# OOP/Computer Programming

By: Dr. Danish Shehzad

# Today's Lecture

- Multilevel Inheritance
- Hierarchy of Inheritance
- Multiple Inheritance
- Diamond Problem
- Virtual Inheritance

# Multilevel Inheritance

- In C++ programming, not only you can derive a class from the base class but you can also derive a class from the derived class. This form of inheritance is known as multilevel inheritance.

```cpp
#include <iostream>
using namespace std;

class A
{
     public:
        void display()
        {
             cout<<"Base class content.";
        }
};

class B : public A
{

};

class C : public B
{

};

int main()
{
     C obj;
     obj.display();
     return 0;
}
```

**Output**

```
Base class content.
```

# Hierarchy of Inheritance

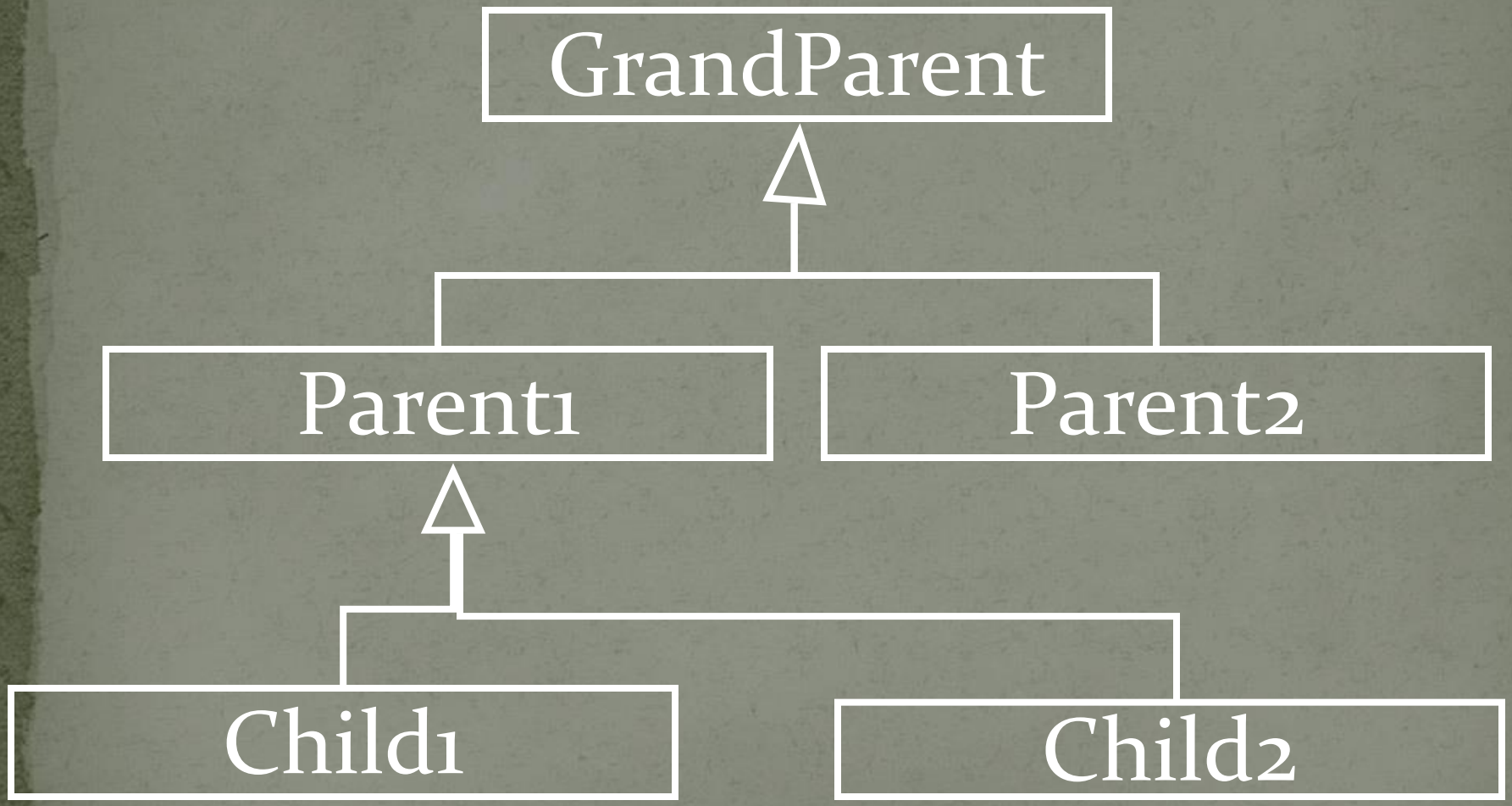
- We represent the classes involved in inheritance relation in tree like hierarchy
- More than one classes can be derived from a single class

