2, 100, "Honda", 125, 4);
  cout << "Manufacturer: " << manufacturer << endl:
                                                                 // error: 'int Vehicle::wheels_count' is private
                                                                 cout << "Wheel Count: " << automobile.wheels count << endl;
                                                                 // error: 'float Vehicle::max_speed' is protected
class Automobile : public Vehicle{
                                                                 cout << "Max Speed: " << automobile.max_speed << endl;</pre>
  float engine_cc;
                                                                 cout << "Manufacturer: " << automobile.manufacturer << endl:
  int gear count;
                                                                 automobile.print();
public:
                                                                 return 0;
```

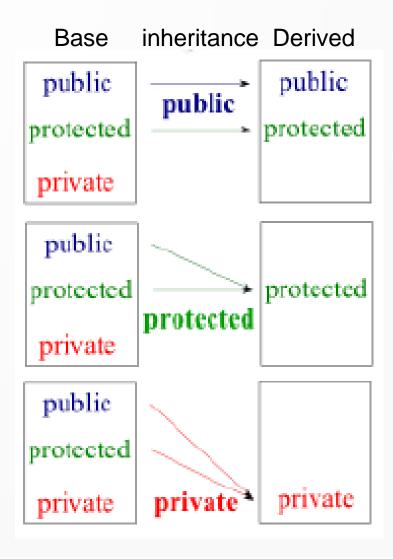
```
Vehicle constructor called
class Vehicle {
                            Automobile constructor called
  int wheels count;
                            Manufacturer: Honda //(from main)
protected:
                            Max Speed: 100
  float max_speed;
                            Manufacturer: Honda
public:
                            Engine Capacity: 125
  string manufacturer;
                            Gear Count: 4
  Vehicle(int, float, string);
  void vehicle_print();
Vehicle::Vehicle(int wheels_count, float max_speed
                                            string manufacturer):
wheels count(wheels count),
max_speed(max_speed),
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";
void Vehicle::vehicle print() {
  cout << "Wheel Count: " << wheels_count << endl;
                                                             int main() {
  cout << "Max Speed: " << max speed << endl;
  cout << "Manufacturer: " << manufacturer << endl:
class Automobile : public Vehicle{
  float engine_cc;
  int gear count;
public:
                                                                return 0;
```

```
Automobile(int wheels_count, float max_speed,
                   string manufacturer,
       float engine_cc, int gear_count):
engine_cc(engine_cc),
gear_count(gear_count),
Vehicle(wheels_count, max_speed, manufacturer) {
  cout << "Automobile constructor called\n";
void print() {
  // error: 'int Vehicle::wheels_count' is private
  // cout << "Wheel Count: " << wheels_count << endl;
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl:
  cout << "Engine Capacity: " << engine_cc << endl;</pre>
  cout << "Gear Count: " << gear_count << endl;</pre>
Automobile automobile(2, 100, "Honda", 125, 4);
// error: 'int Vehicle::wheels count' is private
// cout << "Wheel Count: " << automobile.wheels count << endl;
// error: 'float Vehicle::max_speed' is protected
// cout << "Max Speed: " << automobile.max_speed << endl;
cout << "Manufacturer: " << automobile.manufacturer << endl:
automobile.print();
```

Inheritance: visibility mode

class derived-class : visibility-mode base-class

Visibility mode	Base class	Derived class
public	public	public
	protected	protected
	private	No access
protected	public	protected
	protected	protected
	private	No access
private	public	private
	protected	private
	private	No access



```
Vehicle constructor called
class Vehicle {
                            Automobile constructor called
  int wheels count;
                            Max Speed: 100
protected:
                            Manufacturer: Honda
  float max_speed;
                            Engine Capacity: 125
public:
                            Gear Count: 4
  string manufacturer;
  Vehicle(int, float, string);
  void vehicle_print();
Vehicle::Vehicle(int wheels_count, float max_speed
                                            string manufacturer):
wheels_count(wheels_count),
max_speed(max_speed),
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";
void Vehicle::vehicle print() {
  cout << "Wheel Count: " << wheels_count << endl;
  cout << "Max Speed: " << max speed << endl;
  cout << "Manufacturer: " << manufacturer << endl:
class Automobile : protected Vehicle{
  float engine_cc;
  int gear count;
public:
```

```
Automobile(int wheels_count, float max_speed,
                     string manufacturer,
          float engine_cc, int gear_count):
  engine_cc(engine_cc),
  gear_count(gear_count),
  Vehicle(wheels_count, max_speed, manufacturer) {
     cout << "Automobile constructor called\n";
  void print() {
    // error: 'int Vehicle::wheels count' is private
    // cout << "Wheel Count: " << wheels count << endl:
     cout << "Max Speed: " << max_speed << endl;</pre>
     cout << "Manufacturer: " << manufacturer << endl:
     cout << "Engine Capacity: " << engine_cc << endl;</pre>
     cout << "Gear Count: " << gear_count << endl;</pre>
int main() {
  Automobile automobile(2, 100, "Honda", 125, 4);
  // error: 'int Vehicle::wheels_count' is private
  // cout << "Wheel Count: " << automobile.wheels count << endl;
  // error: 'float Vehicle::max speed' is protected
  // cout << "Max Speed: " << automobile.max_speed << endl;
  // error: string Vehicle::manufacturer is inaccessible
  // cout << "Manufacturer: " << automobile.manufacturer << endl;
  automobile.print();
  return 0;
                    Protected Understanding:
```

For Derive class it is like public

For Other classes and functions it is like private

```
Vehicle constructor called
class Vehicle {
                           Automobile constructor called
  int wheels count;
                           Max Speed: 100
protected:
                           Manufacturer: Honda
  float max_speed;
                           Engine Capacity: 125
public:
                           Gear Count: 4
  string manufacturer;
  Vehicle(int, float, string),
  void vehicle_print();
Vehicle::Vehicle(int wheels_count, float max_speed
                                            string manufacturer):
wheels count(wheels count),
max_speed(max_speed),
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";
void Vehicle::vehicle print() {
  cout << "Wheel Count: " << wheels_count << endl;
  cout << "Max Speed: " << max speed << endl;
  cout << "Manufacturer: " << manufacturer << endl:
class Automobile : private Vehicle{
  float engine_cc;
  int gear count;
public:
```

```
Automobile(int wheels_count, float max_speed,
                      string manufacturer,
          float engine_cc, int gear_count):
  engine_cc(engine_cc),
  gear_count(gear_count),
  Vehicle(wheels_count, max_speed, manufacturer) {
     cout << "Automobile constructor called\n";</pre>
  void print() {
    // error: 'int Vehicle::wheels count' is private
    // cout << "Wheel Count: " << wheels count << endl:
     cout << "Max Speed: " << max_speed << endl;</pre>
     cout << "Manufacturer: " << manufacturer << endl:
     cout << "Engine Capacity: " << engine_cc << endl;</pre>
     cout << "Gear Count: " << gear_count << endl;</pre>
int main() {
  Automobile automobile(2, 100, "Honda", 125, 4);
  // error: 'int Vehicle::wheels_count' is private
  // cout << "Wheel Count: " << automobile.wheels count << endl;
  // error: 'float Vehicle::max speed' is protected
  // cout << "Max Speed: " << automobile.max_speed << endl;
  // error: string Vehicle::manufacturer is inaccessible
  // cout << "Manufacturer: " << automobile.manufacturer << endl;
  automobile.print();
  return 0;
```

