OBJECT ORIENTED PROGRAMMIG

Lecture Six

Object Oriented Programming

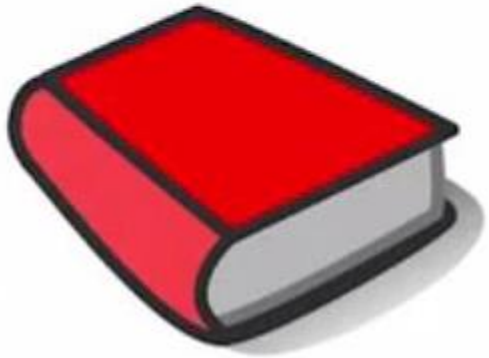
Object

Object

Object

Everything in Java is an Object

I'm not an object!



OOP Languages

Objective-C

Visual Basic

R

Python

C++

Ruby

Java

JavaScript

C#

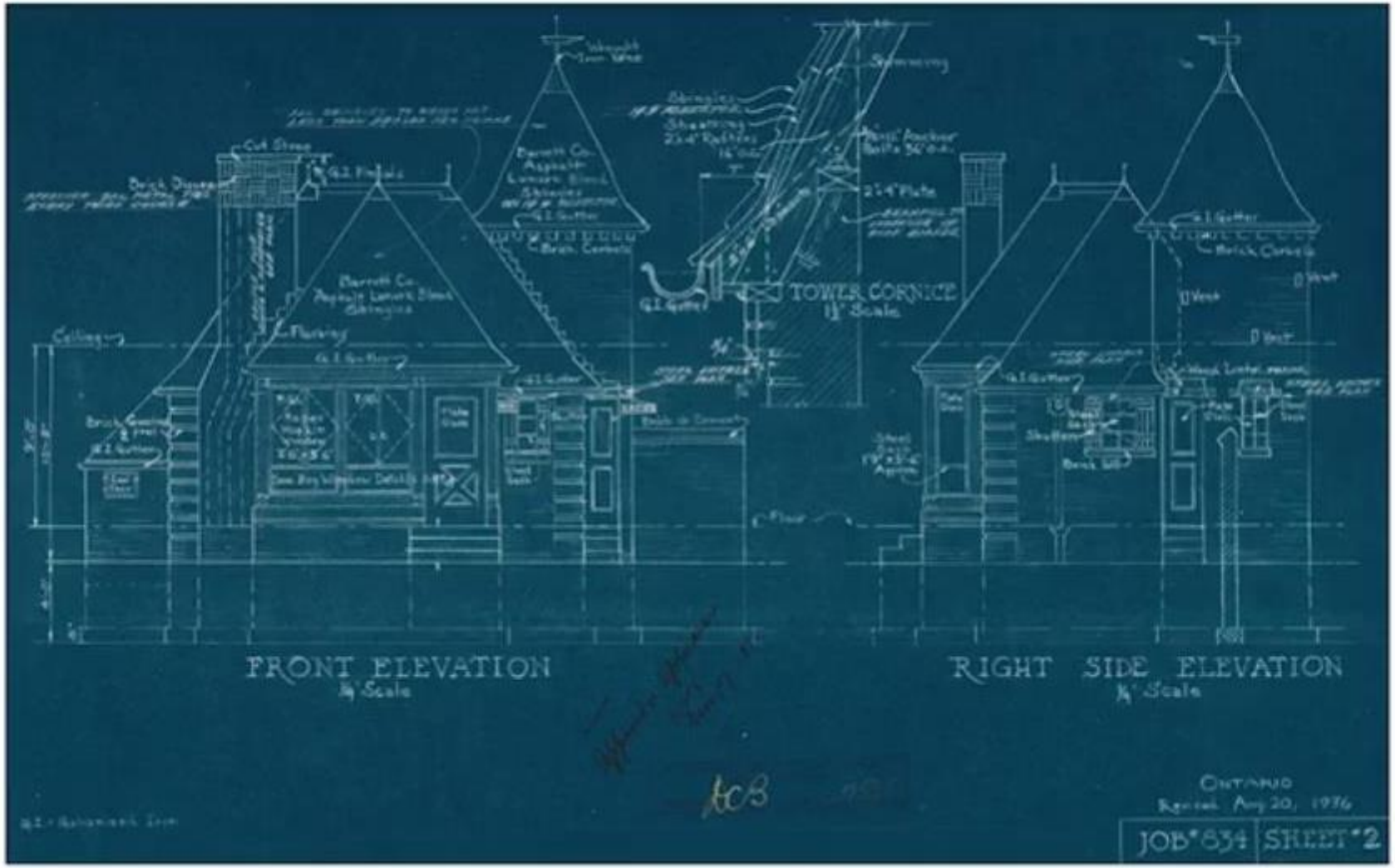
PHP

Delphi

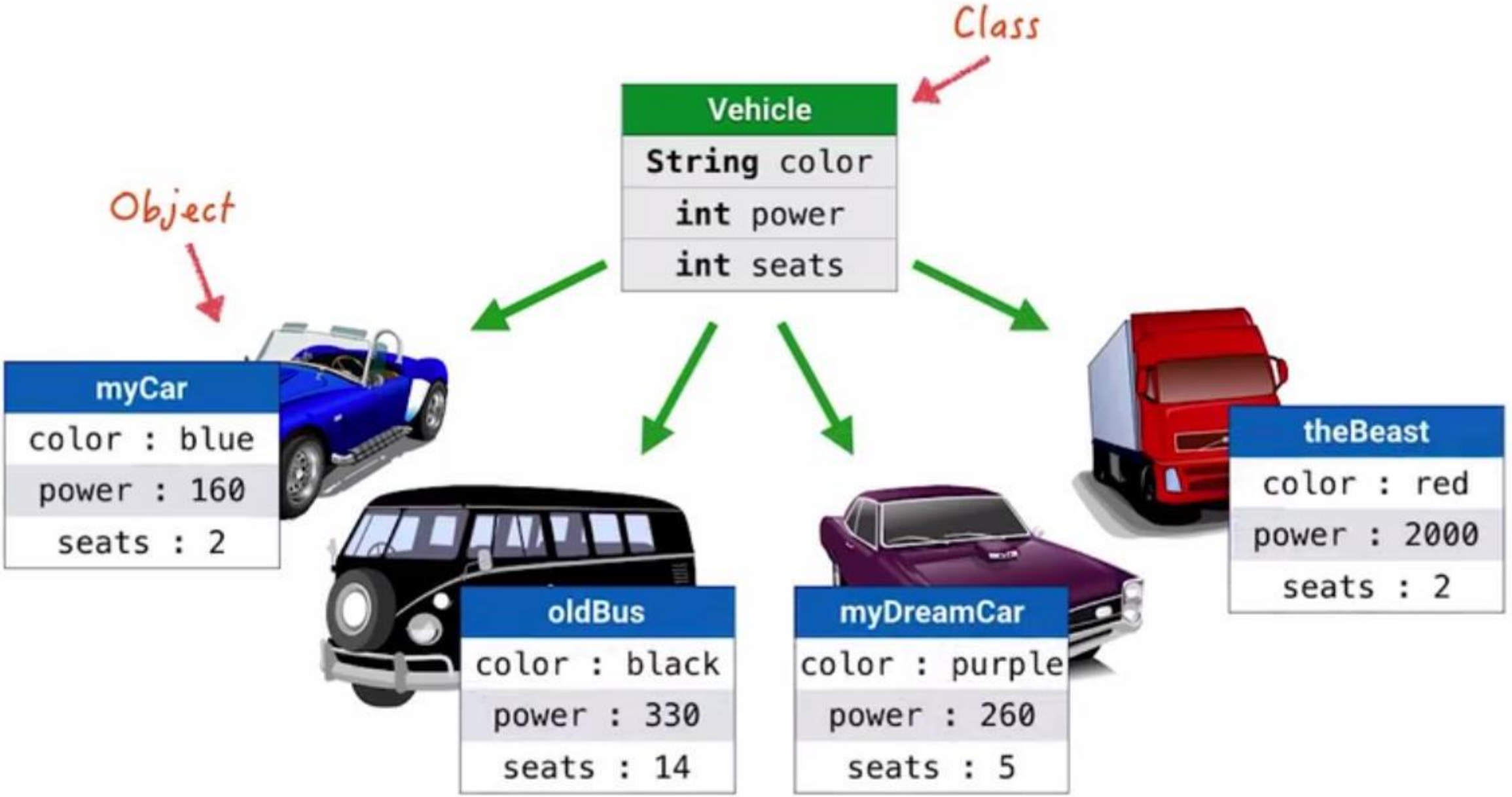
Perl

Scala

Classes


TOWER CORNICE
1½" Scale
FRONT ELEVATION
¼" Scale
RIGHT SIDE ELEVATION
¼" Scale
JOB 834
SHEET 2

Objects


Class
Vehicle
String color
int power
int seats
Object
myCar
color : blue
power : 160
seats : 2
oldBus
color : black
power : 330
seats : 14
myDreamCar
color : purple
power : 260
seats : 5
theBeast
color : red
power : 2000
seats : 2

```
class Pokemon {

    String name;
    String type;
    int health;

    boolean dodge(){
        return Math.random()>0.5;
    }

    void attack(Pokemon enemy){
        if(!enemy.dodge()){
            enemy.health--;
        }
    }

}
```

Fields

Methods

Pokemon Objects

Pokemon
Name: Pikachu
Type: Electric
Health: 70
attack()
dodge()
evolve()