# Direct Base Class

- A direct base class is explicitly listed in a derived class's header with a colon (:)

```
class Child1:public Parent1
...
```

# Example



# Indirect Base Class

- An indirect base class is not explicitly listed in a derived class's header with a colon (:)
- It is inherited from two or more levels up the hierarchy of inheritance

```
class GrandParent{};
class Parent1:public GrandParent {};
class Child1:public Parent1{};
```

# Base Initialization

- The child can only perform the initialization of direct base class through *base class initialization list*
- The child can not perform the initialization of an indirect base class through *base class initialization list*

# Example

```
class GrandParent{
   int gpData;
public:
   GrandParent() : gpData(0){…}
   GrandParent(int i) : gpData(i){…}
   void Print() const;
};
```

# Example

```
class Parent1: public GrandParent{
   int pData;
public:
   Parent1() : GrandParent(),
          pData(0) {...}
};
```

# Example

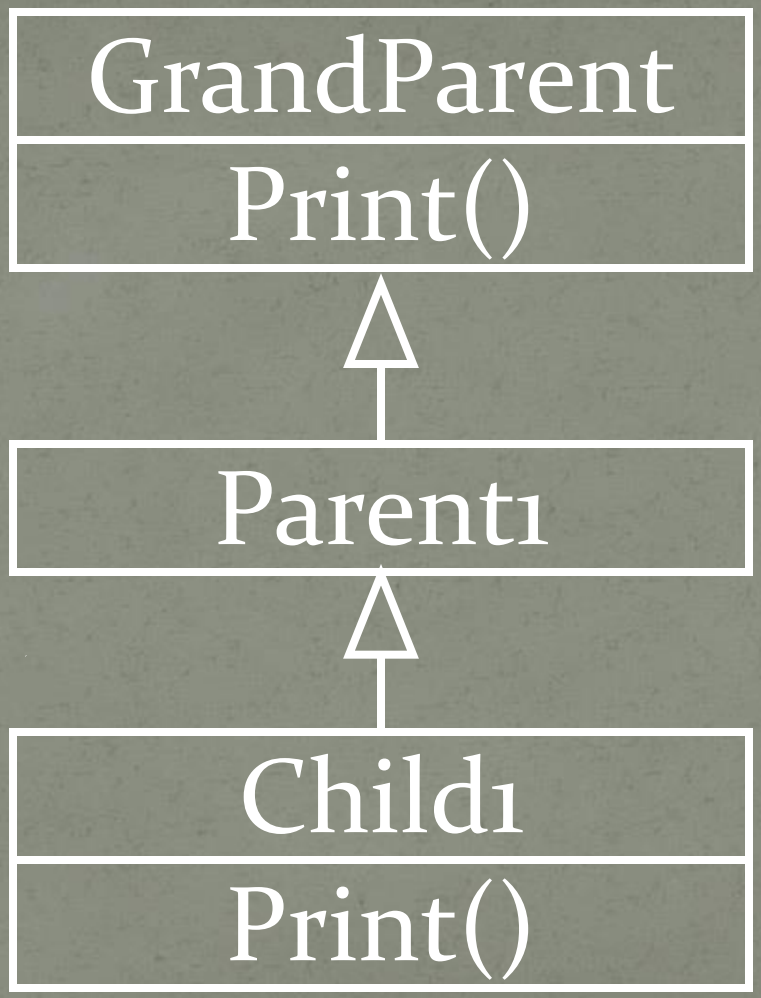
```
class Child1 : public Parent1 {
public:
   Child1() : Parent1()        {...}
   Child1(int i) : GrandParent (i) //Error
   {...}
   void Print() const;
};
```

The child can not perform the initialization of an indirect base class through *base class initialization list*

# Overriding

- Child class can override the function of GrandParent class

# Example



# Example

```
void GrandParent::Print() {
    cout     << “GrandParent::Print”
             << endl;
}

void Child1::Print() {
    cout << “Child1::Print” << endl;
}
```

# Example

```
int main(){
   Child1 obj;
   obj.Print();
   obj.Parent1::Print();
   obj.GrandParent::Print();
   return 0;
}
```