```
class Vehicle {
                                                        class Automobile: public Vehicle{
  int wheels_count;
                                                          float engine_cc;
protected:
                                                        protected:
  float max speed;
                                                          int gear count;
public:
                                                        public:
  string manufacturer;
                                                          Automobile(int wheels_count, float max_speed,
  Vehicle(int, float, string);
                                                            string manufacturer, float engine_cc,
  void vehicle_print();
                                                            int gear count):
                                                          engine cc(engine cc),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                          gear_count(gear_count),
  string manufacturer):
                                                          Vehicle(wheels_count, max_speed, manufacturer) {
wheels count(wheels count),
                                                            cout << "Automobile constructor called\n";
max_speed(max_speed),
manufacturer(manufacturer){
                                                          void automobile print() {
  cout << "Vehicle constructor called\n";
                                                            // error: 'int Vehicle::wheels count' is private
                                                            // cout << "Wheel Count: " << wheels_count << endl;
void Vehicle::vehicle_print() {
                                                            cout << "Max Speed: " << max speed << endl;
  cout << "Wheel Count: " << wheels count << endl;
                                                             cout << "Manufacturer: " << manufacturer << endl;</pre>
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                            cout << "Engine Capacity: " << engine_cc << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl:</pre>
                                                            cout << "Gear Count: " << gear count << endl:
 Vehicle constructor called
 Automobile constructor called
 PassangerVehicle constructor called
 Manufacturer: Honda
                         Vehicle
                                          Automobile
                                                                 PassangerVehicle
                                                                                        pv (object in main)
 Max Speed: 100
 Manufacturer: Honda
                         wheels count
                                          wheels count
                                                                 wheels count
                                                                                        wheels count
 Gear Count: 4
                         max_speed
                                          max_speed
                                                                 max_speed
                                                                                        max_speed
 Max Passan: 60
                         manufacturer
                                          manufacturer
                                                                 manufacturer
                                                                                        manufacturer
                         vehicle_print
                                          vehicle_print
                                                                 vehicle_print
                                                                                        vehicle print
                                          engine cc
                                                                 engine cc
                                                                                        engine cc
```

gear count

print

automobile_print

max_passangers

gear count

print

automobile_print

max_passangers

gear count

automobile_print

Vehicle

Automobile

PassangerVehicle

```
class PassangerVehicle: public Automobile {
  int max_passangers;
public:
  PassangerVehicle(int wheels count, float max speed,
     string manufacturer, float engine_cc,
     int gear_count, int max_passangers):
  max_passangers(max_passangers),
  Automobile(wheels_count, max_speed, manufacturer,
     engine cc, gear count) {
     cout << "PassangerVehicle constructor called\n";</pre>
  void print() {
     // error: 'int Vehicle::wheels count' is private
     // cout << "Wheel Count: " << wheels count << endl;
     cout << "Max Speed: " << max_speed << endl;
     cout << "Manufacturer: " << manufacturer << endl:
     // error: 'float Automobile::engine_cc' is private
     // cout << "Eng Cap: " << engine_cc << endl;
     cout << "Gear Count: " << gear count << endl;
     cout << "Max Passan: " << max passangers << endl:
int main() {
  PassangerVehicle pv(2, 100, "Honda", 125, 4, 60);
  // error: 'int Vehicle::wheels count' is private
  // cout << "Wheel Count: " << pv.wheels count << endl;
  // error: 'float Vehicle::max speed' is protected
  // cout << "Max Speed: " << pv.max_speed << endl;
  cout << "Manufacturer: " << pv.manufacturer << endl;</pre>
  // error: 'float Automobile::engine cc' is private
  // cout << "Engine Capacity: " << pv.engine_cc << endl;
  // error: 'int Automobile::gear_count' is protected
  // cout << "Gear Count: " << pv.gear_count << endl;
  pv.print();
  return 0;
```

```
class Vehicle {
                                                       class Automobile : protected Vehicle{
  int wheels_count;
                                                         float engine_cc;
protected:
                                                       protected:
  float max speed;
                                                         int gear count;
public:
                                                       public:
  string manufacturer;
                                                          Automobile(int wheels_count, float max_speed,
  Vehicle(int, float, string);
                                                            string manufacturer, float engine_cc,
  void vehicle_print();
                                                            int gear count):
                                                          engine cc(engine cc),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                          gear_count(gear_count),
  string manufacturer):
                                                          Vehicle(wheels_count, max_speed, manufacturer) {
wheels_count(wheels_count),
                                                            cout << "Automobile constructor called\n";
max_speed(max_speed),
manufacturer(manufacturer){
                                                          void automobile print() {
  cout << "Vehicle constructor called\n";
                                                            // error: 'int Vehicle::wheels count' is private
                                                            // cout << "Wheel Count: " << wheels_count << endl;
void Vehicle::vehicle_print() {
                                                            cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Wheel Count: " << wheels count << endl;
                                                            cout << "Manufacturer: " << manufacturer << endl:
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                            cout << "Engine Capacity: " << engine_cc << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl:</pre>
                                                            cout << "Gear Count: " << gear count << endl:
 Vehicle constructor called
 Automobile constructor called
 PassangerVehicle constructor called
 Max Speed: 100
                        Vehicle
                                         Automobile
                                                                PassangerVehicle
                                                                                       pv (object in main)
 Manufacturer: Honda
 Gear Count: 4
                        wheels_count
                                         wheels count
                                                                wheels count
                                                                                       wheels count
 Max Passan: 60
                        max_speed
                                         max_speed
                                                                max_speed
                                                                                       max_speed
                        manufacturer
                                                                manufacturer
                                                                                       manufacturer
                                                                vehicle_print
                        vehicle_print
                                         vehicle_print
                                                                                       vehicle print
                                         engine_cc
                                                                engine cc
                                                                                       engine cc
                                          gear_count
                                                                gear count
                                                                                       gear_count
        Vehicle
                                          automobile_print
                                                                automobile_print
                                                                                       automobile_print
                                                                max_passangers
                                                                                       max_passangers
                                                                print
                                                                                       print
     Automobile
```

PassangerVehicle

```
class PassangerVehicle : public Automobile {
  int max_passangers;
public:
  PassangerVehicle(int wheels count, float max speed,
     string manufacturer, float engine_cc,
     int gear_count, int max_passangers):
  max_passangers(max_passangers),
  Automobile(wheels_count, max_speed, manufacturer,
     engine cc, gear count) {
     cout << "PassangerVehicle constructor called\n";</pre>
  void print() {
     // error: 'int Vehicle::wheels count' is private
     // cout << "Wheel Count: " << wheels count << endl;
     cout << "Max Speed: " << max_speed << endl;
     cout << "Manufacturer: " << manufacturer << endl;
     // error: 'float Automobile::engine_cc' is private
     // cout << "Eng Cap: " << engine_cc << endl;
     cout << "Gear Count: " << gear count << endl;
     cout << "Max Passan: " << max passangers << endl:
int main() {
  PassangerVehicle pv(2, 100, "Honda", 125, 4, 60);
  // error: 'int Vehicle::wheels count' is private
  // cout << "Wheel Count: " << pv.wheels count << endl;
  // error: 'float Vehicle::max speed' is protected
  // cout << "Max Speed: " << pv.max_speed << endl;
  // error: string Vehicle::manufacturer is inaccessible
  // cout << "Manufacturer: " << pv.manufacturer << endl;
  // error: 'float Automobile::engine cc' is private
  // cout << "Engine Capacity: " << pv.engine_cc << endl;
  // error: 'int Automobile::gear_count' is protected
  // cout << "Gear Count: " << pv.gear_count << endl;
  pv.print();
  return 0;
```

```
class Vehicle {
                                                       class Automobile : private Vehicle{
  int wheels_count;
                                                         float engine_cc;
                                                                                                                    int max_passangers;
protected:
                                                       protected:
                                                                                                                  public:
  float max speed;
                                                         int gear count;
public:
                                                       public:
  string manufacturer;
                                                          Automobile(int wheels_count, float max_speed,
  Vehicle(int, float, string);
                                                            string manufacturer, float engine_cc,
  void vehicle_print();
                                                            int gear count):
                                                          engine cc(engine cc),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                          gear_count(gear_count),
  string manufacturer):
                                                          Vehicle(wheels_count, max_speed, manufacturer) {
wheels count(wheels count),
                                                            cout << "Automobile constructor called\n";
                                                                                                                    void print() {
max_speed(max_speed),
manufacturer(manufacturer){
                                                          void automobile print() {
  cout << "Vehicle constructor called\n";</pre>
                                                            // error: 'int Vehicle::wheels count' is private
                                                            // cout << "Wheel Count: " << wheels_count << endl;
void Vehicle::vehicle_print() {
                                                            cout << "Max Speed: " << max speed << endl;
  cout << "Wheel Count: " << wheels count << endl;
                                                            cout << "Manufacturer: " << manufacturer << endl:
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                            cout << "Engine Capacity: " << engine_cc << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl:</pre>
                                                            cout << "Gear Count: " << gear count << endl:
 Vehicle constructor called
 Automobile constructor called
 PassangerVehicle constructor called
                                                                                                                  };
 Gear Count: 4
                        Vehicle
                                         Automobile
                                                                PassangerVehicle
                                                                                       pv (object in main)
 Max Passan: 60
                                                                                                                 int main() {
                                         wheels count
                                                                wheels count
                        wheels count
                                                                                       wheels count
                        max_speed
                                         max_speed
                                                                max_speed
                                                                                       max_speed
                        manufacturer
                                         manufacturer
                                                                manufacturer
                                                                                       manufacturer
                        vehicle_print
                                         vehicle_print
                                                                vehicle_print
                                                                                       vehicle print
                                          engine_cc
                                                                engine cc
                                                                                       engine cc
                                                                gear count
                                          gear_count
                                                                                       gear_count
        Vehicle
                                          automobile_print
                                                                automobile_print
                                                                                       automobile_print
                                                                max_passangers
                                                                                       max_passangers
                                                                print
                                                                                       print
     Automobile
                                                                                                                    pv.print();
 PassangerVehicle
```

