Object-oriented Programming

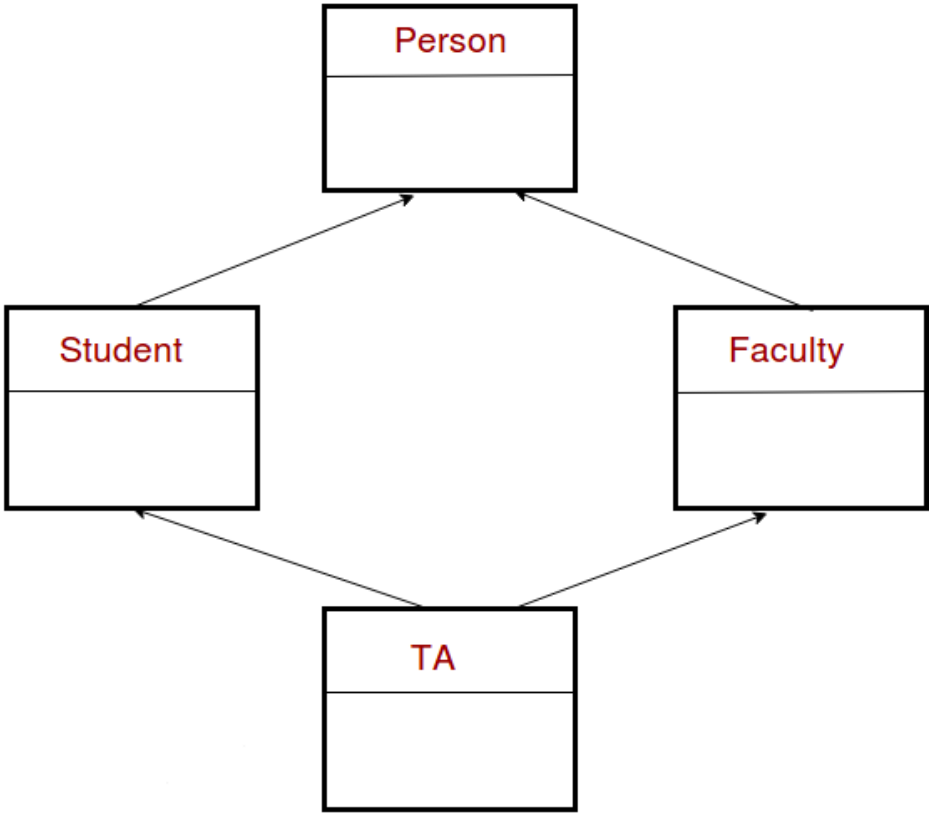
Diamond Problem |

Virtual Inheritance

Diamond Problem

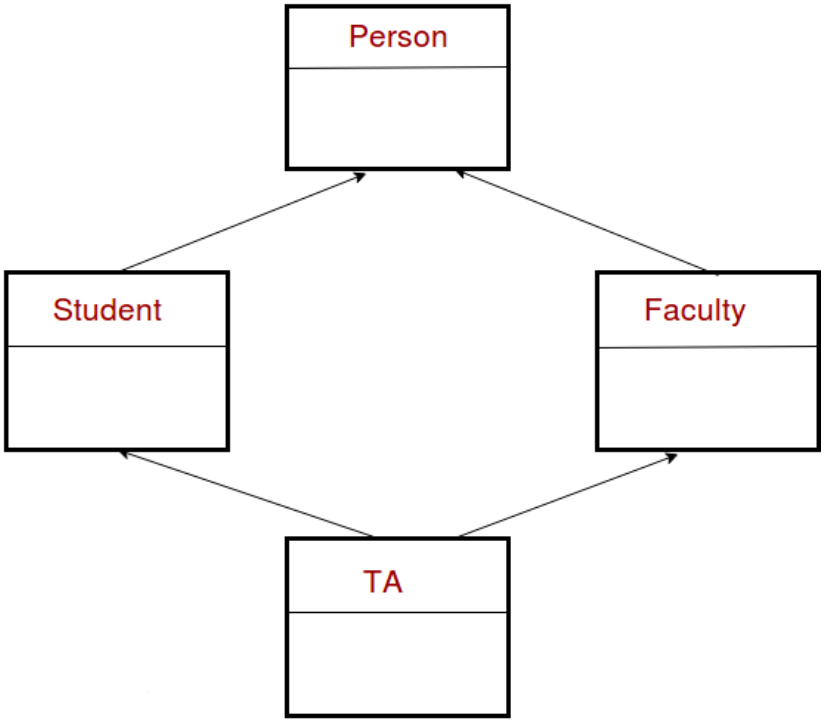
Diamond Problem

- The diamond problem can occur when two classes have a common parent as well as a common child (in case of multiple inheritance).



