Website

https://www.curiousinfotechsolutions.com



# Java Interface

# Java Programming

## Day-18

Email id:
**info@curiousinfotechsolutions.com**

# Syllabus Day-18

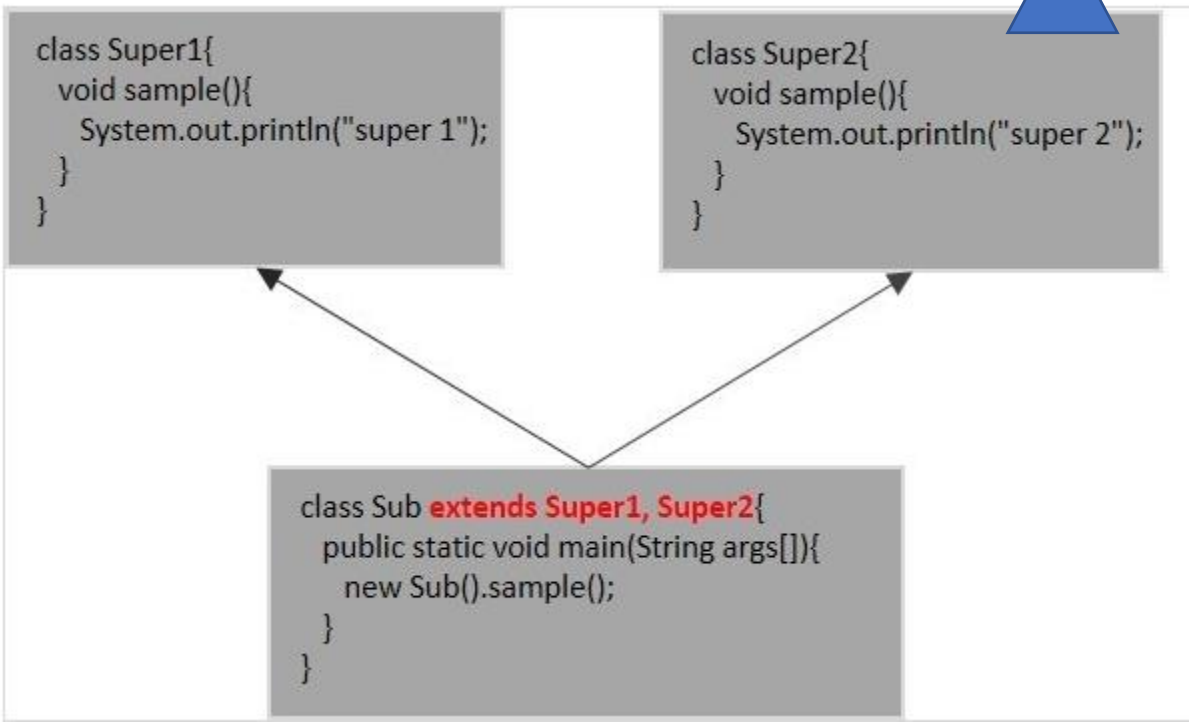


- **Brief**
  - ✓Why multiple inheritance is not supported by Java?
  - ✓What is an Interfaces in Java?
  - ✓How can Define Interface in Java?
  - ✓How can Implement Interfaces in Java?
  - ✓What is Extending Interface in Java?
  - ✓Explain Various Forms of interface implementation.
  - ✓How can Implement Multiple Inheritance in Java?
  - ✓Difference between Class and Interface.

# Why multiple inheritance is not supported by Java?



- For example,
  - if there is a class named Sub
  - and there are **two classes**
    - Super1 and Super2 and
    - if both *contains a method named* ***sample()***.
- And **if the class sub** inherits both classes Super1 and Super2
  - then there will be two copies of the sampling method one from each superclass and *it is ambiguous to decide which method to be executed.*



# Interfaces in Java





- Another way to achieve abstraction in Java, is with interfaces.
- An interface is basically a kind of class.
- Like classes, interfaces contain methods and variables
  - But with a major difference.
- The difference is that *interface define only abstract methods and final fields.*
- This means that interface do not specify any code to implement these methods and data fields contain only constants.

# Defining Interface




- static final type VariableName = Value;
- Return-type methodName1 (parameter_list);

Syntax

```
interface InterfaceName
{
        variable declarations;
        method declaration;
}
```

**Note:**

interface is a keyword.

InterfaceName is any valid Java variable.

- Example-1
  - // interface
  - interface Shape {
  - public void Area(); // interface method (does not have a body)
  - public void Volume(); // interface method (does not have a body)
  - }


# Interface Examples





- Example-2
  - // interface
  - interface item {
  - static final int code = 1001;
  - static final String name = “Tubelight”;
  - void display();
  - }

# Implementing Interfaces





- Interfaces are used as “super classes” whose properties are inherited by classes.
- It is therefore necessary to create a class that inherits the given interface.
- Syntax
  - class className implements InterfaceName
    - {
      - Body of className;
    - }

Syntax

```
interface InterfaceName
{
        variable declarations;
        method declaration;
}
```

**Note:**

interface is a keyword.

