Multiple inheritance through interface

To resolve the ambiguity between interfaces we are using super keyword to call parent interface methods.

```
package com.pack1;
                                                                      package com.pack1;
 3 public interface Interface
                                                                      public class ClassA implements InterfaceA, InterfaceB
 4 {
        default void msg()
                                                                           public void msg()
             System.out.println("Java is awesome");
                                                                               System.out.println("We are overridding the msg()");
                                                                               InterfaceA.super.msg();
 9 }
                                                                   9
                                                                          public static void main(String[] args)
10
                                                                  100
                                                                  11
                                                                  12
                                                                               ClassA aobj=new ClassA();
☑ InterfaceB.java ×
                                                                  13
                                                                               aobj.msg();
> 🔀 Training > 💯 src > 🔠 com.pack1 > 0 Interface8 >
                                                                  14
 1 package com.pack1;
                                                                  15 }
                                                                  15
 3 public interface InterfaceB
                                                                  17
                                                                                                                   default void msg()
                                                                  Console X
                                                                 <terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (14-Apr-2025, 7:56:44 am - 7:56:45 am) [pid: 15264]
 6
             System.out.println("Java is amazing!!!");
                                                                 We are overridding the msg()
 8
 9 }
   package com.pack1;
   public interface InterfaceA
                                                                     public class ClassA implements InterfaceA, InterfaceB
 4 {
                                                                   4
 5e
        default void msg()
                                                                          public void msg()
             System.out.println("Java is awesome");
                                                                              System.out.println("We are overridding the msg()");
 8
                                                                              InterfaceA.super.msg();
9 }
                                                                              InterfaceB.super.msg();
10
                                                                  10
                                                                  11=
                                                                          public static void main(String[] args)
                                                                  12
☑ InterfaceB.java ×
                                                                  13
                                                                              ClassA aobj=new ClassA();
▶ 👺 Training ▶ 🕭 src ▶ 🔠 com.pack1 ▶ 🕡 InterfaceB ▶
                                                                  14
                                                                              aobj.msg();
   package com.pack1;
                                                                  16 }
   public interface InterfaceB
 4
 50
        default void msg()
                                                                                                                  <terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (14-Apr-2025, 7:57:20 am - 7:57:20 a
 6
             System.out.println("Java is amazing!!!");
 8
 9 }
```

```
1 package com.pack1;
                                                                      package com.pack1;
   public interface InterfaceA
                                                                      public class ClassA implements InterfaceA, InterfaceB
 4
        default void msg()
                                                                          @Override
                                                                          public void msg()
             System.out.println("Java is awesome");
                                                                               System.out.println("We are overridding the msg()");
                                                                               InterfaceA.super.msg();
10
                                                                  10
                                                                               InterfaceB.super.msg();
                                                                  11
                                                                  12e
                                                                          public static void main(String[] args)
☑ InterfaceB.java ×
                                                                  13
> 🤛 Training → 🚐 src → 🗿 com.pack1 → 🕡 InterfaceB →
                                                                  14
                                                                               ClassA aobj=new ClassA();
 1 package com.pack1;
                                                                  15
                                                                               aobj.msg();
                                                                  16
 3 public interface InterfaceB
                                                                  17 1
        default void msg()
                                                                  Console X
                                                                                                                   6
                                                                 <terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (14-Apr-2025, 7:57:20 am - 7:57:20 am
 7
             System.out.println("Java is amazing!!!");
 8
 9 }
```

Note

Multiple inheritance in java is not supported thought classes, but we can achieve this by using interfaces.

Wherever we are inheriting multiple interfaces into a class, if both the interfaces are having same method name, we can resolve that ambiguity by overriding the ambiguous method. We can call our parent interface methods by using super keyword.

```
1 package com.pack1;
                                                                 14
                                                                             ClassA aobj=new ClassA();
  3 public interface InterfaceA
                                                                 15
                                                                             aobj.msg();
                                                                 16
        default void msg()
                                                                             System.out.println("----");
  58
                                                                 17
 6
                                                                 18
                                                                             InterfaceA obj=new ClassA();
  7
             System.out.println("Java is awesome");
                                                                 19
 8
                                                                 20
                                                                             obj.msg();
 9 }
                                                                 21
10
                                                                 22
                                                                         }
                                                                 23 }
                                                                 24
☑ InterfaceB.java ×
                                                                 25
> 📂 Training > 🌁 src > 🗿 com.pack1 > 0 InterfaceB > 💣 msg(): void
  1 package com.pack1;
                                                                <terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (14-a
  3 public interface InterfaceB
                                                                We are overridding the msg()
                                                                Java is awesome
        default void msg()
                                                                Java is amazing!!!
 6
             System.out.println("Java is amazing!!!");
  7
                                                                We are overridding the msg()
 8
                                                                Java is awesome
 9 }
                                                                Java is amazing!!!
10
                                                                              InterfaceA.super.msg();
1 package com.pack1;
                                                                 10
                                                                              InterfaceB.super.msg();
                                                                 11
   public interface InterfaceA
 3
                                                                 12∞
                                                                         public static void main(String[] args)
 4 {
50
                                                                 13
        default void msg()
                                                                 14
                                                                              ClassA aobj=new ClassA();
 6
                                                                 15
                                                                             aobj.msg();
 7
            System.out.println("Java is awesome");
8
                                                                 16
9 }
                                                                 17
                                                                              System.out.println("----");
                                                                 18
10
                                                                 19
                                                                              InterfaceA obj1=new ClassA();
                                                                 20
                                                                              obj1.msg();
☐ InterfaceB.java ×
                                                                 21
Training ▶ 😕 src ▶ 🔠 com.pack1 ▶ 🛈 Interface8 ▶
                                                                 22
                                                                              System.out.println("----");
 1 package com.pack1;
                                                                 23
 2
                                                                 24
                                                                              InterfaceB obj2=new ClassA();
3 public interface InterfaceB
                                                                 25
                                                                              obj2.msg();
4 {
                                                                 26
 50
        default void msg()
                                                                         }
 6
                                                                 28 }
            System.out.println("Java is amazing!!!");
                                                                 29
 8
                                                                 30
9 }
```