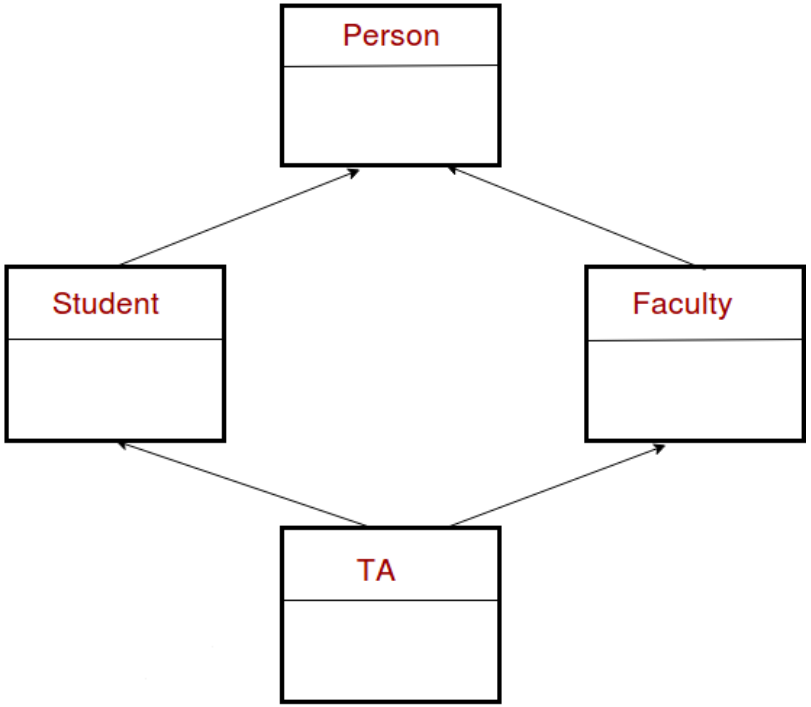
Diamond Problem

- In this example, if we make an object of TA, it implicitly “constructs” the class Person twice (through Faculty and also through Student)



