Object Oriented Programming

INTERFACE

What is Interface?

- An interface is a completely "abstract class" that is used to group related methods with empty bodies.
- Collection of abstract methods
- It has static constants and abstract methods
- Java Interface also represents the **IS-A relationship**

Why do we need an Interface?

- For fully abstraction
- It supports multiple Inheritance

How interface is similar to class?

- Interface can have any number of methods
- It can have same file extension as class (.java)
- Cannot instantiate an interface
- Interface does not contain constructor
- All methods in interface are abstract
- Cannot contain instance variable
- Can extend multiple interface

Implementing an Interface

- Like abstract classes, we cannot create objects of interfaces.
- To use an interface, other classes must implement it. We use the ***implements*** keyword to implement an interface.

Declaring an Interface

```
interface Animal{

}
```

```
interface <interface_name>{
        // declare constant fields
        // declare methods that abstract

}
```

Interface Method

```
interface Animal{

    void eat();

}
```

Interface Method

```
interface Animal{
        void eat();
}
```


```
interface Animal{
        public abstract void eat();
}
```

- Java compiler adds public and abstract keyword before the interface method automatically.

Interface (variable)

```
interface Animal{
        int speed = 100;
        void eat();

}
```

Compiler

Compiler

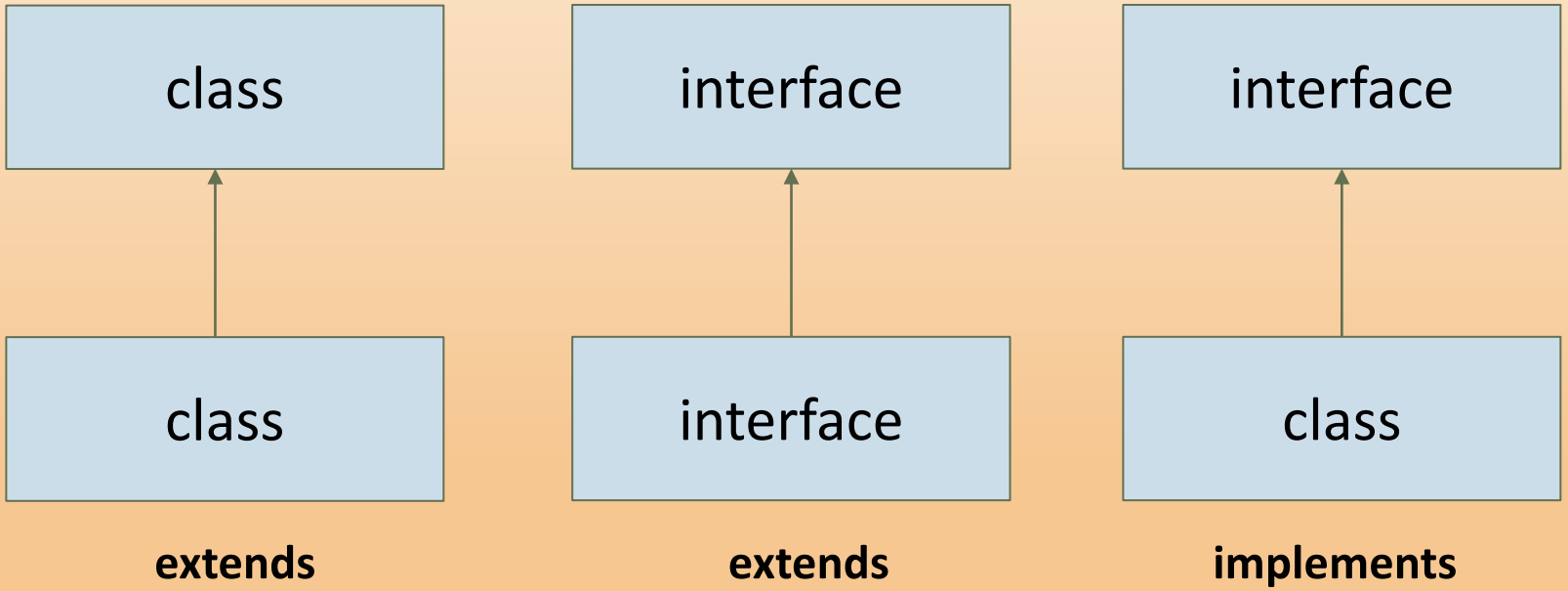
```
interface Animal{
        public static final int speed = 100;
        public abstract void eat();

}
```