InterfaceName is any valid Java variable.

# Implementing Interface





```
interface A {
static final int x=10;
public void myA(); // interface method without code
}
class DemoClass implements A {
public void myA() {
System.out.println("x="+x); //x=10
}
}
```

```
class Main {
  public static void main(String[] args) {
    DemoClass myObj = new DemoClass();
    myObj.myA();//x=10
  }
}
```

1. Can u make object of class? Yes
2. Can u make object of interface? No

# Extending Interface





- Like classes, interfaces can also be extended.
- That is, an interface can be subinterfaced from other interfaces.
- The new subinterface will inherit all the members of the superinterface in the manner similar to subclasses.
- This is achieved using the keyword extends as:
  - interface name2 extends name1
    - {
      - Body of name2
    - }

1. Can u extend class? Yes
2. Can u extend interface? Yes

# Implementing Extending Interface



- interface ItemConstants
- {
  - int code = 1001;
  - String name = “Tubelight”;
- }
- interface ItemMethods
- {
  - void display();
- }

```
interface Item extends ItemConstants, ItemMethods
{
        ……
        ……
}
```



```
interface A {
  static final float pi = 3.14F;
  public float Compute(float Radius); // interface method
 }
class AreaCircle implements A
{
    public float Compute(float Radius)
    {
        return (pi*Radius*Radius);
    }
}
class VolumeSphere implements A
{
    public float Compute(float Radius)
    {
        return (4.0F/3.0F * pi * Radius * Radius * Radius);
    }
}
```

```
class Main {
  public static void main(String[] args) {
    AreaCircle C1 = new AreaCircle();
    VolumeSphere S1 = new VolumeSphere();
    A a1;
    a1 = C1;
    System.out.println(a1.Compute(5));
    a1 = S1;
    System.out.println(a1.Compute(5));
  }
}
```