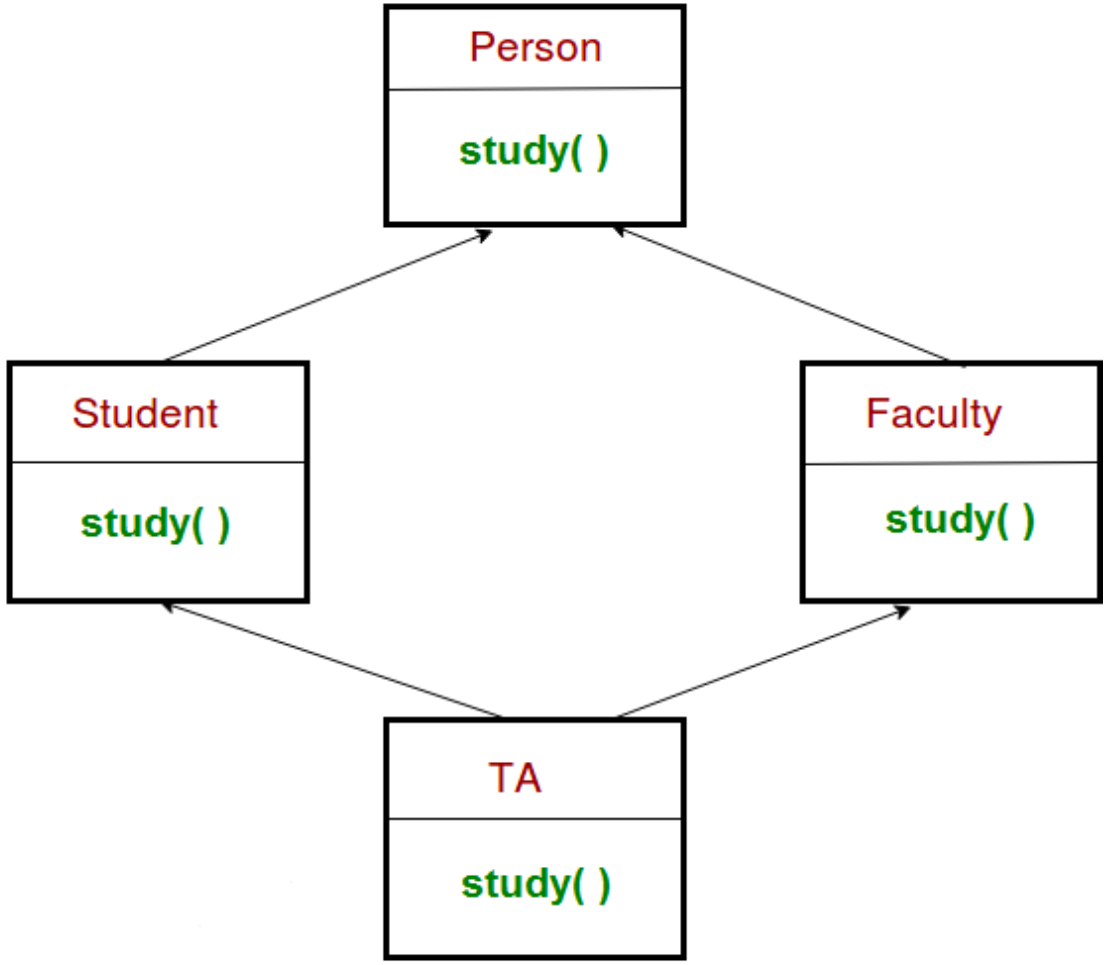
Diamond Problem

- Let’s define some functions in the given four classes to see how diamond problem can arise with function overriding