# Output

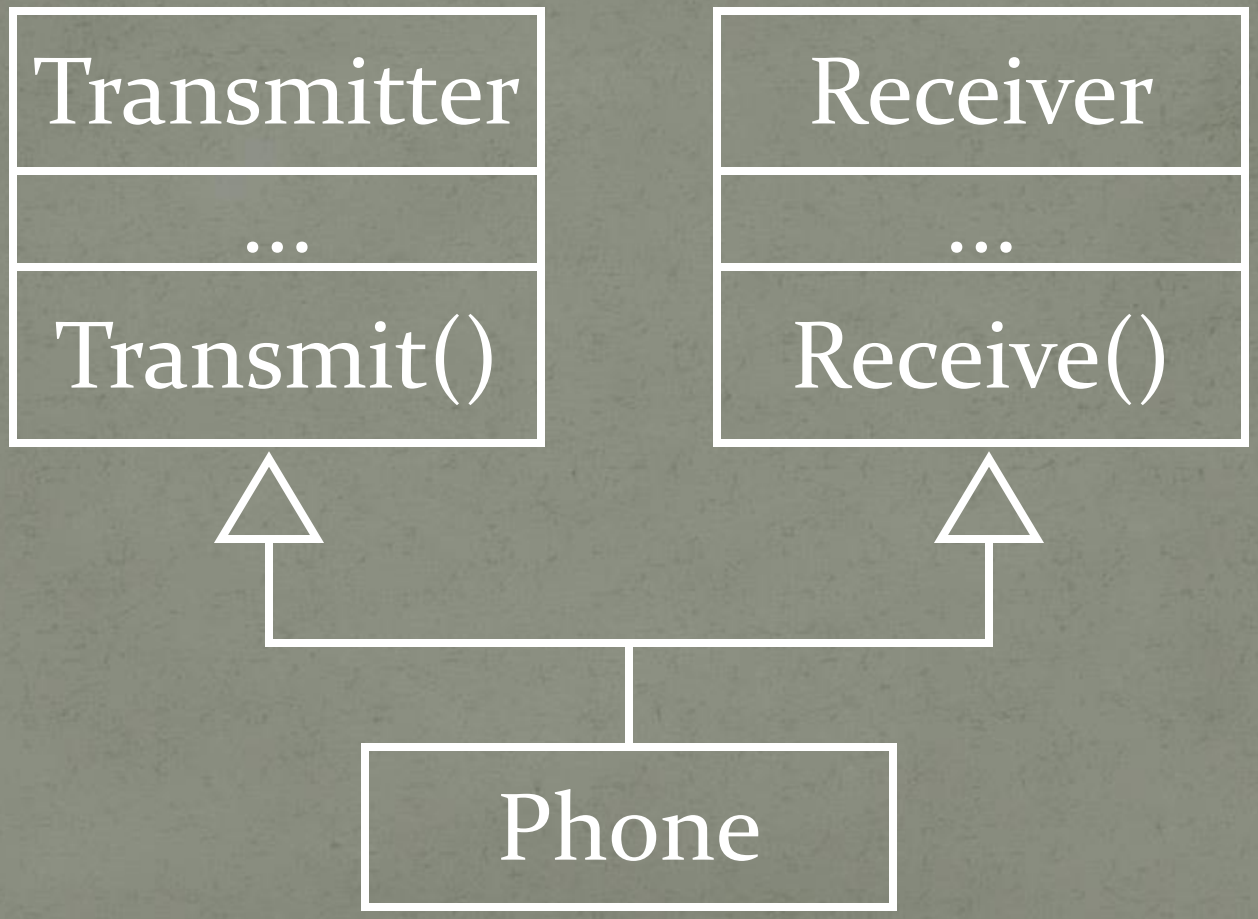
- Output is as follows

Child1::Print
GrandParent::Print
GrandParent::Print

# Multiple Inheritance

- A class can inherit from more then one class

# Multiple Inheritance



# Example

```
class Phone: public Transmitter,
             public Receiver
{
...
};
```

# Multiple Inheritance

- Derived class can inherit from public base class as well as private and protected base classes

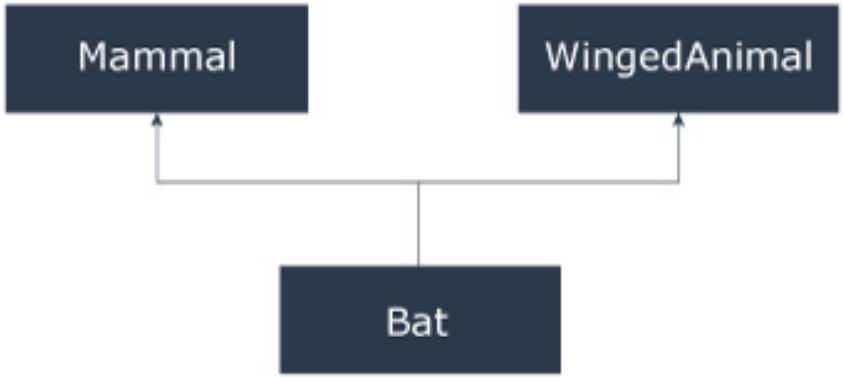
```
class Mermaid:
        private Woman, private Fish
```