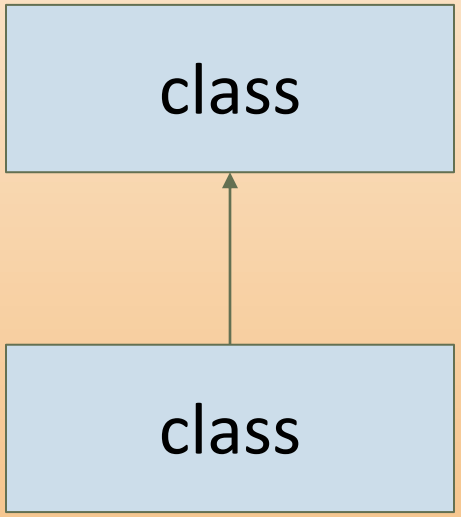
Properties of Interface

- Each method in an interface is also abstract, so no need to use abstract keyword
- Methods in an interface are implicitly public
- A class can inherit just from one superclass, but interface can inherit multiple interfaces.

extends vs. implements

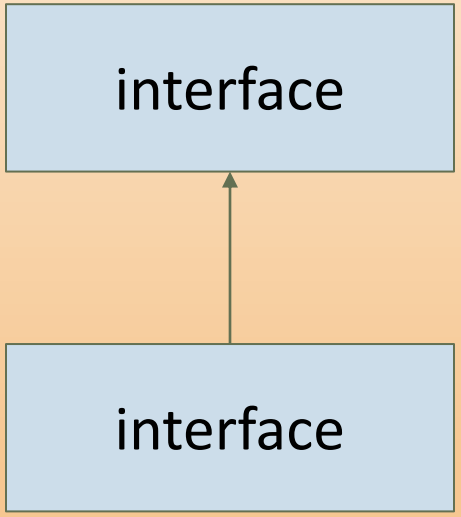


Extends from one class to another




```
class Animal{
.....
.....
}
class Dog extends Animal{
.....
.....
}
```

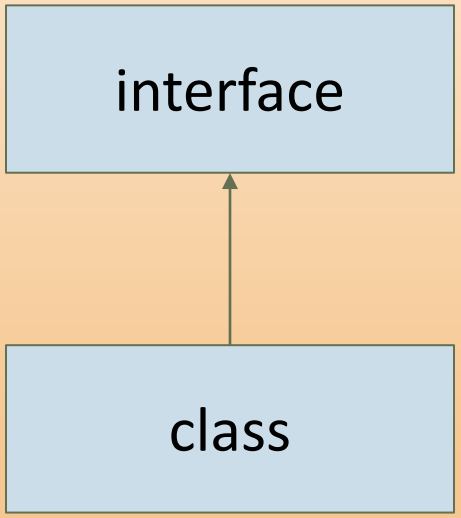
Extends from one interface to another



extends

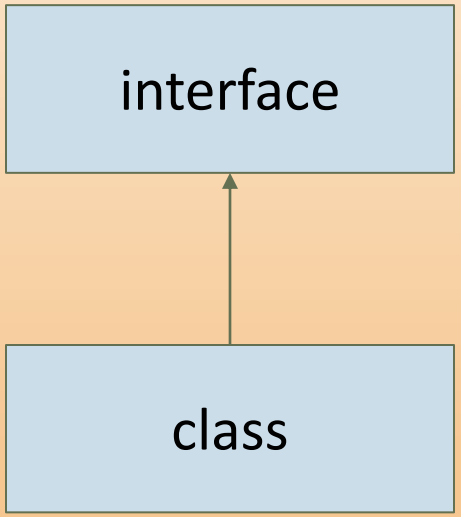
```
interface Animal{
……
……
}
interface Dog extends Animal{
……
……
}
```

Implements an interface from any class



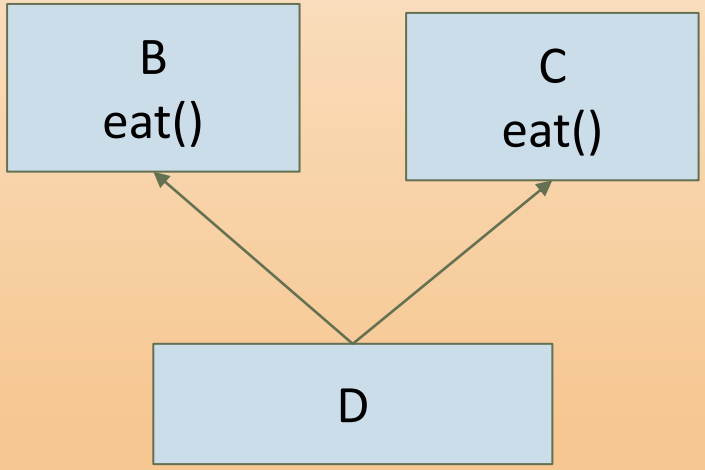

```
interface Animal{
……
……
}
class Dog implements Animal{
……
……
}
```

