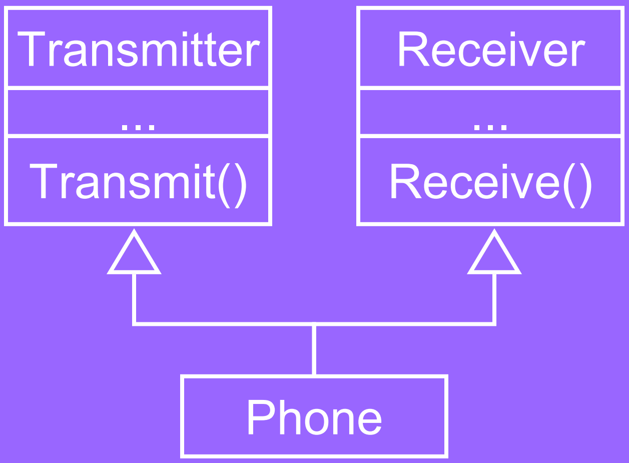
Object-Oriented Programming (OOP) Lecture No. 31

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;
}
```

Multiple Inheritance

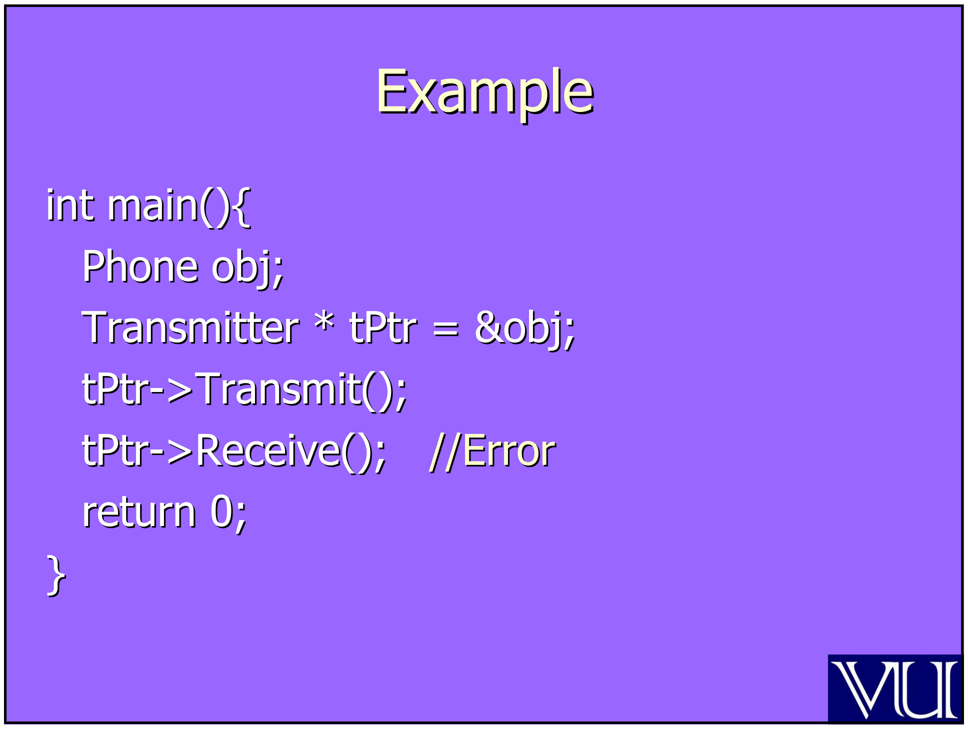
- When using public multiple inheritance, the object of derived class can replace the objects of all the base classes

Example

```
int main(){
   Phone obj;
   Transmitter * tPtr = &obj;
   Receiver * rPtr = &obj;
   return 0;
}
```

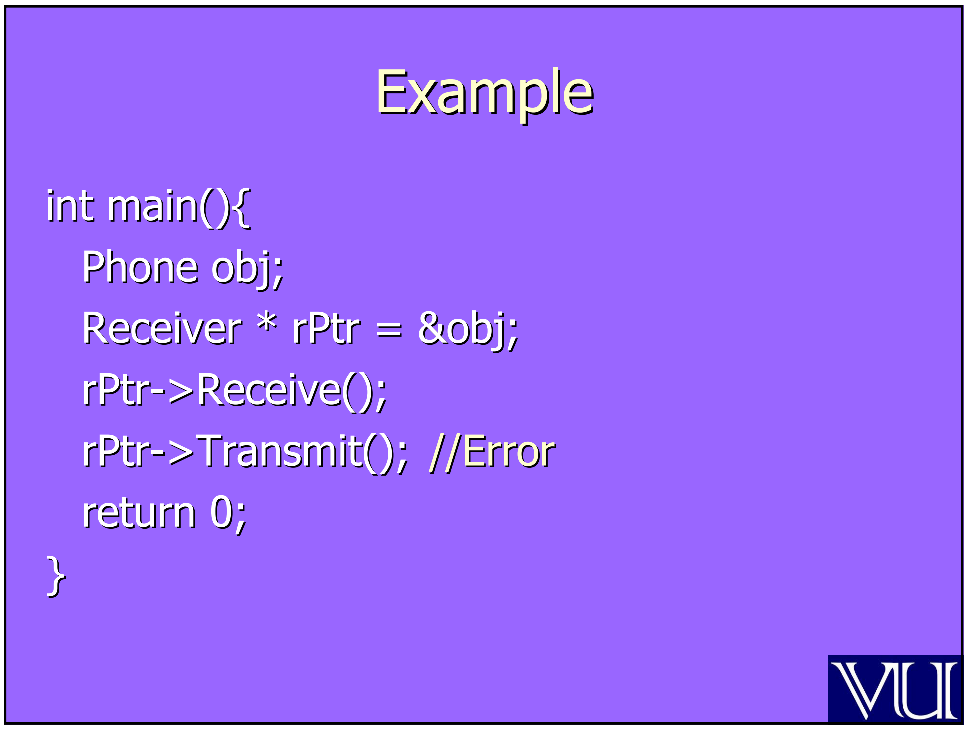