```
class PassangerVehicle: public Automobile {
  PassangerVehicle(int wheels count, float max speed,
     string manufacturer, float engine_cc,
     int gear_count, int max_passangers):
  max_passangers(max_passangers),
  Automobile(wheels_count, max_speed, manufacturer,
     engine cc, gear count) {
     cout << "PassangerVehicle constructor called\n";</pre>
     // error: 'int Vehicle::wheels count' is private
     // cout << "Wheel Count: " << wheels count << endl;
     // error: 'float Vahicle::max speed' is protected
     // cout << "Max Speed: " << max_speed << endl;
     // error: 'string Vehicle::manufacturer' is inaccessible
     // cout << "Manufacturer: " << manufacturer << endl:
     // error: 'float Automobile::engine cc' is private
     // cout << "Eng Cap: " << engine_cc << endl;
     cout << "Gear Count: " << gear_count << endl;</pre>
     cout << "Max Passan: " << max passangers << endl;
  PassangerVehicle pv(2, 100, "Honda", 125, 4, 60);
  // error: 'int Vehicle::wheels count' is private
  // cout << "Wheel Count: " << pv.wheels_count << endl;
  // error: 'float Vehicle::max speed' is protected
  // cout << "Max Speed: " << pv.max speed << endl;
  // error: string Vehicle::manufacturer is inaccessible
  // cout << "Manufacturer: " << pv.manufacturer << endl;
  // error: 'float Automobile::engine_cc' is private
  // cout << "Engine Capacity: " << pv.engine_cc << endl;
  // error: 'int Automobile::gear count' is protected
  // cout << "Gear Count: " << pv.gear count << endl;
  return 0;
```

```
class Vehicle {
  int wheels count;
protected:
  float max speed;
public:
  string manufacturer;
  Vehicle(int, float, string);
  void vehicle_print();
Vehicle::Vehicle(int wheels_count, float max_speed,
  string manufacturer):
wheels_count(wheels_count),
max_speed(max_speed),
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";
void Vehicle::vehicle_print() {
  cout << "Wheel Count: " << wheels count << endl;
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl:
```

```
class Automobile : public Vehicle{
  float engine_cc;
protected:
  int gear count;
public:
  Automobile(int wheels_count, float max_speed,
     string manufacturer, float engine_cc,
    int gear count):
  engine cc(engine cc),
  gear_count(gear_count),
  Vehicle(wheels_count, max_speed, manufacturer) {
     cout << "Automobile constructor called\n";
  void automobile print() {
     // error: 'int Vehicle::wheels count' is private
     // cout << "Wheel Count: " << wheels_count << endl;
     cout << "Max Speed: " << max_speed << endl;
     cout << "Manufacturer: " << manufacturer << endl;</pre>
     cout << "Engine Capacity: " << engine_cc << endl;</pre>
     cout << "Gear Count: " << gear count << endl:
};
```

```
Vehicle
                v (object in main)
                                     Automobile
                                                           a (object in main)
wheels count
                                     wheels count
                                                           ?
max_speed
                                     max_speed
manufacturer
                                     manufacturer
vehicle_print
                                     vehicle_print
                                     engine cc
                                     gear_count
                                     automobile_print
```

```
Vehicle

↑
Automobile

↑
PassangerVehicle
```

```
int main() {
    Vehicle v(2, 100, "Honda");
    cout << "Wheel Count: " << v.wheels_count << endl;
    cout << "Max Speed: " << v.max_speed << endl;
    cout << "Manufacturer: " << v.manufacturer << endl;

Automobile a(2, 100, "Honda", 125, 4);
    cout << "Wheel Count: " << a.wheels_count << endl;
    cout << "Max Speed: " << a.max_speed << endl;
    cout << "Manufacturer: " << a.manufacturer << endl;
    cout << "Engine Capacity: " << a.engine_cc << endl;
    cout << "Gear Count: " << a.gear_count << endl;
    return 0;
}</pre>
```

```
class Vehicle {
  int wheels_count;
protected:
  float max_speed;
public:
  string manufacturer;
  Vehicle(int, float, string);
  void print();
  void print(bool);
Vehicle::Vehicle(int wheels_count, float max_speed,
  string manufacturer):
wheels_count(wheels_count),
max speed(max speed),
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";</pre>
void Vehicle::print() {
  cout << "Wheel Count: " << wheels_count << endl;
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl:
void Vehicle::print(bool in_single_line = true) {
  if(in_single_line) {
     cout << wheels count << " " << max speed;
     cout << " " << manufacturer << endl;
  else {
     cout << wheels count << endl << max speed;
     cout << endl << manufacturer << endl;
```

```
int main() {
    Vehicle vehicle(2, 100, "Honda");
    vehicle.print();
    vehicle.print(true);
    vehicle.print(false);
    return 0;
}
```

Guess the output of this code

```
class Vehicle {
  int wheels_count;
protected:
  float max_speed;
public:
  string manufacturer;
  Vehicle(int, float, string);
  void print();
  void print(bool);
Vehicle::Vehicle(int wheels_count, float max_speed,
  string manufacturer):
wheels_count(wheels_count),
max speed(max speed),
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";</pre>
void Vehicle::print() {
  cout << "Wheel Count: " << wheels_count << endl;</pre>
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl:
void Vehicle::print(bool in_single_line = true) {
  if(in_single_line) {
     cout << wheels count << " " << max speed;
     cout << " " << manufacturer << endl;
  else {
     cout << wheels_count << endl << max_speed;</pre>
     cout << endl << manufacturer << endl;
```

```
int main() {
  Vehicle vehicle(2, 100, "Honda");
  // error: call of overloaded 'print()' is ambiguous
  vehicle.print();
  vehicle.print(true);
  vehicle.print(false);
  return 0;
```

```
class Vehicle {
  int wheels_count;
protected:
  float max_speed;
public:
  string manufacturer;
  Vehicle(int, float, string);
  void print();
  void print(bool);
Vehicle::Vehicle(int wheels_count, float max_speed,
  string manufacturer):
wheels_count(wheels_count),
max speed(max speed),
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";</pre>
void Vehicle::print() {
  cout << "Wheel Count: " << wheels_count << endl;
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl:
void Vehicle::print(bool in_single_line) {
  if(in_single_line) {
     cout << wheels count << " " << max speed;
     cout << " " << manufacturer << endl;
  else {
     cout << wheels count << endl << max speed;
     cout << endl << manufacturer << endl;
```

```
int main() {
    Vehicle vehicle(2, 100, "Honda");
    vehicle.print();
    vehicle.print(true);
    vehicle.print(false);
    return 0;
}
```

Guess the output of this code

```
class Vehicle {
  int wheels count:
protected:
  float max_speed;
public:
  string manufacturer;
  Vehicle(int, float, string);
  void print();
  void print(bool);
Vehicle::Vehicle(int wheels_count, float max_speed,
  string manufacturer):
wheels_count(wheels_count),
max speed(max speed),
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";</pre>
void Vehicle::print() {
  cout << "Wheel Count: " << wheels_count << endl;
  cout << "Max Speed: " << max_speed << endl;</pre>
  cout << "Manufacturer: " << manufacturer << endl:
void Vehicle::print(bool in_single_line) {
  if(in_single_line) {
     cout << wheels count << " " << max speed;
     cout << " " << manufacturer << endl;
  else {
     cout << wheels count << endl << max speed;
     cout << endl << manufacturer << endl;
```