# Various Forms of interface implementation



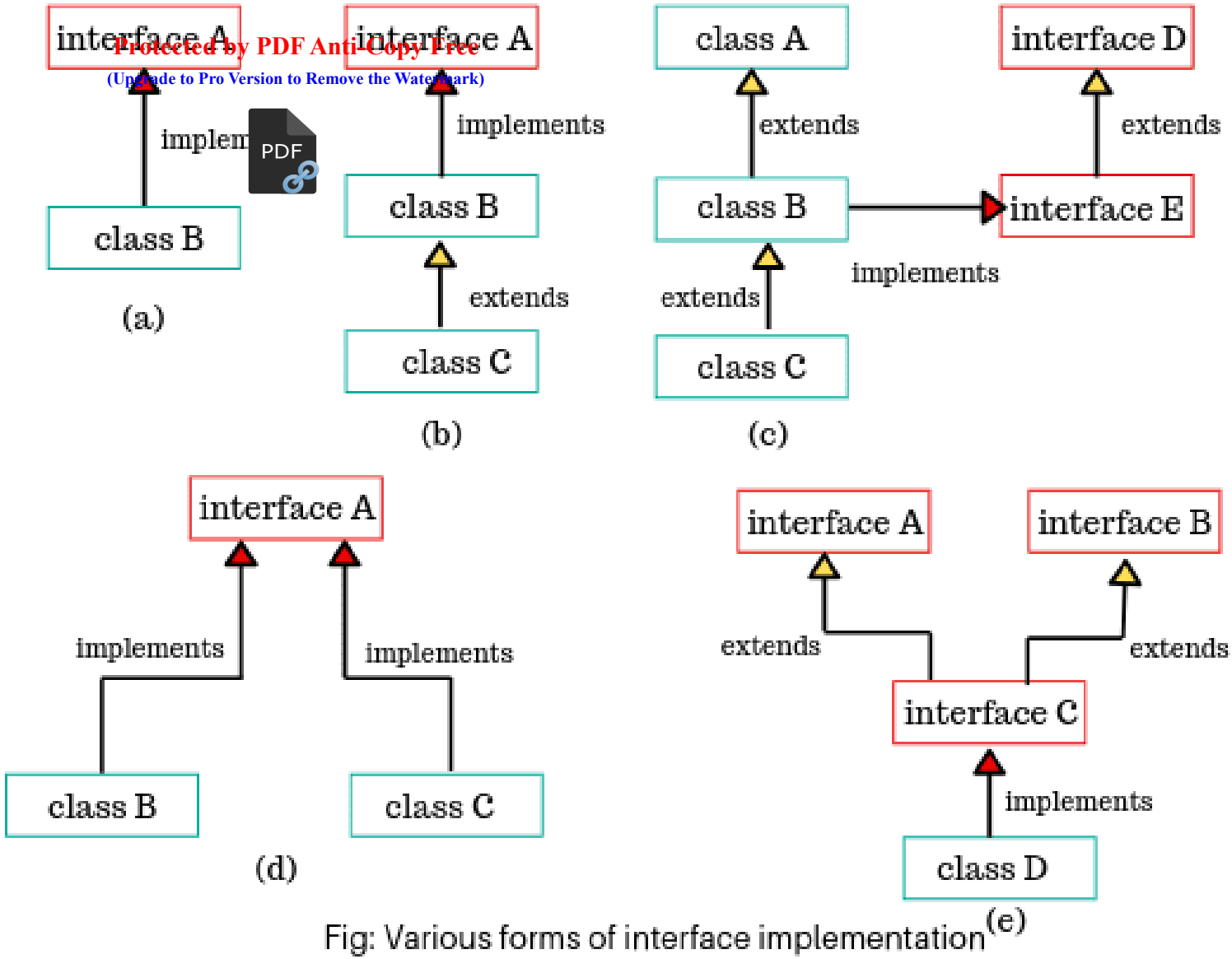
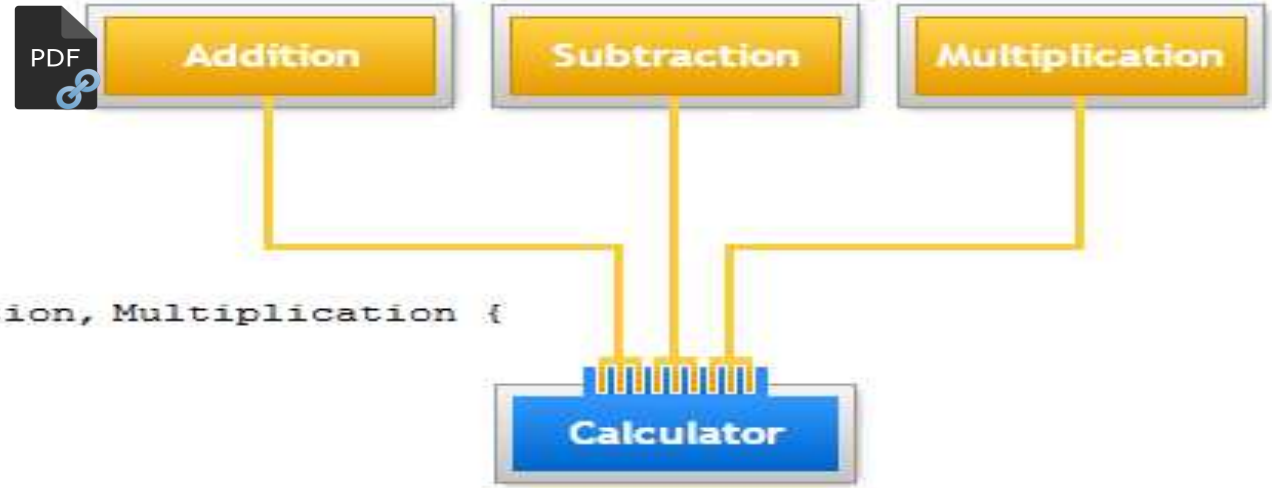



Fig: Various forms of interface implementation



# Implementing Multiple Inheritance




```java
interface Addition {
    int addition(int a,int b);
}
interface Subtraction {
    int subtraction(int a,int b);
}
interface Multiplication {
    int multiplication(int a,int b);
}

class Calculator implements Addition, Subtraction, Multiplication {
    public int addition(int a,int b)
    {
        return a+b;
    }
    public int subtraction(int a,int b)
    {
        return a-b;
    }
    public int multiplication(int a,int b)
    {
        return a*b;
    }
}

public class Javaapp {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Addition       : "+calc.addition(25, 35));
        System.out.println("Subtraction    : "+calc.subtraction(250, 35));
        System.out.println("Multiplication : "+calc.multiplication(25, 100));
    }
}
```

# Difference Between Class and Interface




| Class | Interface |
| --- | --- |
| The keyword used to create a class is “class” | The keyword used to create an interface is “interface” |
| A class can be instantiated i.e, objects of a class can be created. | An Interface cannot be instantiated i.e, objects cannot be created. |
| Classes does not support multiple inheritance. | Interface supports multiple inheritance. |
| It can be inherit another class. | It cannot inherit a class. |
| It can be inherited by another class using the keyword ‘extends’. | It can be inherited by a class by using the keyword ‘implements’ and it can be inherited by an interface using the keyword ‘extends’. |
| It can contain constructors. | It cannot contain constructors. |
| It cannot contain abstract methods. | It contains abstract methods only. |
| Variables and methods in a class can be declared using any access specifier(public, private, default, protected) | All variables and methods in a interface are declared as public. |
| Variables in a class can be static, final or neither. | All variables are static and final. |