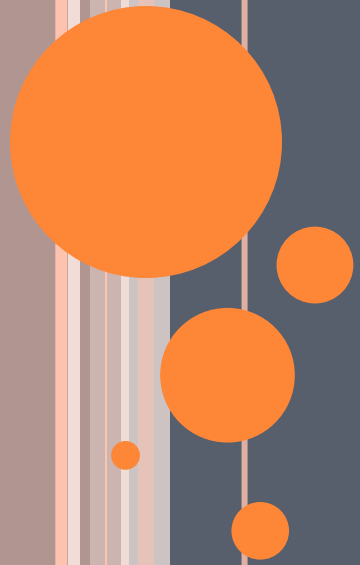
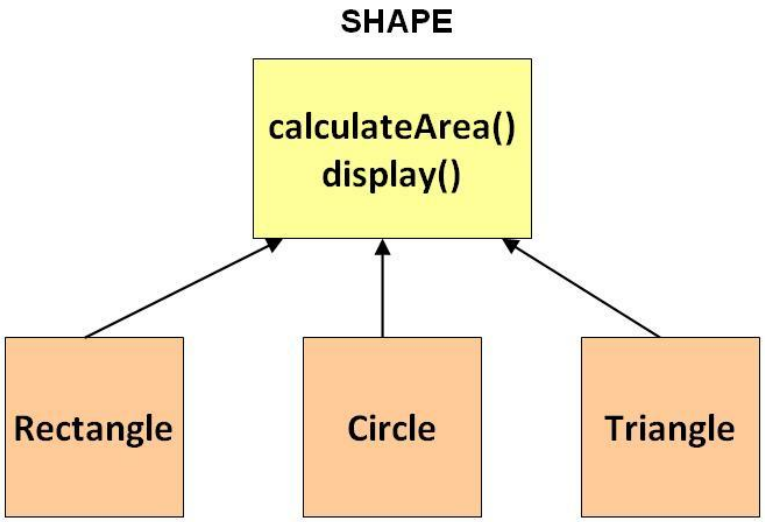
JAVA

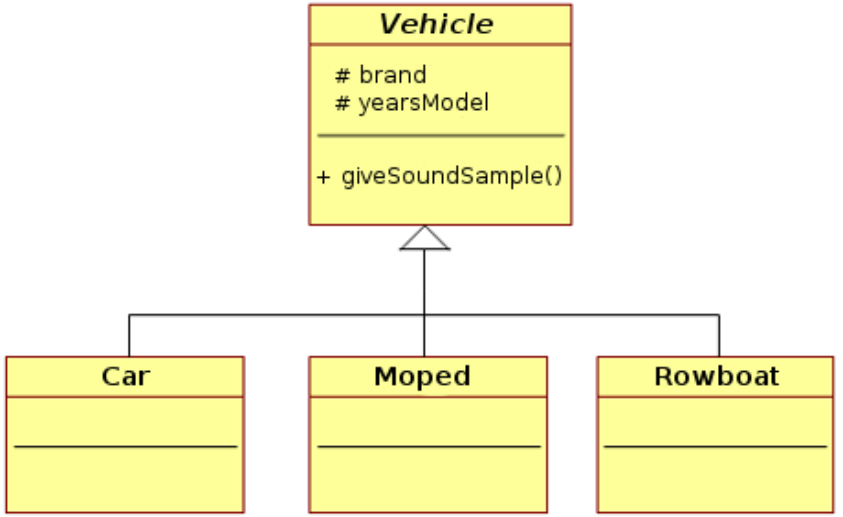
Agenda

- Abstract class
- Interface



Abstract Class


SHAPE
calculateArea()
display()
Rectangle
Circle
Triangle


Vehicle
brand
yearsModel
+ giveSoundSample()
Car
Moped
Rowboat

Syntax

- Abstract Class:

```
abstract class class_name { }
```

- Abstract method:

```
abstract return_type function_name ();    // No definition
```

Abstract Class

- Used to declare common characteristics of subclasses.
- An abstract class cannot be instantiated.
- It can only be used as a superclass for other classes that extend the abstract class.
- Abstract classes are declared with the abstract keyword.
- Abstract classes are used to provide a template or design for concrete subclasses down the inheritance tree.
- An abstract class can contain fields that describe the characteristics and methods that describe the actions that a class can perform.