Multiple Inheritance

- The pointer of one base class cannot be used to call the function of another base class
- The functions are called based on static type



Example

```
int main(){
   Phone obj;
   Transmitter * tPtr = &obj;
   tPtr->Transmit();
   tPtr->Receive();   //Error
   return 0;
}
```



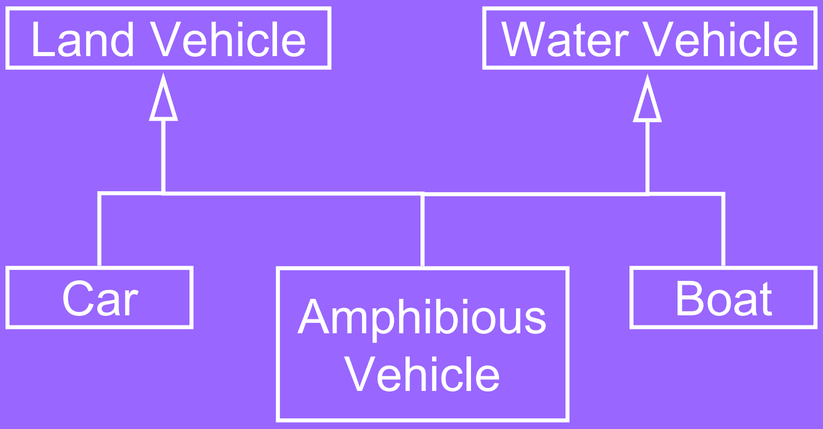
Example

```
int main(){
   Phone obj;
   Receiver * rPtr = &obj;
   rPtr->Receive();
   rPtr->Transmit(); //Error
   return 0;
}
```