```
int main() {
    Vehicle vehicle(2, 100, "Honda");
    vehicle.print();
    vehicle.print(true);
    vehicle.print(false);
    return 0;
}
```

Vehicle constructor called

Wheel Count: 2

Max Speed: 100

Manufacturer: Honda

2 100 Honda

2

100

Honda

```
class Vehicle {
                                                         class Automobile : public Vehicle{
  int wheels_count;
                                                            float engine_cc;
protected:
                                                         protected:
  float max_speed;
                                                            int gear count;
public:
                                                         public:
  string manufacturer;
                                                            Automobile(int wheels_count, float max_speed,
  Vehicle(int, float, string);
                                                               string manufacturer, float engine_cc,
  void print(bool);
                                                              int gear count):
                                                            engine_cc(engine_cc),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                            gear_count(gear_count),
  string manufacturer):
                                                            Vehicle(wheels_count, max_speed, manufacturer) {
wheels_count(wheels_count),
                                                              cout << "Automobile constructor called\n";</pre>
max_speed(max_speed),
manufacturer(manufacturer){
                                                            void print() {
  cout << "Vehicle constructor called\n";</pre>
                                                              cout << max_speed << endl << manufacturer << endl;</pre>
                                                              cout << engine_cc << endl << gear_count << endl;
void Vehicle::print(bool in_single_line) {
  if(in_single_line) {
                                                         };
     cout << wheels_count << " " << max_speed;
     cout << " " << manufacturer << endl;
                                                         int main() {
                                                            Automobile automobile(2, 100, "Honda", 125, 4);
  else {
                                                            automobile.print();
     cout << wheels_count << endl << max_speed;
                                                            return 0;
     cout << endl << manufacturer << endl:
```

Vehicle constructor called Automobile constructor called 100 Honda 125

```
class Vehicle {
                                                          class Automobile : public Vehicle{
  int wheels_count;
                                                             float engine_cc;
protected:
                                                          protected:
  float max_speed;
                                                             int gear_count;
public:
                                                          public:
  string manufacturer;
                                                             Automobile(int wheels_count, float max_speed,
  Vehicle(int, float, string);
                                                               string manufacturer, float engine_cc,
  void print(bool);
                                                               int gear count):
                                                             engine_cc(engine_cc),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                             gear_count(gear_count),
  string manufacturer):
                                                             Vehicle(wheels_count, max_speed, manufacturer) {
wheels_count(wheels_count),
                                                               cout << "Automobile constructor called\n";</pre>
max_speed(max_speed),
manufacturer(manufacturer){
                                                             void print() {
  cout << "Vehicle constructor called\n";
                                                               cout << max_speed << endl << manufacturer << endl;</pre>
                                                               cout << endl << gear_count << endl;</pre>
void Vehicle::print(bool in_single_line) {
  if(in_single_line) {
                                                           };
     cout << wheels_count << " " << max_speed;
     cout << " " << manufacturer << endl;
                                                          int main() {
                                                             Automobile automobile(2, 100, "Honda", 125, 4);
  else {
                                                             automobile.print(true);
     cout << wheels_count << endl << max_speed;
                                                             return 0;
     cout << endl << manufacturer << endl:
```

Guess the output

```
class Vehicle {
                                                           class Automobile : public Vehicle{
  int wheels count:
                                                             float engine_cc;
protected:
                                                           protected:
  float max_speed;
                                                             int gear_count;
public:
                                                           public:
  string manufacturer;
                                                             Automobile(int wheels_count, float max_speed,
  Vehicle(int, float, string);
                                                                string manufacturer, float engine_cc,
  void print(bool);
                                                                int gear count):
                                                             engine_cc(engine_cc),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                             gear_count(gear_count),
  string manufacturer):
                                                             Vehicle(wheels_count, max_speed, manufacturer) {
wheels_count(wheels_count),
                                                                cout << "Automobile constructor called\n";</pre>
max_speed(max_speed),
manufacturer(manufacturer){
                                                             void print() {
  cout << "Vehicle constructor called\n";</pre>
                                                                cout << max_speed << endl << manufacturer << endl;</pre>
                                                                cout << endl << gear_count << endl;</pre>
void Vehicle::print(bool in_single_line) {
  if(in_single_line) {
                                                           };
     cout << wheels_count << " " << max_speed;
     cout << " " << manufacturer << endl;
                                                           int main() {
                                                             Automobile automobile(2, 100, "Honda", 125, 4);
  else {
                                                             // error: no matching function for call to
     cout << wheels_count << endl << max_speed;
                                                             // 'Automobile::print(bool)
     cout << endl << manufacturer << endl:
                                                             automobile.print(true);
                                                             return 0;
```

- Functions/methods can be overloaded within the same scope only
- Here Vehicle and Automobile are two differenct classes and hence different scopes
- When we define print function in Automobile then all the print functions of the Vehicle class are inaccessible in Automobile class

```
class Vehicle {
                                                          class Automobile : public Vehicle{
  int wheels_count;
                                                            float engine_cc;
protected:
                                                          protected:
  float max_speed;
                                                            int gear_count;
public:
                                                          public:
  string manufacturer;
                                                            Automobile(int wheels_count, float max_speed,
  Vehicle(int, float, string);
                                                               string manufacturer, float engine_cc,
  void print(bool);
                                                               int gear count):
  void print();
                                                            engine_cc(engine_cc),
                                                            gear_count(gear_count),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                            Vehicle(wheels_count, max_speed, manufacturer) {
  string manufacturer):
                                                               cout << "Automobile constructor called\n";</pre>
wheels_count(wheels_count),
max speed(max speed),
                                                            void print() {
manufacturer(manufacturer){
                                                               cout << max_speed << endl << manufacturer << endl;</pre>
  cout << "Vehicle constructor called\n";</pre>
                                                               cout << endl << gear_count << endl;</pre>
void Vehicle::print(bool in_single_line) {
                                                          };
  if(in_single_line) {
     cout << wheels_count << " " << max_speed;
                                                          int main() {
     cout << " " << manufacturer << endl:
                                                            Automobile automobile(2, 100, "Honda", 125, 4);
                                                            automobile.print();
  else {
                                                            return 0;
     cout << wheels_count << endl << max_speed;
     cout << endl << manufacturer << endl;
                                                                                                            Vehicle constructor called
void Vehicle::print() {
                                                                                                            Automobile constructor
  cout << "Wheel Count: " << wheels count << endl;
                                                                                                            called
  cout << "Max Speed: " << max speed << endl;
                                                                                                             100
  cout << "Manufacturer: " << manufacturer << endl;</pre>
                                                                                                            Honda
                                                                                                             125
```

4

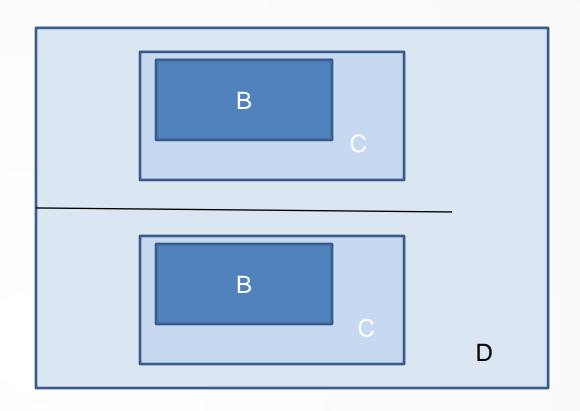
```
class Vehicle {
                                                           class Automobile : public Vehicle{
  int wheels_count;
                                                             float engine_cc;
protected:
                                                          protected:
  float max_speed;
                                                             int gear_count;
public:
                                                          public:
  string manufacturer;
                                                             Automobile(int wheels_count, float max_speed,
  Vehicle(int, float, string);
                                                               string manufacturer, float engine_cc,
  void print(bool);
                                                               int gear count):
  void print();
                                                             engine_cc(engine_cc),
                                                             gear_count(gear_count),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                             Vehicle(wheels_count, max_speed, manufacturer) {
  string manufacturer):
                                                               cout << "Automobile constructor called\n";</pre>
wheels_count(wheels_count),
max speed(max speed),
                                                             void print(bool in_single_line) {
manufacturer(manufacturer){
                                                               if(in_single_line) {
  cout << "Vehicle constructor called\n";</pre>
                                                                  cout << max_speed << " " << manufacturer;</pre>
                                                                  cout << " " << engine_cc << " " << gear_count;
void Vehicle::print(bool in_single_line) {
                                                                  cout << endl;
  if(in_single_line) {
     cout << wheels_count << " " << max_speed;
                                                               else {
     cout << " " << manufacturer << endl:
                                                                  cout << max speed << endl << manufacturer;
                                                                  cout << endl << engine_cc <<endl << gear_count;</pre>
  else {
                                                                  cout << endl;
    cout << wheels_count << endl << max_speed;</pre>
     cout << endl << manufacturer << endl;
void Vehicle::print() {
                                                          int main() {
  cout << "Wheel Count: " << wheels_count << endl;</pre>
                                                             Automobile automobile(2, 100, "Honda", 125, 4);
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                             automobile.print(true);
                                                                                                               Vehicle constructor called
  cout << "Manufacturer: " << manufacturer << endl;</pre>
                                                             return 0;
                                                                                                               Automobile constructor called
                                                                                                               100 Honda 125 4
```