Case 1:

```
int main()
{
      TA  ob;
      ob.study( );
}
```

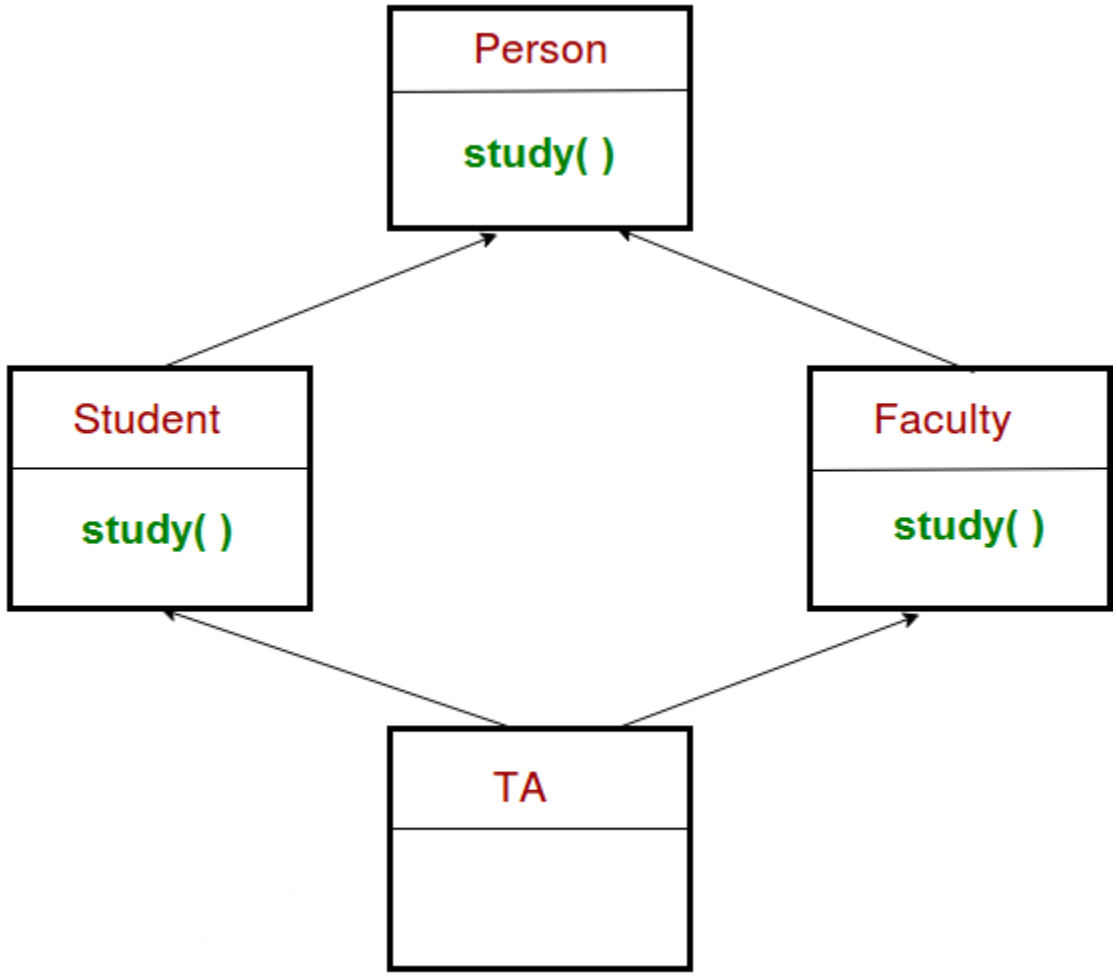


Output:

Will compile successfully: study() function of TA class will be called

Case 2:

```
int main()
{
      TA  ob;
      ob.study( );
}
```

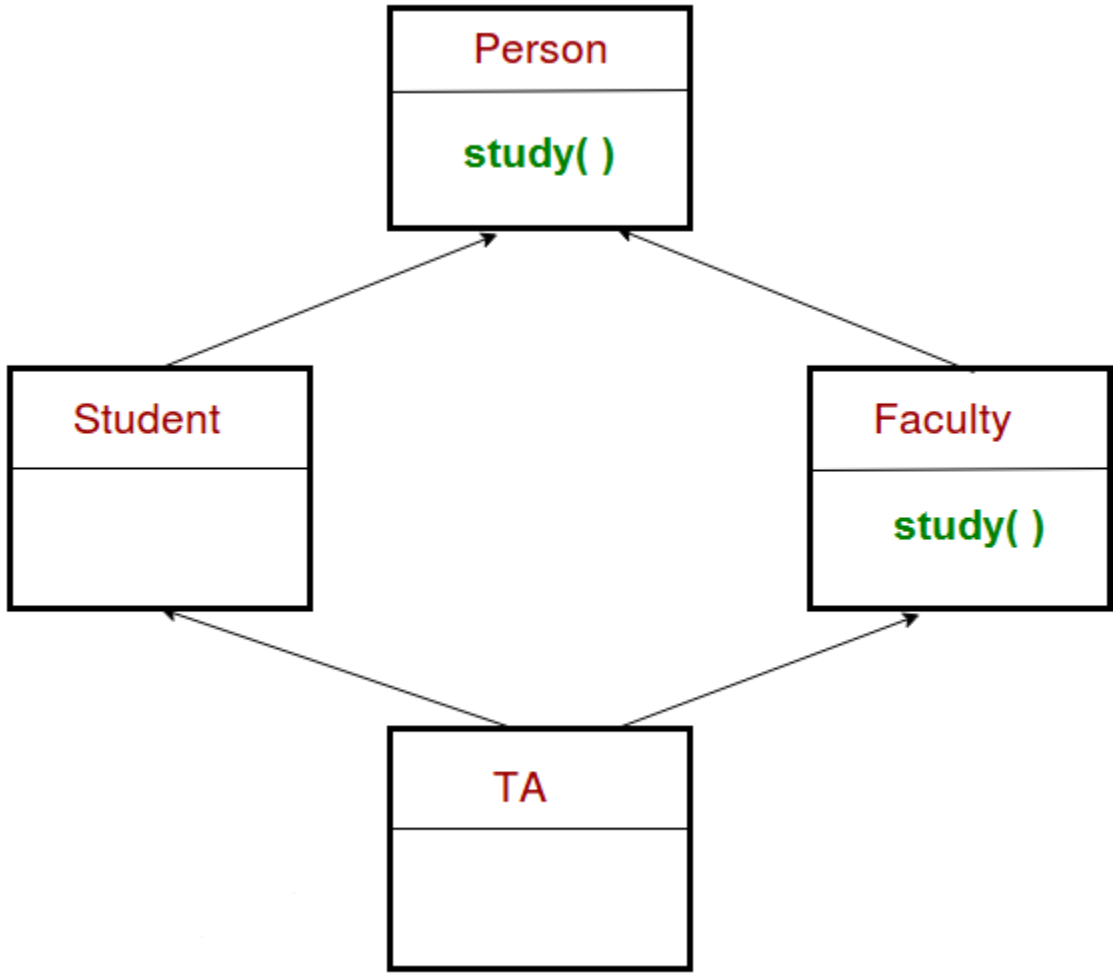


Output: **Error: ambiguous function call**

Reason: **TA** inheriting *study*() from both of its parents

Case 3:

```
int main()
{
      TA  ob;
      ob.study( );
}
```

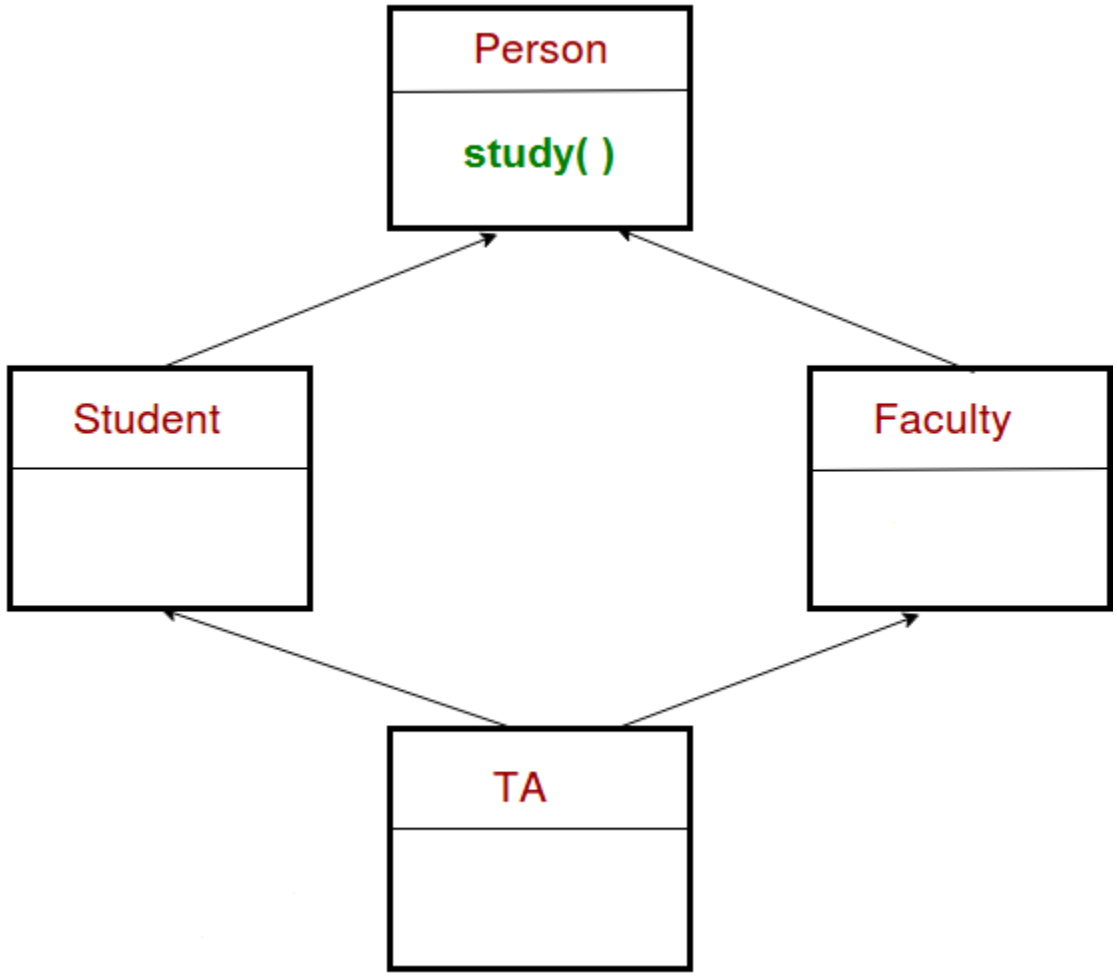


Output: **Error: ambiguous function call**

Reason: **TA** inheriting *study*() from both of its parents, since **Student** still contains *study*( ) inherited from **Person**