Multiple Inheritance

- 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

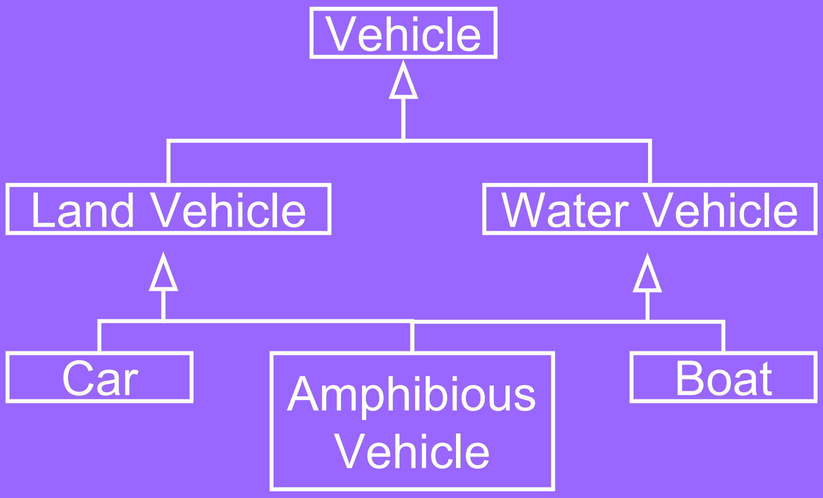
Example

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

Multiple Inheritance

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

Multiple Inheritance



Example

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

Example

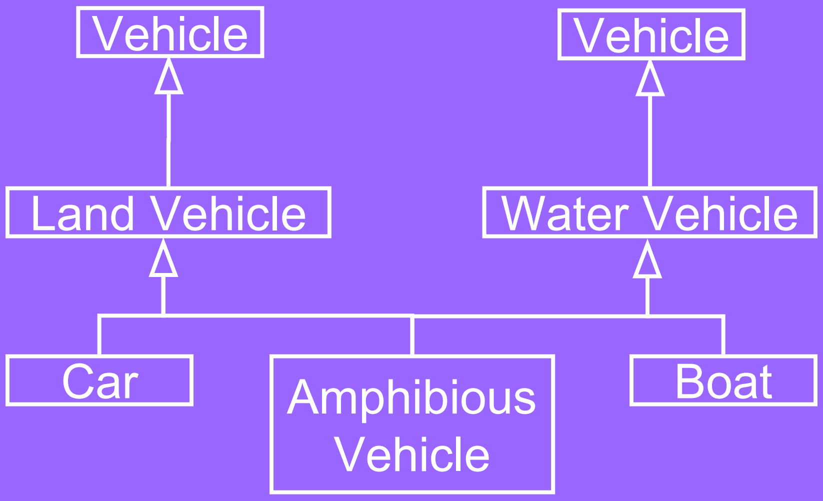
```
class AmphibiousVehicle:
                public LandVehicle,
                public WaterVehicle{