```
class Vehicle {
                                                       class Automobile : public Vehicle{
                                                                                                   int main() {
  int wheels count:
                                                         float engine_cc;
                                                                                                      Automobile automobile(2, 100, "Honda", 125, 4);
protected:
                                                         float wheels_count = 16;
                                                                                                      automobile.print(true);
  float max_speed;
                                                       protected:
                                                                                                      return 0;
public:
                                                         int gear_count;
  string manufacturer;
                                                         int max speed = 1236; // supersonic
  Vehicle(int, float, string);
                                                       public:
  void print(bool);
                                                         string manufacturer = "Magic Corp.";
  void print();
                                                         Automobile(int wheels count, float max speed,
                                                            string manufacturer, float engine_cc,
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                            int gear_count):
  string manufacturer):
                                                         engine_cc(engine_cc),
wheels_count(wheels_count),
                                                         gear_count(gear_count),
max speed(max speed),
                                                          Vehicle(wheels_count, max_speed, manufacturer) {
manufacturer(manufacturer){
                                                            cout << "Automobile constructor called\n":
  cout << "Vehicle constructor called\n";</pre>
                                                         void print(bool in_single_line) {
void Vehicle::print(bool in_single_line) {
                                                            if(in_single_line) {
  if(in_single_line) {
                                                              cout << wheels_count << " ";
     cout << wheels_count << " " << max_speed;
                                                              cout << max_speed << " " << manufacturer;</pre>
     cout << " " << manufacturer << endl:
                                                              cout << " " << engine cc << " " << gear count;
                                                              cout << endl:
  else {
    cout << wheels_count << endl << max_speed;</pre>
                                                            else {
     cout << endl << manufacturer << endl;
                                                              cout << wheels count << endl;
                                                              cout << max_speed << endl << manufacturer;</pre>
                                                              cout << endl << engine_cc <<endl << gear_count;
void Vehicle::print() {
                                                              cout << endl;
  cout << "Wheel Count: " << wheels count << endl;
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                                                                              Vehicle constructor called
  cout << "Manufacturer: " << manufacturer << endl;</pre>
                                                                                                              Automobile constructor called
                                                                                                              16 1236 Magic Corp. 125 4
```

```
class Automobile: public Vehicle{
                                                                                              int main() {
class Vehicle {
                                                          float engine cc;
                                                                                                Automobile automobile(2, 100, "Honda", 125, 4);
  int wheels_count;
                                                          float wheels_count = 16;
                                                                                                automobile.print(true);
protected:
                                                       protected:
                                                                                                cout << automobile.manufacturer << endl;
  float max_speed;
                                                                                                cout << automobile.Vehicle::manufacturer << endl:</pre>
                                                          int gear_count;
public:
                                                          int max_speed = 1236;
                                                                                                return 0;
  string manufacturer;
                                                       public:
  Vehicle(int, float, string);
                                                          string manufacturer = "Magic Corp.";
  void print(bool);
                                                          Automobile(int wheels_count, float max_speed,
  void print();
                                                            string manufacturer, float engine_cc,
                                                            int gear_count):
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                          engine_cc(engine_cc),
  string manufacturer):
                                                          gear_count(gear_count),
wheels_count(wheels_count),
                                                          Vehicle(wheels count, max speed, manufacturer) {
max speed(max speed),
                                                            cout << "Automobile constructor called\n";</pre>
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";</pre>
                                                          void print(bool in_single_line) {
                                                            if(in_single_line) {
void Vehicle::print(bool in_single_line) {
                                                               // error: 'int Vehicle::wheels_count' is private
  if(in_single_line) {
                                                                                                                 Vehicle constructor called
                                                               // cout << Vehicle::wheels_count << " ";
    cout << wheels_count << " " << max_speed;</pre>
                                                                                                                 Automobile constructor called
                                                               cout << Vehicle::max_speed << " ";</pre>
     cout << " " << manufacturer << endl:
                                                               cout << Vehicle::manufacturer;</pre>
                                                                                                                 100 Honda 125 4
                                                               cout << " " << engine_cc << " " << gear_count;
  else {
                                                                                                                 Magic Corp.
                                                               cout << endl;
     cout << wheels_count << endl << max_speed;
                                                                                                                 Honda
    cout << endl << manufacturer << endl;</pre>
                                                            } else {
                                                               // error: 'int Vehicle::wheels_count' is private
                                                               // cout << Vehicle::wheels count << endl;
                                                               cout << Vehicle::max_speed << endl;</pre>
void Vehicle::print() {
                                                               cout << Vehicle::manufacturer;
  cout << "Wheel Count: " << wheels count << endl;
                                                               cout << endl << engine_cc << endl << gear_count;
  cout << "Max Speed: " << max_speed << endl;</pre>
                                                               cout << endl;
  cout << "Manufacturer: " << manufacturer << endl;
```

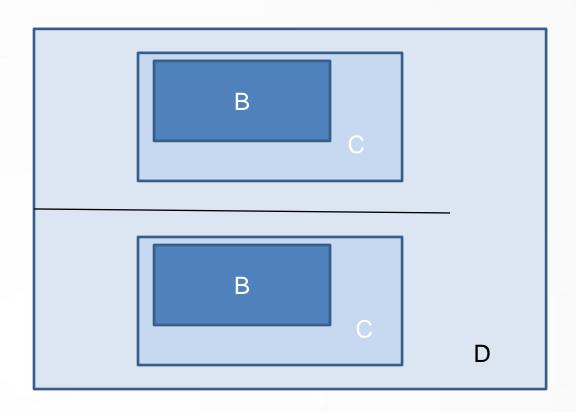
```
class Vehicle {
                                                          class Automobile : public Vehicle{
  int wheels_count;
                                                            float engine_cc;
protected:
                                                          protected:
  float max_speed;
                                                            int gear_count;
public:
                                                          public:
  string manufacturer;
                                                            Automobile(int wheels_count, float max_speed,
  Vehicle(int, float, string);
                                                              string manufacturer, float engine_cc,
  void print(bool);
                                                              int gear count):
                                                            engine_cc(engine_cc),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                            gear_count(gear_count),
  string manufacturer):
                                                            Vehicle(wheels_count, max_speed, manufacturer) {
wheels_count(wheels_count),
                                                              cout << "Automobile constructor called\n";</pre>
max_speed(max_speed),
manufacturer(manufacturer){
                                                            void print() {
  cout << "Vehicle constructor called\n";
                                                              Vehicle::print(false);
                                                              cout << endl << gear_count << endl;</pre>
void Vehicle::print(bool in_single_line) {
  if(in_single_line) {
                                                          };
     cout << wheels_count << " " << max_speed;
     cout << " " << manufacturer << endl;
                                                          int main() {
                                                            Automobile automobile(2, 100, "Honda", 125, 4);
  else {
                                                            // error: no matching function for call to
     cout << wheels_count << endl << max_speed;
                                                            // 'Automobile::print(bool)
     cout << endl << manufacturer << endl:
                                                            // automobile.print(true);
                                                                                                            Vehicle constructor called
                                                            automobile.Vehicle::print(true);
                                                            automobile.print();
                                                                                                            Automobile constructor called
                                                            return 0;
                                                                                                            2 100 Honda
                                                                                                             100
                                                                                                            Honda
                                                                                                            125
```