Abstract Method

- An abstract class can include methods that contain no implementation. These are called abstract methods.
- The abstract method declaration must then end with a semicolon rather than a block.
- If a class has any abstract methods, whether declared or inherited, the entire class must be declared abstract.
- The actual implementations must be provided for the abstract methods in the subclass.
- Any implementation specified can, of course, be overridden by additional subclasses.

```
abstract class Vehicle {

        int numofGears;
        String color;
        abstract boolean hasDiskBrake();
        abstract int getNoofGears();
}
```

Interface

Interface???

- The buttons on the front of your television set, for example, are the interface between you and the electrical wiring on the other side of its plastic casing.
  - You press the "power" button to turn the television on and off
- Similarly , objects define their interaction with the outside world through the methods that they expose.
  - Methods form the object's *interface* with the outside world

Interface in Java?

- An interface is a group of related methods with empty bodies / abstract methods.
- A class implements an interface, thereby inheriting the abstract methods of the interface.

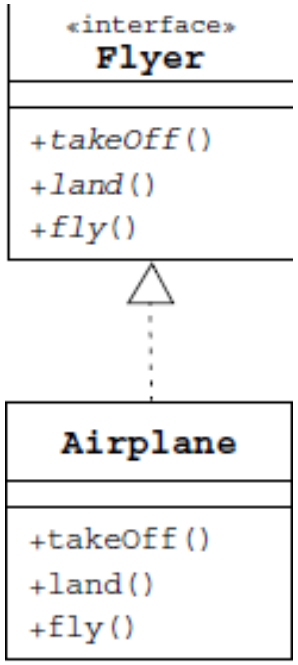
- Syntax of an interface:

```
[visibility] interface InterfaceName [extends other interfaces] {
        constant declarations
        abstract method declarations
}
```

```
public interface Flyer
{
int speed = 10; //public static final int speed=10;
void takeOff(); //public abstract void takeOff();
void land();//public abstract void land();
void fly();//public abstract void fly();
}
```

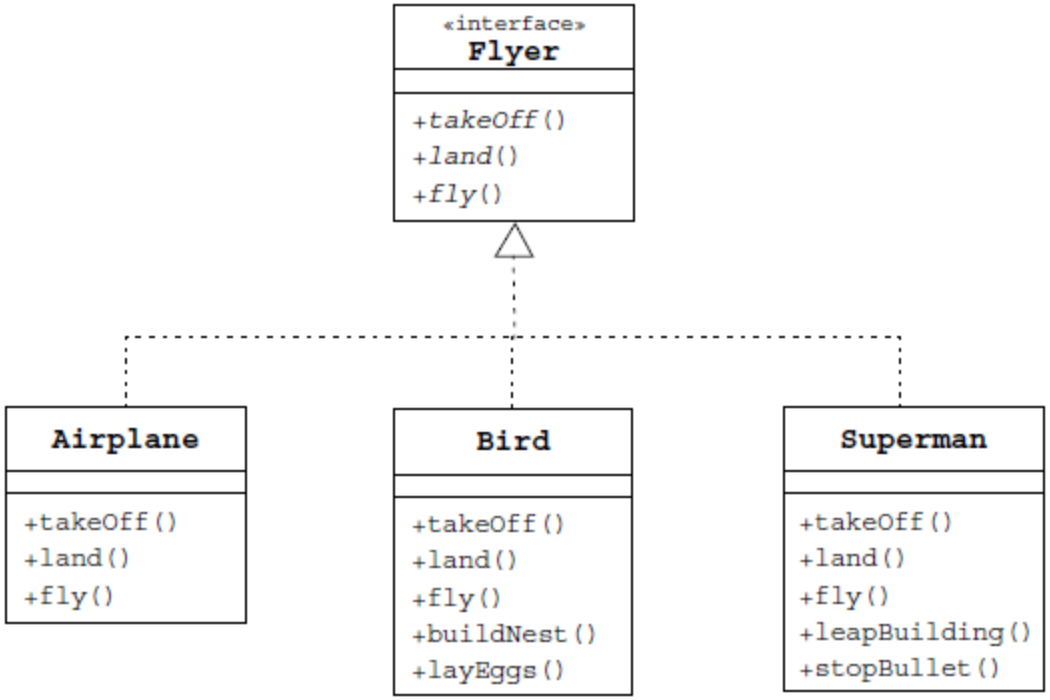
- The java compiler adds
  - **public and abstract** keywords before the interface **method**
  - **public, static and final** keywords before **data members**

```
public interface Flyer
{
public void takeOff();
public void land();
public void fly();
}
```

