SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23CSB101- OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Unit II – INHERITANCE, PACKAGES AND INTERFACES

Topic : INTERFACE

INTERFACE

An interface is a collection of method definitions (without implementations) and constant values. It is a blueprint of a class. It has static constants and abstract methods.

Three reasons to use interface.

- It is used to achieve fully abstraction.
- support multiple inheritance.
- It can be used to achieve loose coupling.

INTERFACE

- An interface can contain any number of methods.
- Interface name is the name of file with a .java extension
- The bytecode of an interface appears in a .class file.
- Interfaces and bytecode file must be in same packages

INTERFACE

- **Abstract Methods**: An interface can contain abstract methods (methods without a body). Any class that implements the interface must provide an implementation for these methods.
- **Multiple Inheritance**: A class can implement multiple interfaces, which allows for multiple inheritance of type (unlike classes, where Java allows only single inheritance).
- **Fields**: All fields in an interface are public, static, and final by default, meaning they are constants.
- **Default Methods**: default methods in interfaces, allow interfaces to have methods with implementations. Classes that implement the interface can choose to override these methods.
- **Static Methods**: static methods in interfaces belong to the interface itself, not to any instance of the implementing class.

INTERFACE

```
[access_specifier] interface InterfaceName
{
Datatype VariableName1=value;
Datatype VariableName2=value;
.
.
Datatype VariableNameN=value;
returnType methodName1(parameter_list);
returnType methodName2(parameter_list);
.
.
returnType methodNameN(parameter_list);
}
```

Where,

Access_specifer :

either public or none.

Name:

name of an interface can be any valid java identifier.

Variables:

They are implicitly public, final and static, meaning that they cannot be changed by the implementing class. They must be initialized with a constant value.

Methods:

They are implicitly public and abstract, meaning that they must be declared without body and defined only by the implementing class.

INTERFACE

```
// Interface definition
interface Animal {
    void sound(); // Abstract method
    default void sleep() {
        System.out.println("The animal is sleeping.");
    }
}

// Class implementing the interface
class Dog implements Animal {
    // Providing implementation for the abstract method
    public void sound() {
        System.out.println("The dog barks.");
    }
    // Optionally, the sleep method can be overridden
    public void sleep() {
        System.out.println("The dog is sleeping.");
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.sound();     // Output: The dog barks.
        dog.sleep();     // Output: The dog is sleeping.
    }
}
```

INTERFACE

Interface vs Abstract Class:

- An **abstract class** can have both abstract and concrete methods, and it can have instance variables.
- An **interface** cannot have instance variables (it only has constants), and all methods are abstract (except for default and static methods).

INTERFACES

- **Class**:
  - A **class** is a blueprint for creating objects (instances), and it defines the state (fields) and behavior (methods) that objects of the class can have.
  - It is used to represent real-world entities or concepts and can be instantiated directly.
- **Interface**:
  - An **interface** is a reference type that defines a contract of methods that a class must implement. It does not provide implementation details for those methods (unless they are default or static).
  - It is used to represent a contract or capability that other classes can implement.

INTERFACE

CATEGORY	CLASS	INTERFACE
Definition and Purpose	A class is a blueprint for creating objects	An interface is a reference type
Keyword	Declared using the **class** keyword. class MyClass	Declared using the **interface** keyword. interface MyInterface
Methods	Contain both instance , abstract and concrete and static methods. Can have any access modifiers	Contain static, default and abstract methods implicitly public and abstract
Fields/Variables	Can have instance variables and any access modifier	Implicitly public, static, and final, meaning they are constants.
Inheritance	Can inherit from only one other class	Cannot inherit from a class.
Constructors	Have multiple constructors (overloaded constructors)	Do not have constructors
Instantiation	Can be instantiated to create an object using new	Cannot be instantiated directly.


Thank
you