Fields

Methods

Pokemon Objects

Pokemon
Name: Bulbasaur
Type: Grass
Health: 100

Pokemon
Name: Squirtle
Type: Water
Health: 0

LureItem
Name: Incense
Duration: 30

Pokemon
Name: Charizard
Type: Fire
Health: 100

Pokemon
Name: Pikachu
Type: Electric
Health: 70

LureItem
Name: Lure Module
Duration: 60

PokeBall
Type: Normal
Used: False

Item
Type: Incubator
Size: 5
Used: False

Variable types

int

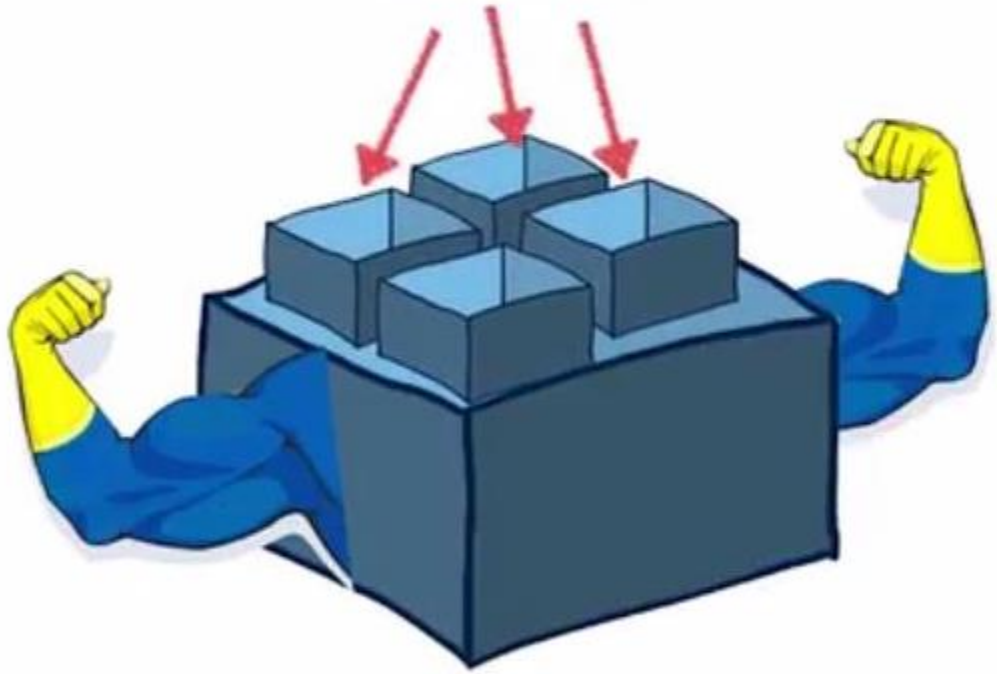
← Primitive variables

Pokemon

← Object variables

Variable types

Primitive variables



Object

Classes and Objects

- **The basic idea behind an object-oriented programming (OOP) is to combine both data and associated procedures (known as methods) into a single unit which operate on the data. Such a unit is called an object.**
- **For instance, an object could represent a person with a name property, age, address, etc., with behaviors like walking, talking, breathing, and running. Or an email with properties like recipient list, subject, body, etc., and behaviors like adding attachments and sending.**
- **Python is an object-oriented language, everything in Python is an object.**
- **Classes and Objects are two different terms and should not be used interchangeably, they can sometimes seem like they both refer to the same thing but each has a different meaning.**
- **In summary, objects are to Classes what variables are to Data types.**

Objects

- **the class is the blueprint, an instance is a copy of the class with actual values, literally an object belonging to a specific class.**
- **It's not an idea anymore; it's an actual animal, like a dog named Roger who's eight years old**
- **objects are a data abstraction that capture:**

– internal representation through data attributes

– interface for interacting with object through methods (procedures), defines behaviors but hides implementation

Objects fields and methods

- **Fields and Methods fields store the object's data while methods perform actions to use or modify those data. objects might have no fields or no methods**

Fields

