HNDIT3012 - OBJECT ORIENTED PROGRAMMING

LECTURE 09 – JAVA INTERFACES

INTERFACES

- An interface is a completely "**abstract class**" that is used to group related methods with empty bodies:

- To access the interface methods, the interface must be "implemented" (like inherited) by another class with the implements keyword (instead of extends).

- The body of the interface method is provided by the "implement" class

- Interfaces **cannot** be used to create objects

FEATURES OF INTERFACE CLASS

- Interface methods do not have a body - the body is provided by the "implement" class
- On implementation of an interface, you must override all of its methods
- Interface methods are by default abstract and public
- Interface attributes are by default public, static and final
- An interface cannot contain a constructor (as it cannot be used to create objects)
- "multiple inheritance" can be achieved with interfaces, because the class can **implement** multiple interfaces.

EXAMPLE

```
// interface
interface Animal {
    public void animalSound(); // interface method (does not have a
body)

    public void run(); // interface method (does not have a body)
}
```

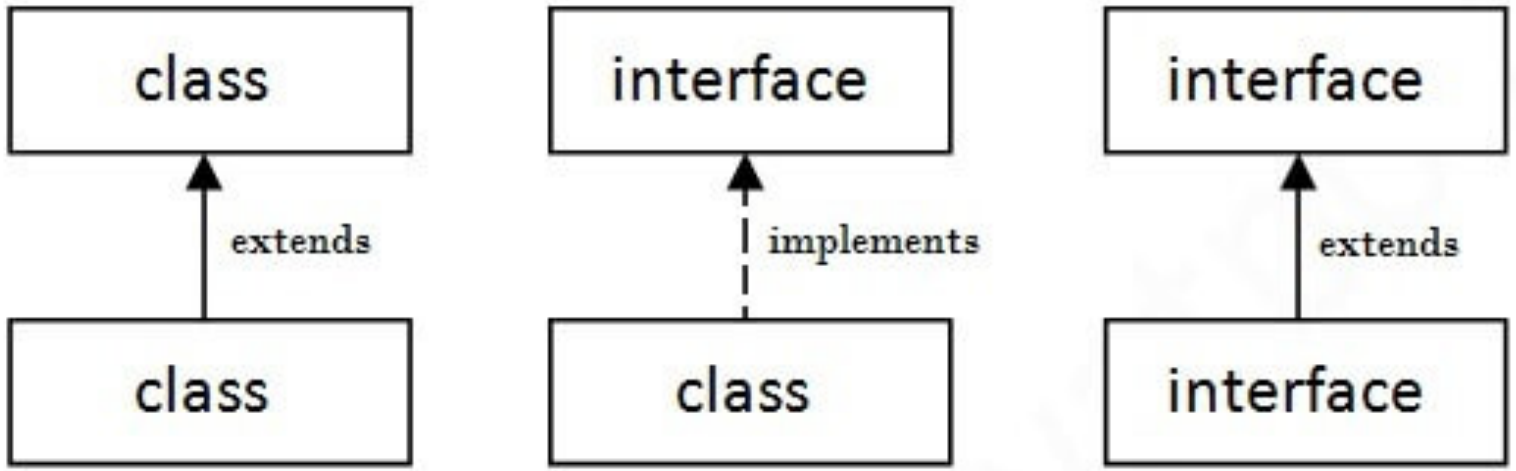
EXAMPLE

```
class Lion implements Animal {
  public void animalSound() {
    System.out.println("The lion roar…..");
  }
  public void sleep() {
    System.out.println("Zzz");
  }
public static void main(String[] args) {
    Lion myLion = new Lion();
    myLion.animalSound();
    myLion.sleep();
  }
}
```

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

THE RELATIONSHIP BETWEEN CLASSES AND INTERFACES



IMPLEMENTING MULTIPLE INTERFACES

```
interface A {
    // members of A
 }

interface B {
    // members of B
}

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

MULTIPLE INTERFACES