# Multiple Inheritance

- The derived class inherits data members and functions form all the base classes
- Object of derived class can perform all the tasks that an object of base class can perform

# Example

```
int main(){
   Phone obj;
   obj.Transmit();
   obj.Receive();
   return 0;
}
```




**Output**

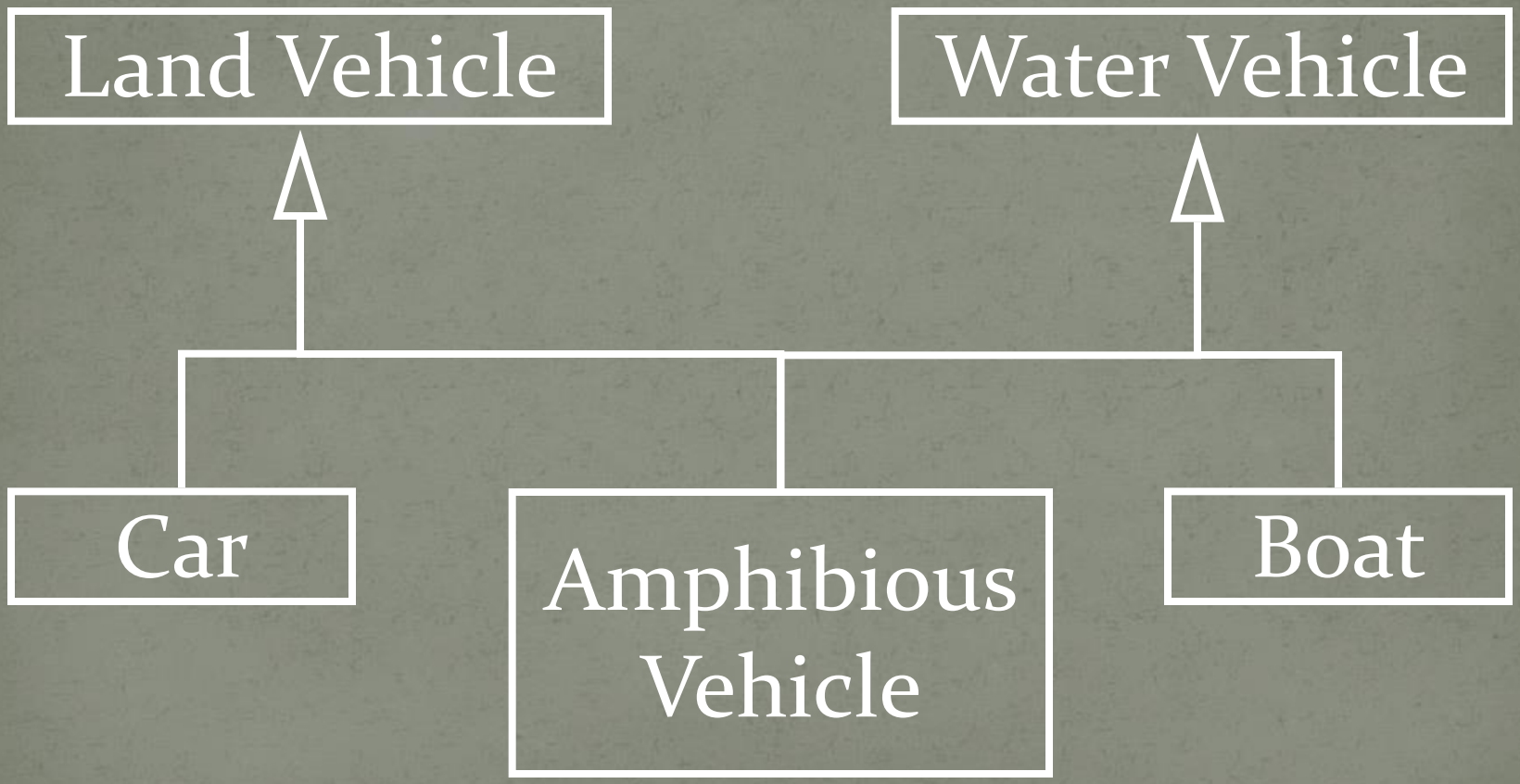
```
Mammals can give direct birth.
Winged animal can flap.
```

```cpp
#include <iostream>
using namespace std;

class Mammal {
  public:
     Mammal()
     {
        cout << "Mammals can give direct birth." << endl;
     }
};

class WingedAnimal {
  public:
     WingedAnimal()
     {
        cout << "Winged animal can flap." << endl;
     }
};

class Bat: public Mammal, public WingedAnimal {

};

int main()
{
     Bat b1;
     return 0;
}
```