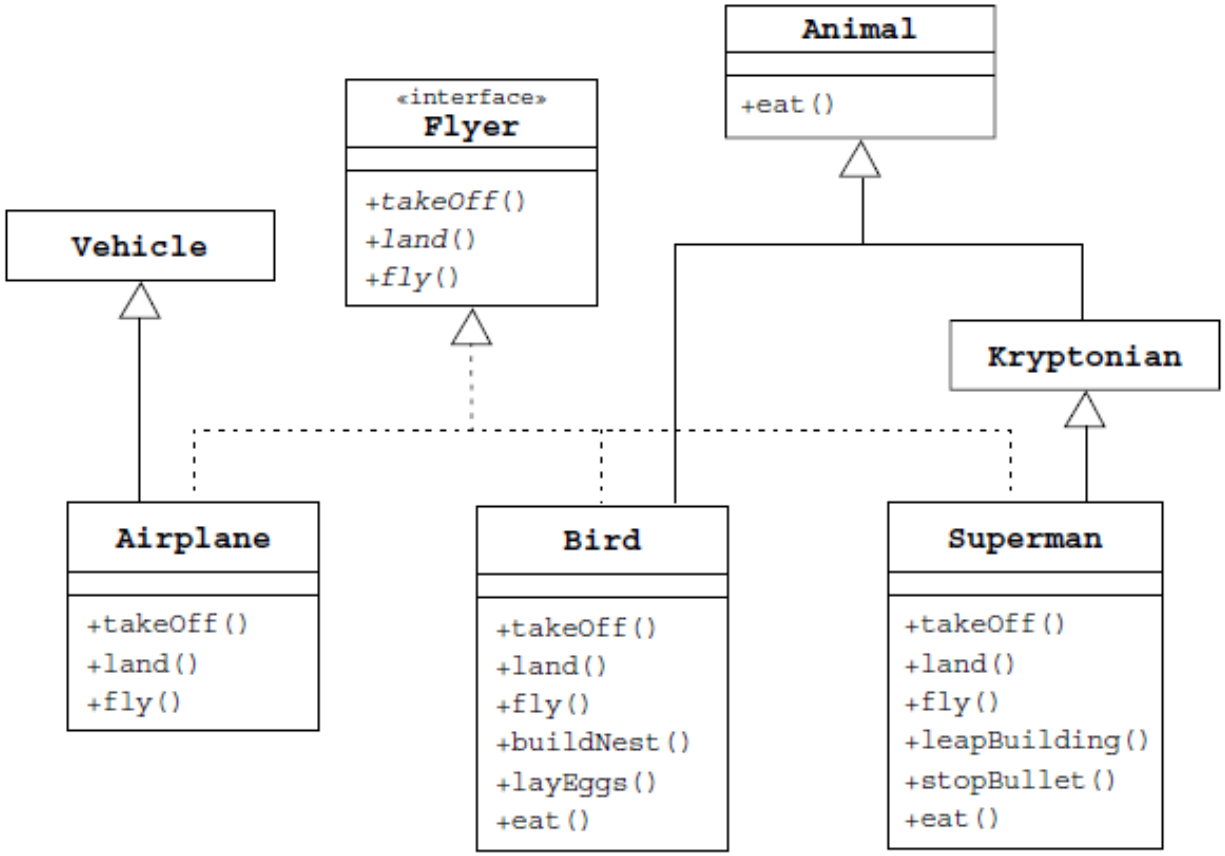



```
public class Airplane implements Flyer {
public void takeOff()
 {
// accelerate until lift-off raise landing gear
}

public void land()
 {
// lower landing gear , decelerate and lower flaps
      until touch-down ,apply brakes
}

public void fly()
 {
// keep those engines running
}

}
```

- Many unrelated classes can implement the same interface.