```
class B {
protected:
B() { cout << "B "; }
~B() { cout << "~B "; }
};
class C : public B {
protected:
C() { cout << "C "; }
~C() { cout << "~C "; }
class D : private C {
C var;
public:
D() { cout << "D " << endl; }
~D() { cout << "~D "; }
};
int main() {
Dd;
return 0;
```



Output: ??

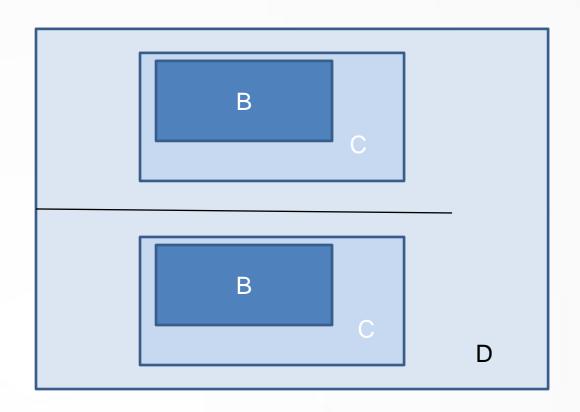
```
class B {
protected:
B() { cout << "B "; }
~B() { cout << "~B "; }
};
class C : public B {
protected:
C() { cout << "C "; }
~C() { cout << "~C "; }
class D : private C {
C var;
public:
D() { cout << "D " << endl; }
~D() { cout << "~D "; }
};
int main() {
Dd;
return 0;
```



Output: Error

var can not access constructor which is protected

```
class B {
protected:
B() { cout << "B "; }
~B() { cout << "~B "; }
};
class C : public B {
public:
C() { cout << "C "; }
~C() { cout << "~C "; }
class D : private C {
C var;
public:
D() { cout << "D " << endl; }
~D() { cout << "~D "; }
};
int main() {
Dd;
return 0;
```



```
class B {
protected:
B() { cout << "B "; }
~B() { cout << "~B "; }
};
class C : public B {
protected:
int data=100;
public:
C() { cout << "C "; }
~C() { cout << "~C "; }
class D : private C {
C var;
public:
D() { cout << "D " << endl; }
~D() { cout << "~D "; }
void change_data();
void print();
};
```

```
int main() {
D d;
d.change_data();
d.print();
return 0;
}
```

```
Output:
```

```
В
В
                      D
```

```
void D:: change_data()
    {
        data=100; //update the data of inherited C
        var.data=500; //update the data of var C
    }
    void D:: print()
    {
        cout<<endl<<data<<endl;
        cout<<var.data<<endl;
    }
}</pre>
```

```
class B {
protected:
B() { cout << "B "; }
~B() { cout << "~B "; }
};
class C : public B {
protected:
int data=100;
public:
C() { cout << "C "; }
~C() { cout << "~C "; }
class D : private C {
C var;
public:
D() { cout << "D " << endl; }
~D() { cout << "~D "; }
void change_data();
void print();
};
```

```
int main() {
D d;
d.change_data();
d.print();
return 0;
}
```

Output:
Error:
var.data is private
and not accessible
in change_data()
and in print()

To allow modification of data using var, we can create pubilc method in class C which modifies data and call it using var

```
B C D
```

```
void D:: change_data()
{
    data=100; //update the data of inherited C
    var.data=500; //ERROR: access not allowed
}
void D:: print()
{
    cout<<endl<<data<<endl;
    cout<<var.data<<endl; //ERROR: access not allowed
}</pre>
```

Inheritance

- Constructors, destructor, friend functions, friend classes, and static members
 of base class are not inherited by derived class.
- •Friend fuctions and classes have access to protected members of the class along with private and public members of the class

```
class Vehicle {
                                                     class Automobile : public Vehicle{
                                                                                                              void friend_fun1() {
  int wheels_count;
                                                                                                                 cout << "friend fun1 called\n";
                                                        float engine_cc;
protected:
                                                                                                                 Automobile automobile(2, 100, "Honda", 125, 4);
                                                     protected:
  float max speed;
                                                                                                                 cout << automobile.wheels count << endl;
                                                        int gear count;
public:
                                                                                                                 cout << automobile.max_speed << endl;</pre>
                                                     public:
  string manufacturer;
                                                        friend void friend_fun2();
                                                                                                                 cout << automobile.manufacturer << endl;
  Vehicle(int, float, string);
                                                        Automobile(int wheels_count, float max_speed,
                                                                                                                // error: 'float Automobile::engine_cc' is private
  void print(bool);
                                                          string manufacturer, float engine cc,
                                                                                                                // cout << automobile.engine cc << endl;
  friend void friend fun1():
                                                                                                                // error: 'int Automobile::gear count' is protected
                                                          int gear count):
                                                        engine_cc(engine_cc),
                                                                                                                // cout << automobile.gear_count << endl;
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                        gear_count(gear_count),
  string manufacturer):
                                                        Vehicle(wheels_count, max_speed, manufacturer) {
wheels_count(wheels_count),
                                                          cout << "Automobile constructor called\n":
                                                                                                              void friend fun2() {
max_speed(max_speed),
                                                                                                                 cout << "friend fun2 called\n";</pre>
manufacturer(manufacturer){
                                                        void print() {
                                                                                                                Automobile automobile(2, 100, "Honda", 125, 4);
  cout << "Vehicle constructor called\n";
                                                          Vehicle::print(false);
                                                                                                                // error: 'int Vehicle::wheels_count' is private
                                                                                                                // cout << automobile.wheels count << endl;
                                                          cout << engine_cc << endl << gear_count << endl;</pre>
void Vehicle::print(bool in_single_line) {
                                                                                                                cout << automobile.max_speed << endl;</pre>
  if(in_single_line) {
                                                     };
                                                                                                                cout << automobile.manufacturer << endl;
    cout << wheels_count << " " << max_speed;</pre>
                                                                                                                cout << automobile.engine_cc << endl;</pre>
    cout << " " << manufacturer << endl;
                                                         friend fun1 called
                                                                                                                 cout << automobile.gear_count << endl;
                                                          Vehicle constructor called
  else {
    cout << wheels_count << endl << max_speed;
                                                          Automobile constructor called
                                                                                                              int main() {
    cout << endl << manufacturer << endl:
                                                                                                                friend fun1();
                                                          2
                                                                                                                friend_fun2();
                                                          100
                                                                                                                 return 0;
                                                         Honda
                                                         friend fun2 called
                                                          Vehicle constructor called
                                                          Automobile constructor called
                                                          100
                                                          Honda
                                                          125
```

```
class Vehicle {
                                                      class Automobile: public Vehicle{
                                                                                                               void friend_fun1() {
  int wheels count;
                                                                                                                 cout << "friend fun1 called\n";
                                                        float engine_cc;
protected:
                                                                                                                 Automobile automobile(2, 100, "Honda", 125, 4);
                                                      protected:
  float max speed;
                                                        int gear count;
public:
                                                      public:
                                                                                                                 cout << automobile.wheels count << endl;
  string manufacturer;
                                                        friend void friend_fun1();
                                                                                                                 cout << automobile.max_speed << endl;
  Vehicle(int, float, string);
                                                                                                                 cout << automobile.manufacturer << endl;
                                                        Automobile(int wheels_count, float max_speed,
  void print(bool);
                                                          string manufacturer, float engine cc,
                                                                                                                 cout << automobile.engine cc << endl;
  friend void friend_fun1();
                                                          int gear count):
                                                                                                                 cout << automobile.gear count << endl;
                                                        engine_cc(engine_cc),
Vehicle::Vehicle(int wheels_count, float max_speed,
                                                                                                                 Vehicle vehicle(3, 70, "Bajaj");
                                                        gear_count(gear_count),
  string manufacturer):
                                                        Vehicle(wheels_count, max_speed, manufacturer) {
wheels_count(wheels_count),
                                                          cout << "Automobile constructor called\n":
                                                                                                                 cout << endl<<vehicle.wheels count << endl;
max speed(max speed),
                                                                                                                 cout << vehicle.max speed << endl;
manufacturer(manufacturer){
                                                        void print() {
                                                                                                                 cout << vehicle.manufacturer << endl;
  cout << "Vehicle constructor called\n";</pre>
                                                          Vehicle::print(false);
                                                                                                                //error: 'class Vehicle' has no member named 'engine_cc'
                                                          cout << engine_cc << endl << gear_count << endl;
void Vehicle::print(bool in_single_line) {
                                                                                                                // cout << vehicle.engine cc << endl;
  if(in_single_line) {
                                                      };
    cout << wheels_count << " " << max_speed;
    cout << " " << manufacturer << endl;
                                                          friend_fun1 called
                                                                                                               int main() {
                                                                                                                 friend fun1();
                                                          Vehicle constructor called
  else {
                                                                                                                 return 0;
                                                          Automobile constructor called
    cout << wheels_count << endl << max_speed;</pre>
    cout << endl << manufacturer << endl;
                                                          2
                                                          100
                                                          Honda
                                                          125
                                                          4
                                                          3
                                                          70
                                                          Bajaj
```