# Multiple Inheritance: Ambiguity

- If more than one base class have a function with same signature then the child will have two copies of that function
- Calling such function will result in ambiguity

# Multiple Inheritance



# Example

```
class LandVehicle{
public:
   int GetMaxLoad();
};
class WaterVehicle{
public:
   int GetMaxLoad();
};
```

# Example

```
class AmphibiousVehicle:
                public LandVehicle,
                public WaterVehicle{
};
int main(){
  AmphibiousVehicle obj;
  obj.GetMaxLoad();              // Error
  return 0;
}
```

# Multiple Inheritance

- Programmer must explicitly specify the class name when calling ambiguous function

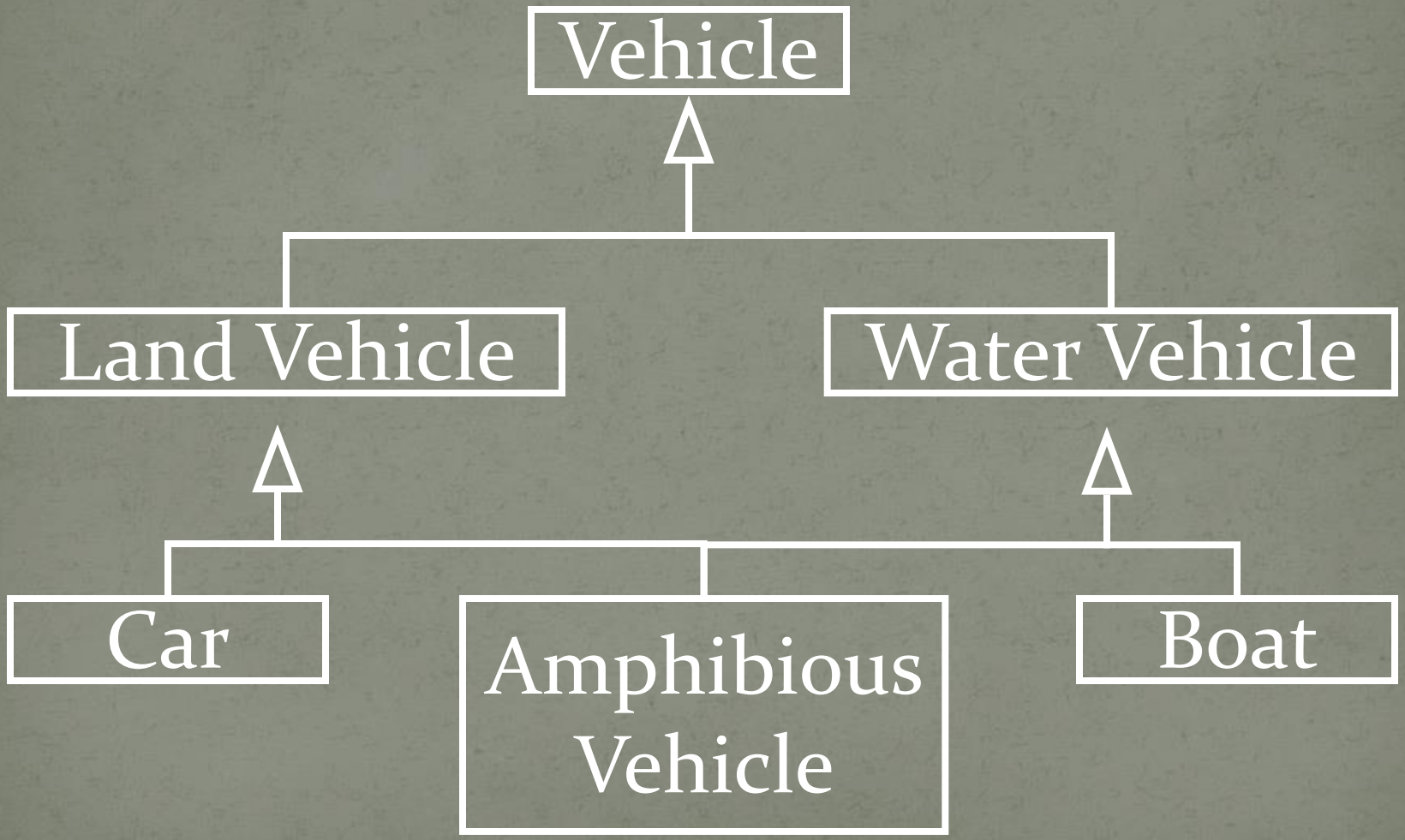
# Example

```
int main(){
   AmphibiousVehicle obj;
   obj.LandVehicle::GetMaxLoad();
   obj.WaterVehicle::GetMaxLoad();
   return 0;
}
```

