SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

19CS204 OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Topic – Interface

Interface

- Using the keyword interface, you can fully abstract a class' interface from its implementation.
- Using interface, you can specify what a class must do, but not how it does it.
- Interfaces are syntactically similar to classes, but they lack instance variables, and,
- Methods are declared without any body in Interfaces. So we cannot implement methods in interface.
- Once it is defined, any number of classes can implement an interface.
- Also, one class can implement any number of interfaces.

Interface

Defining an Interface

- An interface is defined much like a class. This is a simplified general form of an interface:

```
access interface name {
return-type method-name1(parameter-list);
return-type method-name2(parameter-list);
type final-varname1 = value;
type final-varname2 = value;
//...
return-type method-nameN(parameter-list);
type final-varnameN = value;
}
```

Interface

Example of an interface definition

```
interface Callback {
void callback(int param);
}
```

Implementing Interfaces

- Once an interface has been defined, one or more classes can implement that interface.

- To implement an interface, include the implements clause in a class definition, and then create the methods required by the interface.

The general form of a class that includes the implements clause looks like this:

```
class classname [extends superclass] [implements interface [,interface...]]       {
// class-body
}
```

Interface

- If a class implements more than one interface, the interfaces are separated with a comma.
- The methods that implement an interface must be declared public.
- Also, the type signature of the implementing method must match exactly the type signature specified in the interface definition.

```
class Client implements Callback {
// Implement Callback's interface
public void callback(int p) {
System.out.println("callback called with " + p);
}
}
```

Interface

Accessing Implementations Through Interface References

- You can declare variables as object references that use an interface rather than a class type.

- Any instance of any class that implements the declared interface can be referred to by such a variable.

- The following example calls the callback( ) method via an interface reference variable:

```
class TestIface {
public static void main(String args[]) {
Callback c = new Client();
c.callback(42);
}
}
```

Interface

Example

```
interface Callback {
void callback(int param);
}

class Client implements Callback {
// Implement Callback's interface
public void callback(int p) {
System.out.println("callback called with " + p);
}
}

class TestIface {
public static void main(String args[]) {
Callback c = new Client();
c.callback(42);
}
}
```

Interface

Example

```
interface Callback {
void callback(int param);
}

class Client implements Callback {
// Implement Callback's interface
public void callback(int p) {
System.out.println("callback called with " + p);
}
}

// Another implementation of Callback.
class AnotherClient implements Callback {
// Implement Callback's interface
public void callback(int p) {
System.out.println("Another version of callback");
System.out.println("p squared is " + (p*p));
}
}
```

```
class TestIface2 {
public static void main(String args[]) {
Callback c = new Client();
AnotherClient ob = new AnotherClient();
c.callback(42);
c = ob; // c now refers to AnotherClient object
c.callback(42);
}
}
```

The output from this program is shown here:

```
callback called with 42
Another version of callback
p squared is 1764
```

THANK YOU