};
int main(){
   AmphibiousVehicle obj;
   obj.GetMaxLoad();           // Error
   return 0;
}
```

Example

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

- Vehicle is accessible through two paths

Multiple Inheritance

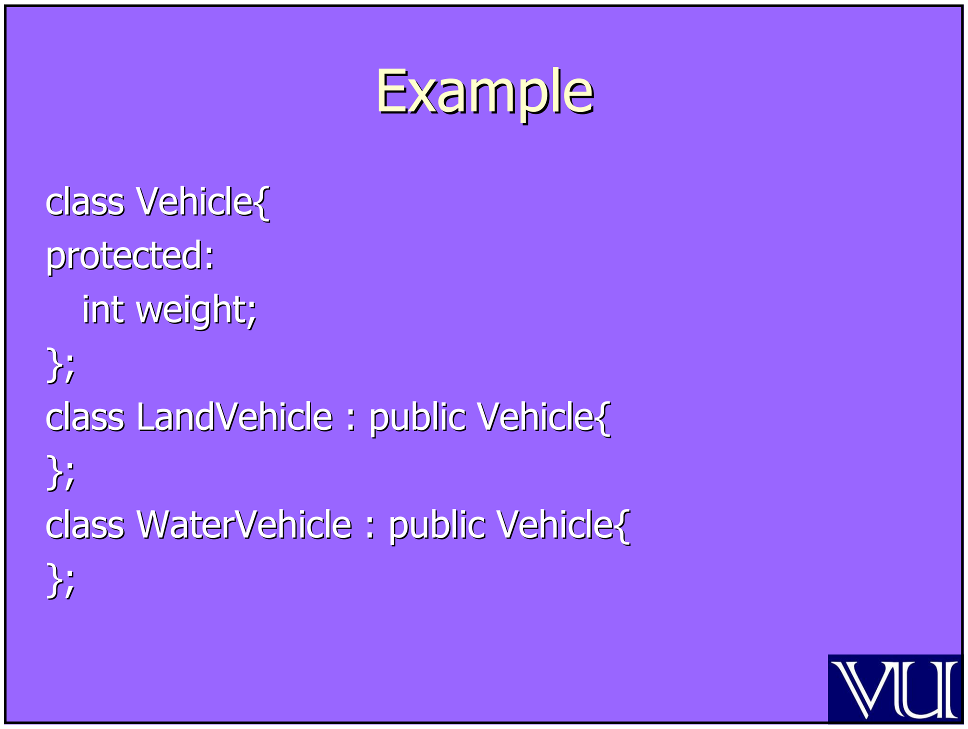


Example

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

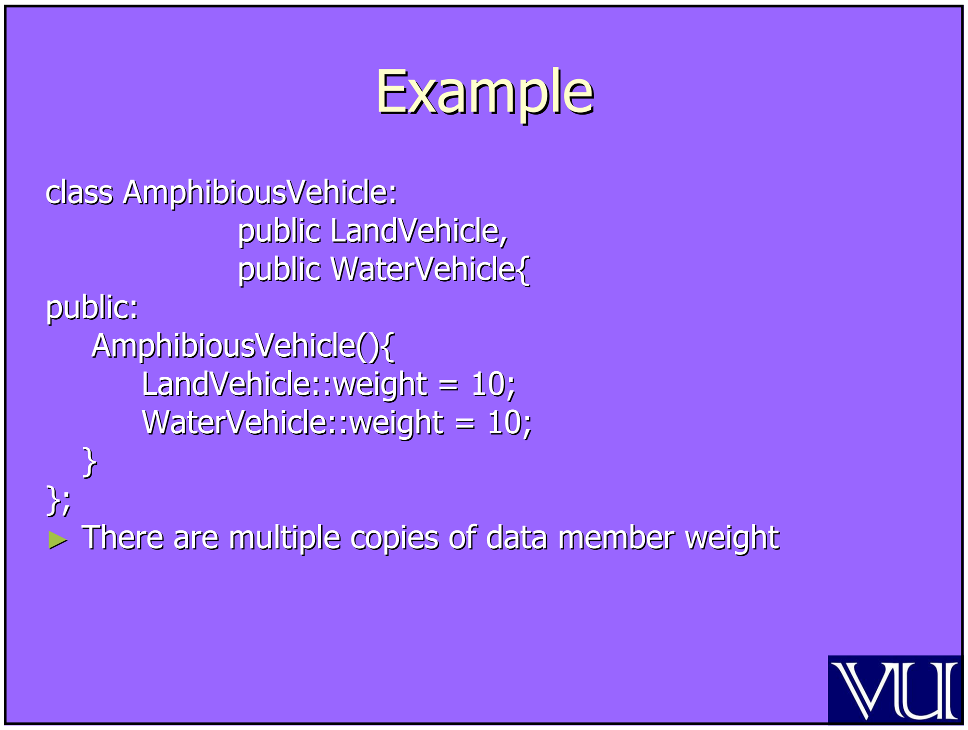
Multiple Inheritance

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



Example