# Hybrid Inheritance

- The ambiguous call problem can arise when dealing with multiple level of multiple inheritance

# Hybrid Inheritance



# Example

```
class Vehicle{
public:
   int GetMaxLoad();
};
class LandVehicle : public Vehicle{
};
class WaterVehicle : public Vehicle{
};
```

# Example

```
class AmphibiousVehicle:
                public LandVehicle,
                public WaterVehicle{
};
int main(){
  AmphibiousVehicle obj;
  obj.GetMaxLoad();              // Error
  return 0;
}
```

# Example

```
int main()
{
  AmphibiousVehicle obj;
  obj.Vehicle::GetMaxLoad();  //Error
  return 0;
}
```

- Vehicle is accessible through two paths

# Multiple Inheritance

- Data member must be used with care when dealing with more then one level of inheritance

# Example

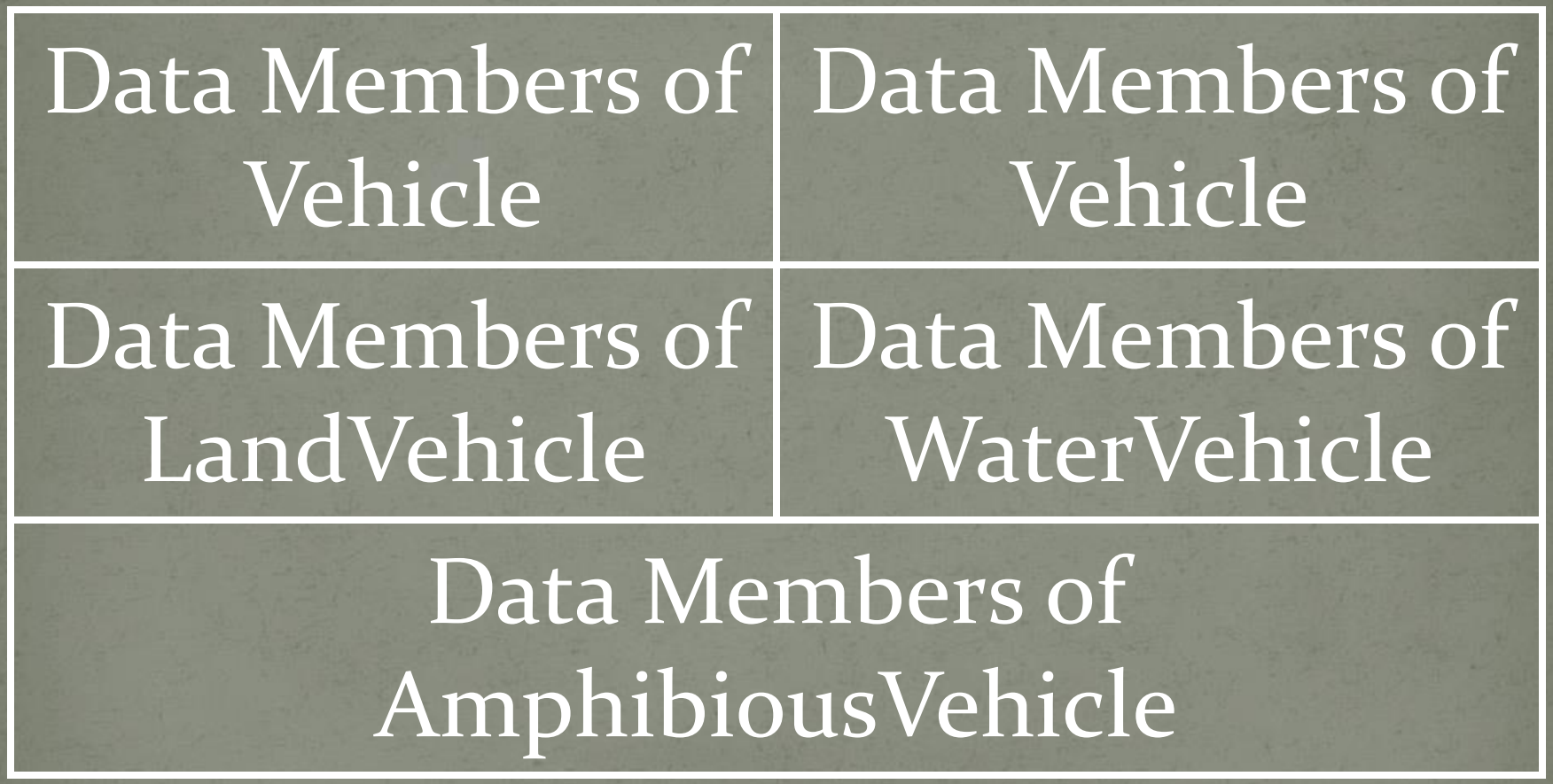
```
class Vehicle{
protected:
   int weight;
};
class LandVehicle : public Vehicle{
};
class WaterVehicle : public Vehicle{
};
```