```
class Vehicle {
  int wheels count;
protected:
  float max speed;
public:
  string manufacturer;
  Vehicle(int, float, string);
  void vehicle_print();
  ~Vehicle() {
     cout << "Vehicle Destructor called\n";
Vehicle::Vehicle(int wheels count, float max speed,
  string manufacturer):
wheels_count(wheels_count),
max speed(max speed),
manufacturer(manufacturer){
  cout << "Vehicle constructor called\n";
void Vehicle::vehicle_print() {
  cout << "Wheel Count: " << wheels_count << endl;
  cout << "Max Speed: " << max speed << endl;
  cout << "Manufacturer: " << manufacturer << endl;</pre>
 Vehicle constructor called
 Engine constructor called
 Automobile constructor called
 Wheel Count: 4
 Max Speed: 200
 Manufacturer: Tata
 cyl cnt: 4
 cyl arr: V-shape
 eng cc: 1200
 Gear Count: 6
 Automobile Destructor called
 Engine Destructor called
 Vehicle Destructor called
```

```
class Engine {
  int cylinders_count;
protected:
  string cylinders arrangement; //v-type, inline etc...
public:
  float engine_cc;
  Engine(int cylinders_count,
     string cylinders_arrangement, float engine_cc):
  cylinders count(cylinders count),
  cylinders_arrangement(cylinders_arrangement),
  engine_cc(engine_cc){
     cout << "Engine constructor called\n";
  ~Engine() {
     cout << "Engine Destructor called\n";</pre>
  void print() {
     cout << "cyl cnt: " << cylinders_count << endl;</pre>
     cout << "cyl arr: " << cylinders arrangement;</pre>
     cout << endl << "eng cc: " << engine cc << endl:
};
```

```
class Automobile : public Vehicle{
  Engine e;
protected:
  int gear_count;
public:
  Automobile(int wheels_count, float max_speed,
     string manufacturer, int cylinders_count,
     string cylinders_arrangement, float engine_cc,
     int gear count):
  gear_count(gear_count),
  e(cylinders_count, cylinders_arrangement,
     engine_cc),
  Vehicle(wheels_count, max_speed, manufacturer) {
     cout << "Automobile constructor called\n";</pre>
  ~Automobile() {
     cout << "Automobile Destructor called\n";
  void print() {
     vehicle_print();
     e.print();
     cout << "Gear Count: " << gear count << endl;
};
int main() {
  Automobile a(4, 200, "Tata", 4, "V-shape", 1200, 6);
  a.print();
  return 0;
```

Different types of inheritance

```
class A {
protected:
  int num1 = 1;
};
class B {
protected:
  int num2 = 2;
class C: public A, public B {
protected:
  int num3 = 3;
public:
  void print() {
     cout << num1 << num2 << num3;</pre>
int main() {
  C c;
  c.print();
  return 0;
```

```
class A {
protected:
  int num1 = 1;
public:
  A() {
     cout << "A constructor called\n";</pre>
  ~A() {
     cout << "A destructor called\n";</pre>
};
class B {
protected:
  int num2 = 2;
public:
   B() {
     cout << "B constructor called\n";</pre>
  ~B() {
     cout << "B destructor called\n";</pre>
```

```
class C: public A, public B {
protected:
  int num3 = 3;
public:
  C() {
     cout << "C constructor called\n";</pre>
  ~C() {
     cout << "C destructor called\n";</pre>
  void print() {
     cout << num1 << num2 << num3 << endl;</pre>
};
int main() {
  C c;
  c.print();
  return 0;
                                             A constructor called
                                             B constructor called
                                             C constructor called
                                             123
                                             C destructor called
```

B destructor called

A destructor called

class A {	class C: public B, public A {		
orotected:	protected:		
int num1;	int num3 = 3;		
oublic:	public:		
A(int n): num1(n) {	B b;		
cout << "A constructor called\n";	C(int num1, int num2, int num3, int num4):		
}	A(num2),		
~A() {	B(num1),		
cout << "A destructor called\n";	num3(num3),		
}	b(num4) {		
	cout << "C constructor calle	ed\n";	
	}		
class B {	~C() {		
protected:	cout << "C destructor called\n";		
int $num2 = 2$;	}		
oublic:	<pre>void print() {</pre>		
B(int n): num2(n) {	cout << num1 << num2 <<	num3;	
<pre>cout << "B constructor called\n";</pre>	}	B constructor called	
}	} ;	A constructor called	
~B() {		B constructor called	
<pre>cout << "B destructor called\n";</pre>	int main() {	C constructor called	
}	C c(1, 3, 2, 4);	3124	
<pre>void print() {</pre>	c.print();	C destructor called	
cout << num2 << endl;	c.b.print();	B destructor called	
}	return 0;	A destructor called	
;	}	B destructor called	

class A {	class C: public B, public A {		
protected:	protected:		
int num1;	int num3;		
oublic:	public:		
A(int n): num1(n) {	B b;		
<pre>cout << "A constructor called\n";</pre>	C(int num1, int num2, int num3, int num4):		
}	A(num2),		
~A() {	B(num1),		
<pre>cout << "A destructor called\n";</pre>	num3(num3),		
}	b(num4) {		
	cout << "C constructor call	ed\n";	
	}		
class B {	~C() {		
protected:	cout << "C destructor called\n";		
int num2;	}		
oublic:	void print() {		
B(int n): num2(n) {	cout << num1 << num2 <<	num3;	
cout << "B constructor called\n";	}	B constructor called	
}	} ;	A constructor called	
~B() {		B constructor called	
cout << "B destructor called\n";	int main() {	C constructor called	
}	C c(1, 3, 2, 4);	3124	
<pre>void print() {</pre>	c.print();	C destructor called	
cout << num2 << endl;	c.b.print();	B destructor called	
}	return 0;	A destructor called	
,	}	B destructor called	

```
class C: public B, public A {
class A {
                                                         protected:
protected:
                                                           int num3 = 1;
  int num1;
                                                           int num4;
public:
                                                         public:
  A(int n): num1(n) {
                                                           B b;
     cout << "A constructor called\n";
                                                           C(int n1, int n2, int n3, int n4):
                                                           A(n2)
  ~A() {
                                                           B(n1),
                                                           num4(num3 * 2),
     cout << "A destructor called\n";
                                                           num3(n3),
                                                           b(n4) {
};
                                                             cout << "C constructor called\n":
class B {
                                                           ~C() {
protected:
                                                              cout << "C destructor called\n";
  int num2;
public:
                                                           void print() {
   B(int n): num2(n) {
                                                              cout << num1 << num2 << num4;
     cout << "B constructor called\n";
                                                         int main() {
  ~B() {
                                                           C c(1, 3, 2, 4);
     cout << "B destructor called\n";
                                                           c.print();
                                                           c.b.print();
  void print() {
                                                           return 0;
     cout << num2 << endl;
```

B constructor called

A constructor called

B constructor called

C constructor called

C destructor called

B destructor called

A destructor called

B destructor called

31244

```
class C: public B, public A {
class A {
                                                           protected:
protected:
                                                              int num3 = 1;
  int num1;
                                                              int num4;
public:
                                                           public:
  A(int n): num1(n) {
                                                              Bb;
     cout << "A constructor called\n";</pre>
                                                              C(int n1, int n2, int n3, int n4):
                                                              A(n2),
  ~A() {
                                                              B(n1),
     cout << "A destructor called\n";</pre>
                                                              num4(num3 * 2),
                                                              num3(n3),
                                                              b(n4) {
};
                                                                cout << "C constructor called\n";</pre>
class B {
                                                              ~C() {
protected:
                                                                cout << "C destructor called\n";
  int num1;
public:
                                                              void print() {
                                                                // error: reference to 'num1' is ambiguous
   B(int n): num1(n) {
                                                                cout << num1 << num1 << num4;</pre>
     cout << "B constructor called\n";</pre>
   ~B() {
                                                           int main() {
     cout << "B destructor called\n";</pre>
                                                              C c(1, 3, 2, 4);
                                                              c.print();
  void print() {
                                                              c.b.print();
     cout << num1 << endl;
                                                              return 0;
```