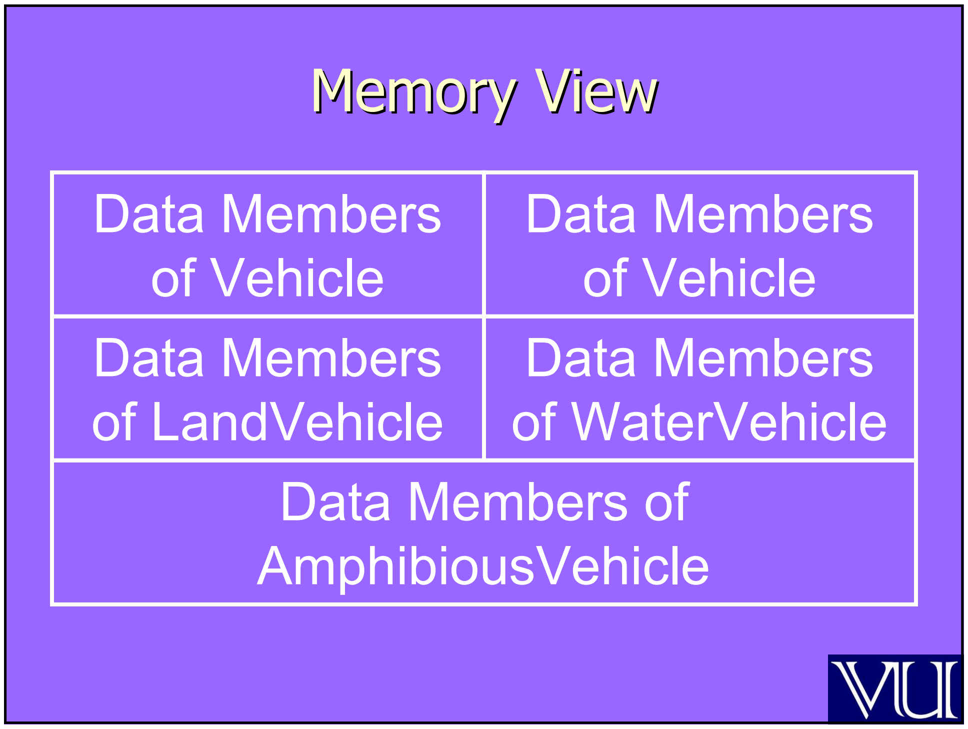
```
class Vehicle{
protected:
   int weight;
};
class LandVehicle : public Vehicle{
};
class WaterVehicle : public Vehicle{
};
```



Example

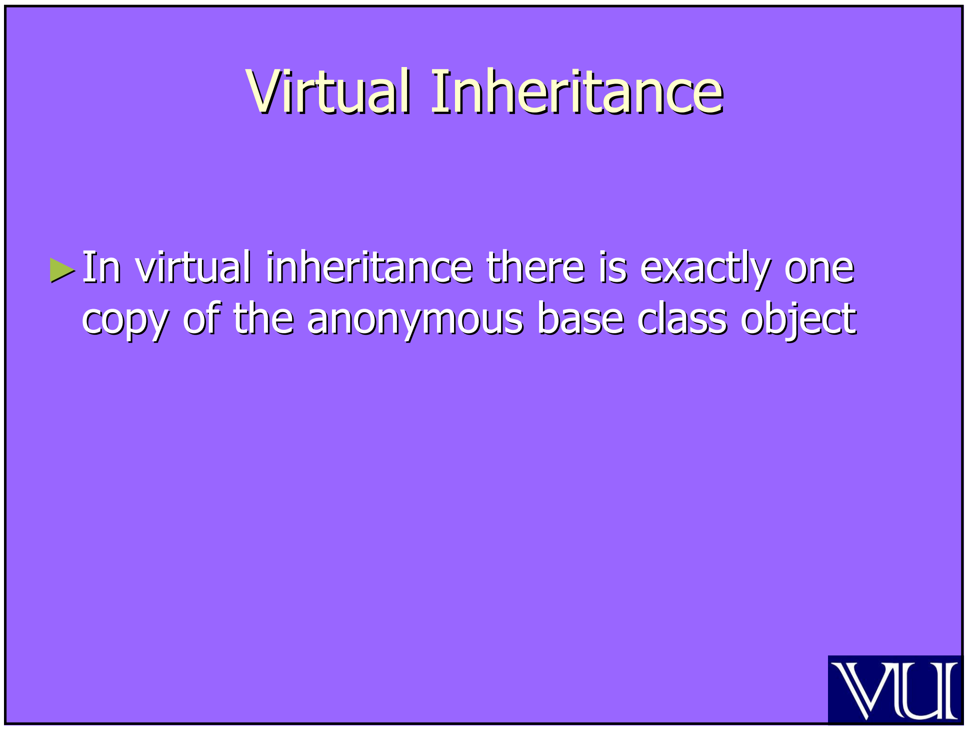
```
class AmphibiousVehicle:
                public LandVehicle,
                public WaterVehicle{
public:
    AmphibiousVehicle(){
        LandVehicle::weight = 10;
        WaterVehicle::weight = 10;
   }
};
```

- 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