```
interface FirstInterface {
	public void myMethod(); // interface method
}
interface SecondInterface {
	public void myOtherMethod(); // interface method
}
class DemoClass implements FirstInterface, SecondInterface {
	public void myMethod() {
		System.out.println("Some text..");
}
public void myOtherMethod() {
	System.out.println("Some other text...");
 }
public static void main(String[] args) {
	DemoClass myObj = new DemoClass();
	myObj.myMethod();
	myObj.myOtherMethod();
	}
}
```

To implement multiple interfaces, separate them with a comma,

EXTENDING AN INTERFACE

```
interface Line {
      // members of Line interface
 }

// extending interface
interface Polygon extends Line {
      // members of Polygon interface
      // members of Line interface
}
```

EXTENDING MULTIPLE INTERFACES

```
interface A {
    ...
}

interface B {
    ...
}

interface C extends A, B {
    ...
}
```

DEFAULT METHOD IN JAVA INTERFACE

```
interface Polygon {
void getArea();

// default method
default void getSides() {
      System.out.println("I can get sides of a polygon.");
}
}
```

```
// implements the interface
class Square implements Polygon {
public void getArea() {
int length = 5;
int area = length * length;
System.out.println("The area of the square is " + area);
}
}
```

```
// implements the interface
class Rectangle implements Polygon {
public void getArea() {
int length = 6;
int breadth = 5;
int area = length * breadth;
System.out.println("The area of the rectangle is " +
area);
}

// overrides the getSides()
public void getSides() {
      System.out.println("I have 4 sides.");
}
}
```

MAIN METHOD

```
class Main {
public static void main(String[] args) {
// create an object of Rectangle
Rectangle r1 = new Rectangle();
r1.getArea();
r1.getSides();

// create an object of Square
Square s1 = new Square();
s1.getArea();
s1.getSides();
}
}
```

Output

```
The area of the rectangle is 30
I have 4 sides.
The area of the square is 25
I can get sides of a polygon.
```

ABSTRACT CLASS VS INTERFACES

Abstract Class	Interface
An Abstract class doesn't provide full abstraction	Interface does provide full abstraction
Using Abstract we can not achieve multiple inheritance	using an Interface we can achieve multiple inheritance.
We can declare a member field	We can not declare a member field in an Interface
An abstract class can contain access modifiers for the subs, functions, properties	We can not use any access modifier i.e. public , private , protected , internal etc. because within an interface by default everything is public
An abstract class can be defined	An Interface member cannot be defined using the keyword static, virtual, abstract or sealed
A class may inherit only one abstract class.	A class may inherit several interfaces.
An abstract class can provide complete, default code and/or just the details that have to be overridden.	An interface cannot provide any code, just the signature.

	Interface	Abstract Class
Constructors	✗	✓
Static Fields	✓	✓
Non-static Fields	✗	✓
Final Fields	✓	✓
Non-final Fields	✗	✓
Private Fields & Methods	✗	✓
Protected Fields & Methods	✗	✓
Public Fields & Methods	✓	✓
Abstract methods	✓	✓
Static Methods	✓	✓
Final Methods	✗	✓
Non-final Methods	✓	✓
Default Methods	✓	✗

QUESTION

```
interface Account{
        double amount=100000.00;
        double interestRate();
}
class FixedAccount implements Account{
        public double interestRate(){
                return 10.0;
        }
}
class SavingsAccount implements Account{
        public double interestRate(){
                return 20.0;
        }
}
class AccountType{
        public static void main(String args[]){
                Account acc1=new FixedAccount();
                System.out.println(acc1.interestRate());
        }
}
```

Explain the code and write down the output.

```
interface Intce{
final int x =100;
void show();
}

class TestPr1 implements Intce{
    public void show(){
            System.out.println("show this");
            }

            public static void main(String [] args){
            TestPr1 ob1=new Testpr1();
            ob1.show();
            System.out.println(x);
            }
            }
```