```
class C: public B, public A {
class A {
                                                         protected:
protected:
                                                           int num3 = 1;
  int num1;
                                                           int num4;
public:
                                                         public:
  A(int n): num1(n) {
                                                           B b;
     cout << "A constructor called\n";
                                                           C(int n1, int n2, int n3, int n4):
                                                           A(n2)
                                                                                                       B constructor called
  ~A() {
                                                           B(n1),
                                                                                                      A constructor called
                                                           num4(num3 * 2),
     cout << "A destructor called\n";
                                                                                                      B constructor called
                                                           num3(n3),
                                                                                                       C constructor called
                                                           b(n4) {
};
                                                                                                      31244
                                                              cout << "C constructor called\n":
                                                                                                       C destructor called
class B {
                                                                                                      B destructor called
                                                           ~C() {
protected:
                                                              cout << "C destructor called\n";
                                                                                                      A destructor called
  int num1;
                                                                                                       B destructor called
public:
                                                           void print() {
                                                              cout << A::num1 << B::num1 << num3 << num4;</pre>
  B(int n): num1(n) {
     cout << "B constructor called\n";
                                                         int main() {
  ~B() {
                                                           C c(1, 3, 2, 4);
     cout << "B destructor called\n";
                                                           c.print();
                                                           c.b.print();
  void print() {
                                                           return 0;
     cout << num1 << endl;
```

```
class A {
public:
  int num1 = 1;
  A(){
     cout << "A constructor called\n";</pre>
  ~A() {
     cout << "A destructor called\n";</pre>
class B: public A {
public:
  B(){
     cout << "B constructor called\n";</pre>
  ~B() {
     cout << "B destructor called\n";</pre>
class C: public A {
public:
  C(){
     cout << "C constructor called\n";</pre>
  ~C() {
     cout << "C destructor called\n";</pre>
};
```

```
class D: public B, public C {
public:
  D(){
     cout << "D constructor called\n";</pre>
  ~D() {
     cout << "D destructor called\n";</pre>
  void print() {
     // error: reference to num1 is ambiguous
     cout << num1 << endl;</pre>
int main() {
  Dd;
  d.C::num1 = 7;
  d.print();
  return 0;
```

```
class A {
public:
  int num1 = 1;
  A(){
     cout << "A constructor called\n";</pre>
  ~A() {
     cout << "A destructor called\n";</pre>
class B: public A {
public:
  B(){
     cout << "B constructor called\n";</pre>
  ~B() {
     cout << "B destructor called\n";</pre>
class C: public A {
public:
  C(){
     cout << "C constructor called\n";</pre>
  ~C() {
     cout << "C destructor called\n";</pre>
};
```

```
class D: public B, public C {
public:
  D(){
     cout << "D constructor called\n";</pre>
  ~D() {
     cout << "D destructor called\n";</pre>
  void print() {
     // error: 'A' is an ambiguous base of 'D'
     cout << A::num1 << endl;</pre>
int main() {
  Dd;
  d.C::num1 = 7;
  d.print();
  return 0;
```

```
class A {
public:
  int num1 = 1;
  A()
     cout << "A constructor called\n";
  ~A() {
     cout << "A destructor called\n";
class B: public A {
public:
  B(){
     cout << "B constructor called\n";</pre>
  ~B() {
     cout << "B destructor called\n";
class C: public A {
public:
  C(){
     cout << "C constructor called\n":
  ~C() {
     cout << "C destructor called\n";
};
```

```
class D: public B, public C {
public:
  D(){
     cout << "D constructor called\n";</pre>
  ~D() {
     cout << "D destructor called\n";
  void print() {
     cout << B::num1 << " " << C::num1 << endl;
};
int main() {
  Dd;
  d.C::num1 = 7;
  d.print();
  return 0;
```

A constructor called B constructor called A constructor called C constructor called D constructor called D destructor called C destructor called A destructor called B destructor called A destructor called

```
class A {
public:
  int num1 = 1;
  A()
     cout << "A constructor called\n";
  ~A() {
     cout << "A destructor called\n";
class B: virtual public A {
public:
  B(){
     cout << "B constructor called\n";</pre>
  ~B() {
     cout << "B destructor called\n";
class C: virtual public A {
public:
  C(){
     cout << "C constructor called\n";</pre>
  ~C() {
     cout << "C destructor called\n";
};
```

```
class D: public B, public C {
public:
  D(){
     cout << "D constructor called\n";</pre>
  ~D() {
     cout << "D destructor called\n";
  void print() {
     cout << B::num1 << " " << C::num1 << endl;
};
                                            A constructor called
int main() {
                                            B constructor called
  Dd;
                                            C constructor called
  d.C::num1 = 7;
                                            D constructor called
  d.print();
                                            77
  return 0;
                                            D destructor called
                                            C destructor called
                                            B destructor called
                                            A destructor called
```

```
class E {
class A {
                                                            public:
public:
                                                                                                           A constructor called
  int num1 = 1:
                                                              E(){
                                                                                                            E constructor called
                                                                 cout << "E constructor called\n";</pre>
  A()
                                                                                                            B constructor called
     cout << "A constructor called\n";
                                                                                                            C constructor called
                                                              ~E() {
                                                                 cout << "E destructor called\n";</pre>
  ~A() {
                                                                                                            D constructor called
     cout << "A destructor called\n":
                                                                                                            77
                                                                                                            D destructor called
                                                            class D: public E, public B, public C {
                                                                                                            C destructor called
class B: virtual public A {
                                                            public:
                                                                                                           B destructor called
public:
                                                              D(){
                                                                                                            E destructor called
  B(){
                                                                 cout << "D constructor called\n":
                                                                                                           A destructor called
     cout << "B constructor called\n";
                                                              ~D() {
                                                                 cout << "D destructor called\n";
  ~B() {
     cout << "B destructor called\n";
                                                              void print() {
                                                                 cout << B::num1 << " " << C::num1 << endl:
class C: virtual public A {
public:
  C(){
                                                           int main() {
     cout << "C constructor called\n":
                                                              D d:
                                                              d.C::num1 = 7;
  ~C() {
                                                              d.print();
     cout << "C destructor called\n":
                                                              return 0;
};
```

Virtual base class is constructed before any other class in multiple inheritance

```
class E: virtual public A {
class A {
public:
                                                           public:
                                                                                                          A constructor called
  int num1 = 1:
                                                             E(){
                                                                                                           E constructor called
                                                                cout << "E constructor called\n":
  A()
                                                                                                           A constructor called
     cout << "A constructor called\n";
                                                                                                          B constructor called
                                                             ~E() {
                                                                cout << "E destructor called\n";</pre>
  ~A() {
                                                                                                           C constructor called
     cout << "A destructor called\n":
                                                                                                           D constructor called
                                                           class D: public E, public B, public C {
                                                                                                           D destructor called
class B: public A {
                                                           public:
                                                                                                           C destructor called
public:
                                                             D(){
                                                                                                           B destructor called
  B(){
                                                                cout << "D constructor called\n";
                                                                                                          A destructor called
     cout << "B constructor called\n";
                                                                                                           E destructor called
                                                              ~D() {
                                                                cout << "D destructor called\n";
  ~B() {
                                                                                                          A destructor called
     cout << "B destructor called\n";
                                                             void print() {
                                                                cout << B::num1 << " " << C::num1;
                                                                cout << " " << E::num1 << endl;
class C: virtual public A {
public:
  C(){
     cout << "C constructor called\n":
                                                           int main() {
                                                             D d:
                                                             d.C::num1 = 7;
  ~C() {
     cout << "C destructor called\n":
                                                             d.print();
                                                             return 0;
};
```

Virtual base class is constructed before any other class in multiple inheritance

Interesting reads

- Accessing protected members in a derived class trhough object of base class is not permitted
 - https://stackoverflow.com/questions/3247671/accessing-protected-members-in-a-derived-class
- Inheritance of constructors and destructor
 - https://stackoverflow.com/questions/14184341/c-constructor-destructor-inheritance