- **The fields of an object are all the data variables that make up that object. They are also sometimes referred to as attributes or member variables.**
- **These fields are usually made up of primitive types like integers or characters, but they can also be objects themselves.**
- **For example a book object may contain fields like title, author and numberOfPages. Then a library object may contain a field named books that will store all book objects in an array.**

Accessing fields

- **Accessing a field in an object is done using the dot operator ‘.’**
- **For example, if we had an object called book that contains the field title. To access the title field you would use book.title**

Setting Fields

- **You can also change a field's value. Say you want to set the number of pages in a book to 234 pages:**

book.numOfPages = 234

Methods

- Methods are functions that belong to a particular object • Calling a method using the dot modifier .
- Methods, just like any function can also take in arguments. For Example: Assume that our book object has a method called setBookmark that takes the page number as a parameter:

```
void setBookmark(int pageNum);
```

- •If you wanted to set a bookmark at page 12, you can call the method and pass in the page number as an argument:

```
book.setBookmark(12);
```

Create a Class

```
class Person:
        def __init__(self, name, age):
                self.name = name self.age = age
        def myfunc(self):
                print("Hello my name is " + self.name)
p1 = Person("John", 36) p1.myfunc()
```

__init__

- **The __init_is called when an instance (object) of the class is created, using the class name as a function.**
- **All methods must have self as their first parameter, although it isn't explicitly passed, Python adds the self argument to the list for you; you do not need to include it when you call the methods. Within a method definition, self refers to the instance calling the method.**

The self Parameter

- **The self parameter is a reference to the class itself, and is used to access variables that belongs to the class. • It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class.**

WHAT IS A METHOD?