# Example

```
class AmphibiousVehicle:
                public LandVehicle,
                public WaterVehicle{
public:
   AmphibiousVehicle(){
        LandVehicle::weight = 10;
        WaterVehicle::weight = 20;
   }
};
```

- There are multiple copies of data member weight

# Memory View


Data Members of Vehicle
Data Members of Vehicle
Data Members of LandVehicle
Data Members of WaterVehicle
Data Members of AmphibiousVehicle

# Virtual Inheritance

- In virtual inheritance there is exactly one copy of the anonymous base class object

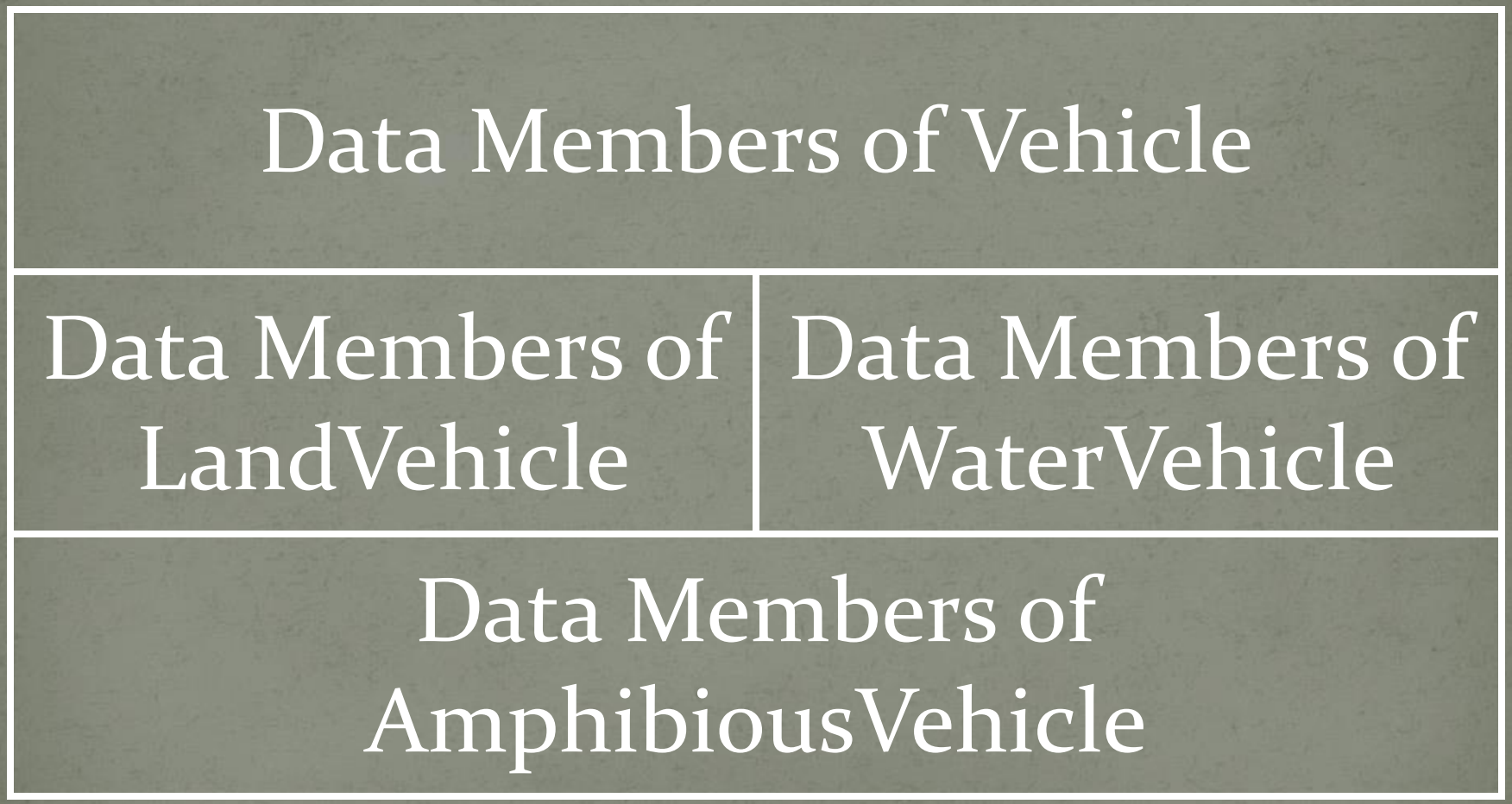
# Example

```
class Vehicle{
protected:
   int weight;
};
class LandVehicle : public virtual Vehicle{};

class WaterVehicle :public virtual Vehicle{};
```

