ENGINEER SERIES

IN JAVA LANGUAGE

5 CH

INTERFACES

ENG : Qusay Khudair

Creativity and Accuracy in Work

AUG PLUS .Edu

Chapter 13

Interface

ENG : Qusay khudair

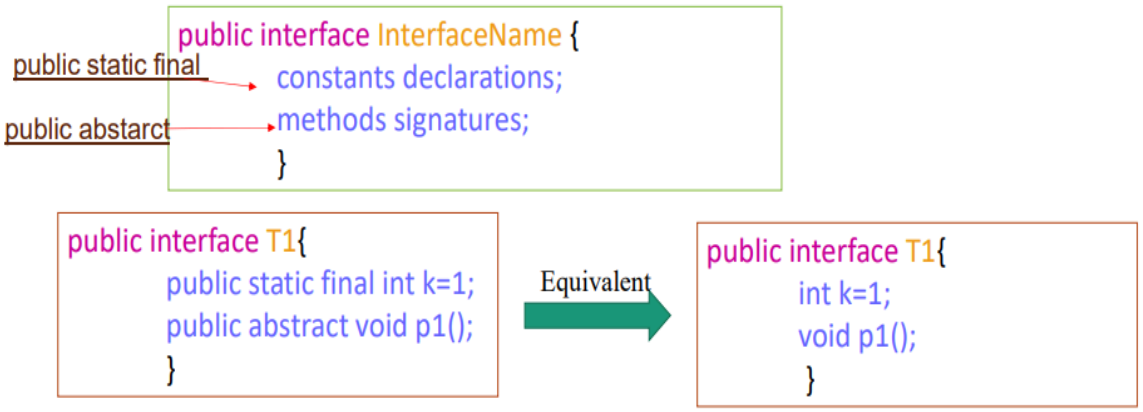
Introduction to Interfaces in Java

- **Definition:**
  - **An Interface in Java is a reference type, similar to a class, that can contain only constants that are public static final by default, method signatures (public only in java 7) . Interfaces cannot contain instance fields or constructors.**
- **Purpose:**
  - **To provide a way to achieve abstraction.**
  - **To define a contract that other classes can implement.**

Interface Syntax

```
interface Animal {
    void sound();  // Abstract method
    void sleep();  // Abstract method
}
```

- **Keywords:**
  - `interface`**: Used to declare an interface.**
  - **Methods in an interface are abstract by default (no body).**



Implementing an Interface

- **Definition:**
  - **A class can implement one or more interfaces, providing concrete implementations for all the abstract methods in those interfaces.**
- **Syntax:**

```
class Dog implements Animal {
    public void sound() {
        System.out.println("Bark");
    }

    public void sleep() {
        System.out.println("Sleeping...");
    }
}
```

- **Keywords:**
  - `implements`: **Used by a class to implement an interface.**
  - **The class must provide implementations for all methods declared in the interface.**

Multiple Interface Implementation

- **Java supports multiple inheritance through interfaces.**

  A class can implement multiple interfaces by separating them with a comma.

- **Example:**

```
interface Pet {
    void play();
}

interface Animal {
    void sound();
    void sleep();
}

class Dog implements Animal, Pet {
    public void sound() {
        System.out.println("Bark");
```

```
}

public void sleep() {
    System.out.println("Sleeping...");
}

public void play() {
    System.out.println("Playing fetch");
}}
```

Interface vs. Abstract Class

Abstract class and Interface

Abstract class	Interface
A programmer uses an abstract class when there are some common features shared by all the objects.	A programmer writes an interface when all the features have different implementations for different objects.
Multiple inheritance not possible (Only one "parent" class)	Multiple inheritance possible Multiple("parent" interfaces)
An abstract class contain both abstract and concrete(non abstract) method	An interface contain only abstract method
In abstract class, abstract keyword is compulsory to declare a method as an abstract	abstract keyword is optional to declare a method as an abstract in interface
An abstract class can have protected, public abstract method	An interface can have only public abstract method
Abstract class contain any type of variable	Interface contain only static final variable (constant)

Interface Inheritance

- **Definition:**
  - **Interfaces can extend other interfaces, forming a hierarchy of interfaces.**

Example:

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

interface Mammal extends Animal {
    void giveBirth();
}

class Dog implements Mammal {
    public void sound() {
        System.out.println("Bark");
    }

    public void giveBirth() {
        System.out.println("Giving birth to
puppies");
    }
}
```

- **Explanation:**
  - `Mammal` **inherits from** `Animal`**, and** `Dog` **must implement methods from both interfaces.**

Real-world Applications of Interfaces

- **Benefits:**
  - **Promote flexibility and reusability.**
  - **Decouple code, making it easier to maintain and test.**
  - **Abstraction and polymorphism .**