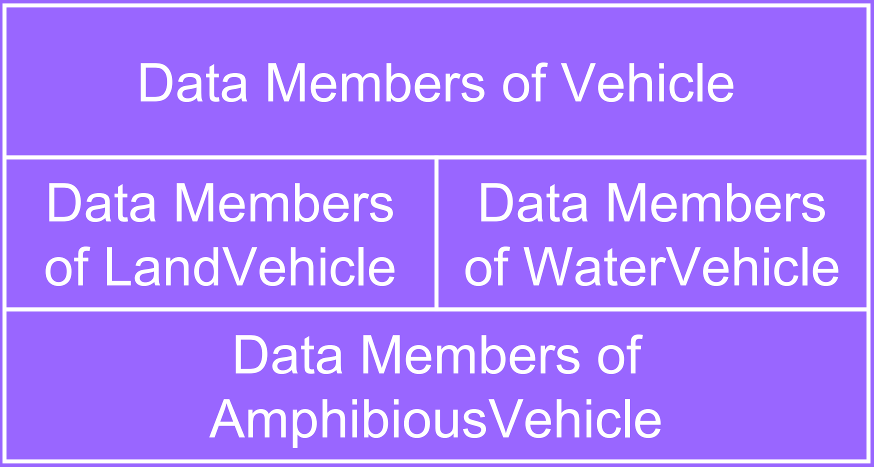
Example

```
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;
    }
};
```

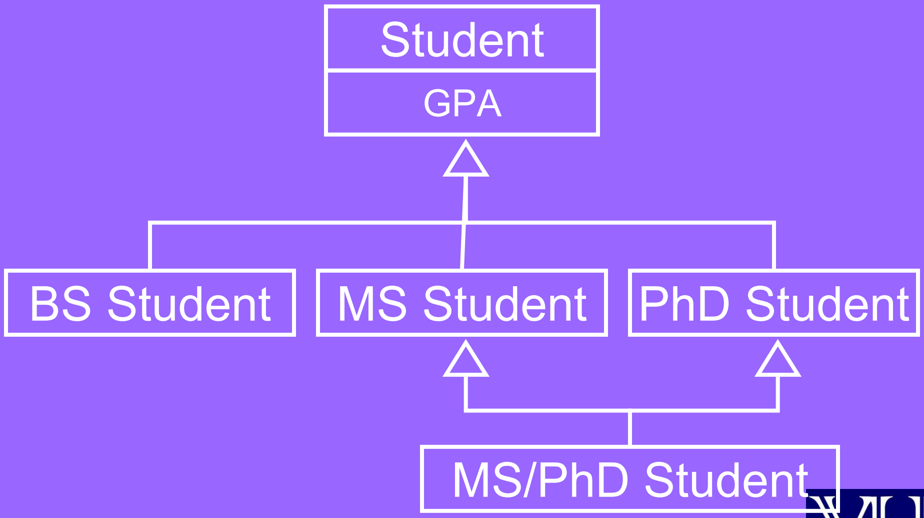
Memory View


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

Virtual Inheritance

- Virtual inheritance must be used when necessary
- There are situation when programmer would want to use two distinct data members inherited from base class rather then one

Example


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