- **function that works only with this class**

– Python always passes the actual object as the first argument, convention is to use self as the name of the first argument of all methods • the “.” operatoris used to access any attribute

– a data attribute of an object

– a method of an object

Inheritance

- **Inheritance provides a way to share functionality between classes. Imagine several classes, Cat, Dog, Rabbit and so on. Although they may differ in some ways (only Dog might have the method bark), they are likely to be similar in others (all having the attributes color and name).**
- **This similarity can be expressed by making them all inherit from a superclass Animal, which contains the shared functionality. To inherit a class from another class, put the superclass name in parentheses after the class name.**

Inheritance

- **A class that inherits from another class is called a subclass. A class that is inherited from is called a superclass. If a class inherits from another with the same attributes or methods, it overrides them**

Overriding the Functionality of a Parent Class

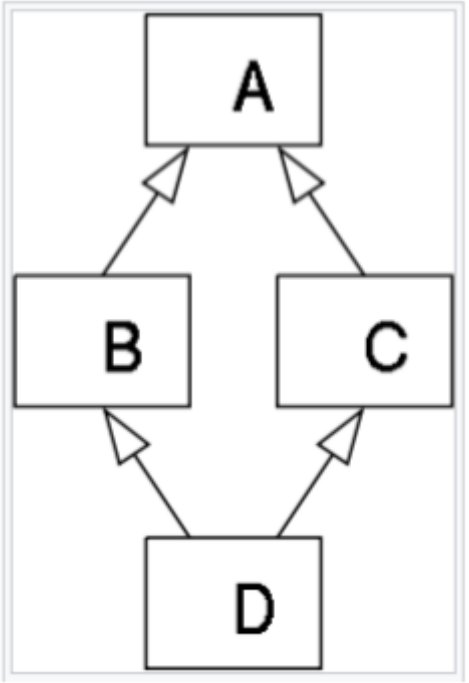
- What is the result of this code?

```
class A:
    def method(self):
        print(1)
class B(A):
        def method(self):
            print(2)
B().method()
```

2

Multiple inheritance

- Multiple inheritance is a feature of some object-oriented computer programming languages in which an object or class can inherit characteristics and features from more than one parent object or parent class



• The "diamond problem" (sometimes referred to as the "deadly diamond of death") is an ambiguity that arises when two classes B and C inherit from A, and class D inherits from both B and C. If there is a method in A that B and C have

Multiple inheritance

```
class A:
    def m(self):
        print("m of A called")

class B(A):
    def m(self):
        print("m of B called")

class C(A):
    def m(self):
        print("m of C called")

class D(B,C):
    pass

d=D()
d.m()
m of B called
```

Multiple inheritance

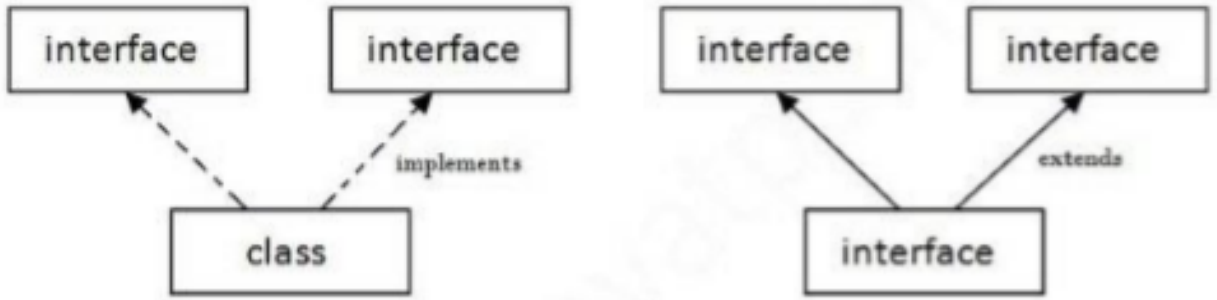
If we transpose the order of the classes in the class header of D in "class D(C,B):"

```
class A:
    def m(self):
        print("m of A called")

class B(A):
    def m(self):
        print("m of B called")

class C(A):
    def m(self):
        print("m of C called")

class D(C,B):
    pass

d=D()
d.m()
```

```
m of C called
```

Java 8SE

```
interface X
{
public void myMethod();
}
interface Y
{
public void myMethod();
}
class Demo implements X, Y
{
public void myMethod()
{
System.out.println(" Multiple inheritance example using interfaces");
}
}
```



Multiple Inheritance in Java