Case 4:

```
int main()
{
      TA  ob;
      ob.study( );
}
```

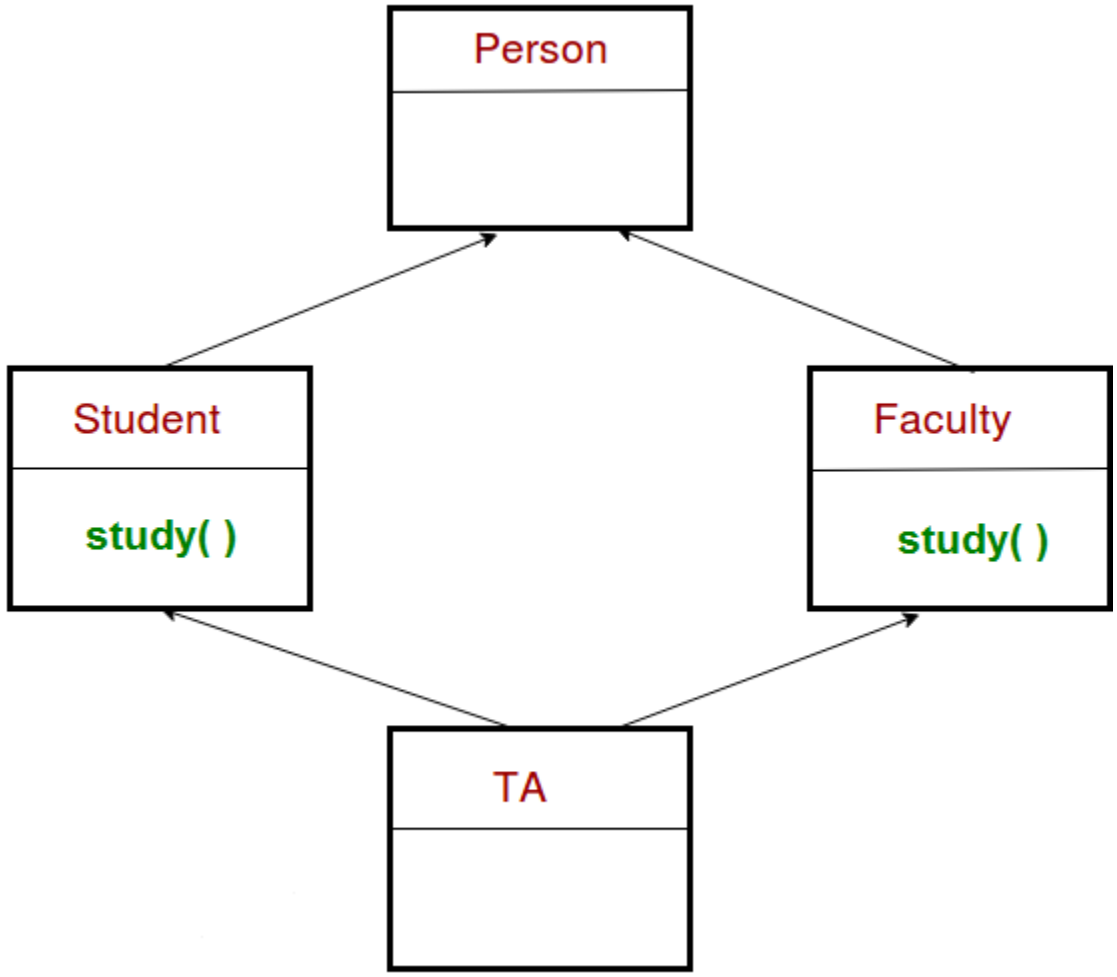


Output: **Error: ambiguous function call**

Reason: **TA** inheriting *study*() from both of its parents, since both Student and Faculty still contain *study*( ) inherited from **Person**

Case 5:

```
int main()
{
      TA  ob;
      ob.study( );
}
```



Output: **Error: ambiguous function call**

Reason: **TA** inheriting *study*() from both of its parents

Solution

- Fortunately, C++ allows us to solve Diamond Problem by using virtual inheritance

- We use the keyword **virtual** when we inherit from the base class in both derived classes

Virtual Inheritance

```
class Person
{       }

class Faculty: virtual public Person {         };

class Student: virtual public Person {         };

class TA: public Faculty, public Student {         };
```

Virtual Inheritance

- Let's see if virtual inheritance can help us in the cases presented previously:

- **Case 1:** No ambiguity with or without virtual
- **Case 2:** Virtual Inheritance won't help
- **Case 3:** Virtual Inheritance will remove ambiguity
- **Case 4:** Virtual Inheritance will remove ambiguity
- **Case 5:** Virtual Inheritance won't help

Discussion

- Even with virtual inheritance, we can never always be sure of “easily” resolving the diamond problem
- For example, cases where we are not allowed to remove the overridden functions from parent classes
- In such cases we have to change the layout/inheritance hierarchy of our classes instead

Discussion

- If multiple inheritance can lead to such issues then why use it at all???

- JAVA/C# use interfaces instead (a better alternative to multiple inheritance).