Interfaces

- *Interface* is a conceptual entity similar to a Abstract class.
- Can contain only constants (final variables) and abstract method (no implementation) - Different from Abstract classes.
- A concrete class must implement the interface (all the abstract methods of the Interface)

Interfaces Definition

- Syntax (appears like abstract class):

```
interface InterfaceName {
// Constant/Final Variable Declaration
// Methods Declaration
}
```

- Example:

```
interface animal {
public void voice( );
}
```

Implementing Interfaces

- Interfaces are used like super-classes who properties are inherited by classes. This is achieved by creating a class that implements the given interface as follows:

```
class ClassName implements InterfaceName 1,
InterfaceName2, …
{
// Body of Class
}
```

EXAMPLE

- INTERFACE ONE {
- VOID DISPLAY();
- }
- CLASS DEMO IMPLEMENTS ONE {
- PUBLIC VOID DISPLAY() {
- SYSTEM.OUT.PRINTLN("INTERFACE METHOD"); }
- VOID SHOW() {
- SYSTEM.OUT.PRINTLN("METHOD FROM DEMO CLASS");
- } }

- CLASS INTERFACE1
- {
- PUBLIC STATIC VOID MAIN(STRING ARGS[])
- {
- DEMO D=NEW DEMO();
- D.DISPLAY();
- D.SHOW();
- }
- }

Extending Interfaces

- Like classes, interfaces can also be extended. The new sub-interface will inherit all the members of the super interface in the manner similar to classes. This is achieved by using the keyword **extends** as follows:

```
interface InterfaceName2 extends InterfaceName1
{
// Body of InterfaceName2
}
```

MULTIPLE INHERITANCE

- INTERFACE ONE {
- INT A=10;
- }
- INTERFACE TWO {
- INT B=20;
- }
- INTERFACE THREE EXTENDS ONE,TWO
- {
- VOID SUM();
- }

- CLASS DEMO IMPLEMENTS THREE {
- PUBLIC VOID SUM() {
- INT SUM=A+B;
- SYSTEM.OUT.PRINTLN(SUM); } }
- CLASS MULITIPLEINHERITANCE {
- PUBLIC STATIC VOID MAIN(STRING ARGS[]) {
- DEMO D=NEW DEMO();
- D.SUM(); }
- }