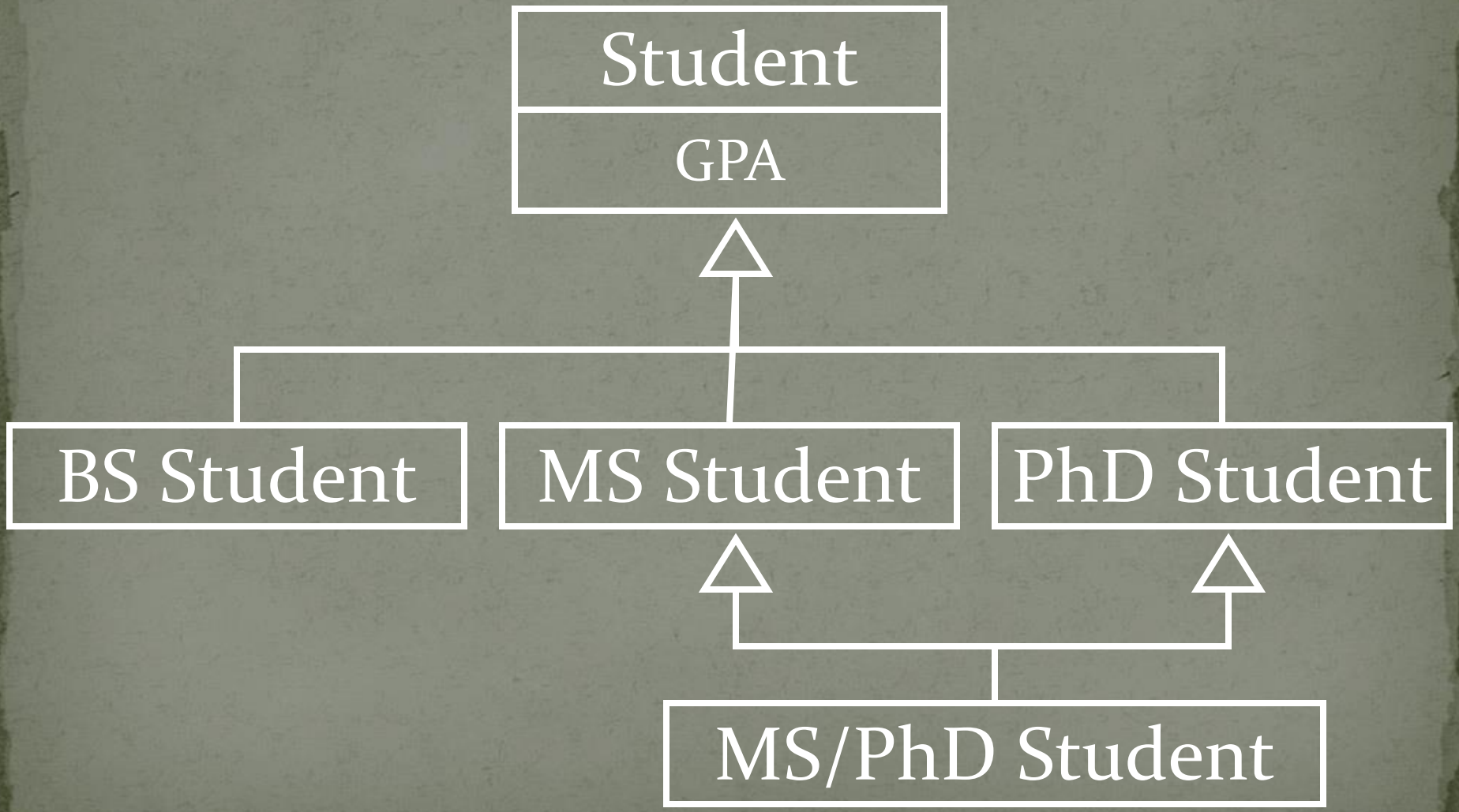
# Example

```
class AmphibiousVehicle:
                public LandVehicle,
                public WaterVehicle{
public:
   AmphibiousVehicle(){
        weight = 10;
   }
};
```

# Memory View


Data Members of Vehicle
Data Members of LandVehicle
Data Members of WaterVehicle
Data Members of AmphibiousVehicle

# Example


Student
GPA
BS Student
MS Student
PhD Student
MS/PhD Student

# Example Code

- https://www.tutorialspoint.com/compile_cpp_online.php