Implements an interface from any class




```
interface Animal{
        void eat();
}
class Dog implements Animal{
        public void eat(){
            System.out.println(“Dogs can eat”);
}
```

Multiple Inheritance (Why is not possible on Java class?)




```
public class B{
        void eat(){
                System.out.println(“Eating from B”);
        }
}
public class C{
        void eat(){
                System.out.println(“Eating from C”);
        }
}
```

Multiple Inheritance (Why is not possible on Java class?)

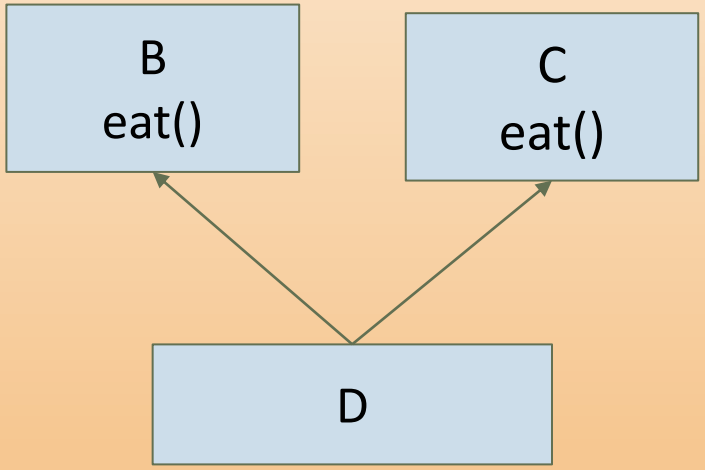
```
public class B{
    void eat(){
        System.out.println("Eating from B");
    }
}
public class C{
    void eat(){
        System.out.println("Eating from C");
    }
}
```

```
public class D extends B,C{
    void eat(){
        System.out.println("Eating from B");
    }
    void eat(){
        System.out.println("Eating from C");
    }
}
```

Multiple Inheritance (Why is not possible on Java class?)

```
public class Test{
    public static void main(String[] args){
        D ob = new D();
        ob.eat(); //ambiguity makes, ERROR
    }
}
```

Multiple Inheritance (interface)




```
interface B{
      void eat();
}

Interface C{
      void eat();
}
```

Multiple Inheritance (interface)

```
interface B{
    void eat();
}
interface C{
    void eat();
}
```

```
public class D implements B,C{
    void eat(){
        System.out.println(“Eating”);
    }
}
```

Multiple Inheritance (interface)

```
public class Test{
    public static void main(String[] args){
        D ob = new D();
        ob.eat();
    }
}
```

Output:

Eating

Implementing Multiple Interfaces

```
interface A {
  // members of A
}
interface B {
  // members of Ba
}
class C implements A, B {
  // abstract members of A
  // abstract members of B
}
```

Example

```
interface Polygon {
  void getArea(int length, int breadth);
}

// implement the Polygon interface
class Rectangle implements Polygon {

  // implementation of abstract method
  public void getArea(int length, int breadth) {
    System.out.println("The area of the rectangle is " + (length * breadth));
  }
}

public class Main {
  public static void main(String[] args) {
    Rectangle r1 = new Rectangle();
    r1.getArea(5, 6);
  }
}
```

Implementing Multiple Interfaces

Interface	Abstract Class
1. Can only have abstract method	1. Can have abstract and non abstract methods
2. It supports multiple inheritance	2. It does not support multiple inheritance
3. Fully abstract	3. Partially abstract
4. Can only have static and final variable	4. Can have static, non static, final and non final variable

5. Example
 	
interface Animal{
void eat();
}
	5. Example
	
	abstract class Animal{
	abstract void eat();
	}

Exercise

- Write a Java program to create an interface **Flyable** with a method called **fly_obj().** Create three classes **Spacecraft, Airplane**, and **Helicopter** that implement the Flyable interface. Implement the **fly_obj()** method for each of the three classes.

Thank You