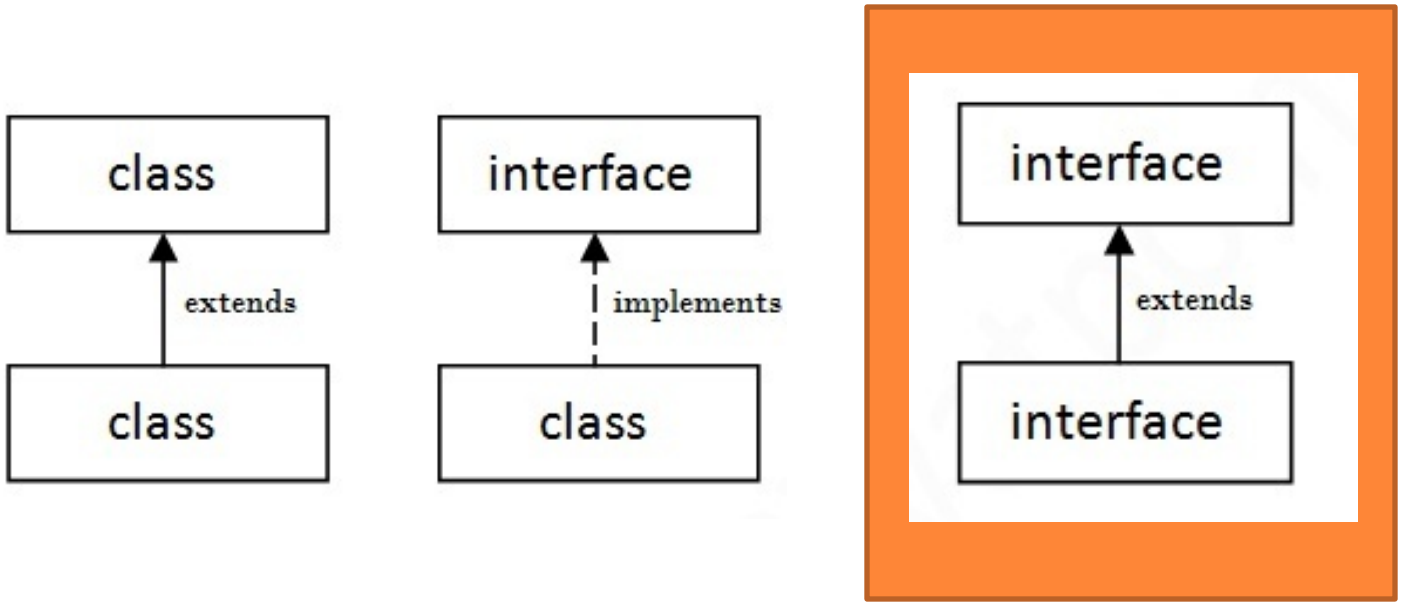


Implements & Extends !!!!



```
public class Bird extends Animal implements Flyer
   {
public void takeOff() { /* take-off implementation */ }
public void land() { /* landing implementation */ }
public void fly() { /* fly implementation */ }
public void buildNest() { /* nest building behavior */ }
public void layEggs() { /* egg laying behavior */ }
public void eat() { /* override eating behavior */ }
}
```

INHERITENCE



Uses of Interfaces

- Declaring methods that one or more classes are expected to implement
- Determining an object's programming interface without revealing the actual body of the class
- Capturing similarities between unrelated classes without forcing a class relationship
- Simulating multiple inheritance by declaring a class that implements several interfaces