Functional interface

An interface which has only one abstract method is known as a functional interface.

Inside a functional interface we can write any number of default methods, static methods, private methods including main method but there should be only one abstract method.

Functional interfaces concept is being introduced in java-1.8 version

Examples

- 1. Runnable interface
- 2. Consumer interface
- 3. Supplier interface
- 4. Pridicate interface
- 5. Comparable interface
- 6. Comparator interface...etc.

```
package com.pack1;

@FunctionalInterface
public interface InterfaceA

{
   void meth1();

   void meth2();

}
```

```
package com.pack1;

package com.pack1;

### Properties of the properties of the
```

Wait for implementation of functional interface in coming classes

Marker interface

An interface without any methods or variables is known as a marker interface.

It is an empty interface.

Whenever a class is inheriting this marker interface that class will achieve some special properties.

examples

- 1. Serializable interface
- 2. Cloneable interface

Overall oops small example

```
1 package com.pack6;
3 public abstract class Boot
4 {
5
        abstract String osName(String name);
6 }
1 package com.pack6;
                                                 1 package com.pack6;
3 public abstract class Boot
                                                 3 public class OperatingSystem extends Boot
     abstract String osName(String name);
                                                       OperatingSystem()
                                                           System.out.println("Select your preffered OS");
                                                       @Override
                                                10
                                                       String osName(String name)
                                                11
                                                12
                                                           returm name;
                                                13
                                                14 }
```

15

```
1 package com.pack6;
  3 import java.util.Scanner;
  4
  5 public class Start extends OperatingSystem
  6 {
  70
          public static void main(String[] args) throws Exception
 8
           {
 9
                Scanner sc=new Scanner(System.in);
10
11
                Boot os=new OperatingSystem();
12
13
                System.out.println("Please enter your preffered OS to boot");
14
15
                String OSname=os.osName(sc.nextLine());
16
17
                System.out.println(OSname + " is starting....");
18
                Thread.sleep(5000); // Execution will stop for 5 sec
19
                System.out.println("----20%----");
20
                Thread.sleep(5000);
                System.out.println("----50%----");
21
22
                Thread.sleep(5000);
23
                System.out.println("----70%----");
24
                Thread.sleep(5000);
                System.out.println("----100% loaded----");
25
                System.out.println("You can use ur os");
26
27
                sc.close();
28
          }
29 }
                                                                            Select your preffered OS
 1 package com.pack6;
                                public static void main(String[] args) throws Exce^
                                                                            Please enter your preffered OS to boot
   public abstract class Be
                                   Scanner sc=new Scanner(System.in);
                                                                            Linux is starting....
                                                                            ----20%----
                                   Boot os=new OperatingSystem();
                                                                            ----50%----
                                   System.out.println("Please enter your preffere
                                                                             ----19<mark>0% l</mark>oa
                                                                            You can use ur os
                                   String OSname=os.osName(sc.nextLine());

    □ OperatingSystem.java ×

> 😂 > 🚇 > 🌐 > O > 🛦 osName(String) : Strin
                                   System.out.println(OSname + " is starting....
 1 package com.pack6;
                                   Thread.sleep(5000); // Execution will stop for System.out.println("----20%----");
   public class OperatingS
                                   Thread.sleep(5000);
                                   System.out.println("----50%----");
      OperatingSystem()
                                   Thread.sleep(5000);
                                   System.out.println("----70%----");
          System.out.prin
                                   Thread.sleep(5000);
System.out.println("----100% loaded----");
System.out.println("You can use ur os");
      @Override
10
      String osName(String
                                   sc.close();
                          29 }
                          30
14 }
```

1. Encapsulation

Definition: Wrapping data (variables) and methods (functions) into a single unit (class) and restricting direct access to some of the object's components.

Uses:

- Protects internal object state.
- Promotes modularity changes to one part don't affect others.
- Allows data hiding using access modifiers like private, protected, public.
- Example:

```
class Student {
  private int age; // cannot be accessed directly outside

public void setAge(int age) {
    this.age = age;
}
```

public int getAge() {

```
return age;
}
```

2. Inheritance

Definition: The process by which one class (child/subclass) inherits properties and behaviors from another class (parent/superclass).

Uses:

- Reusability of code.
- · Hierarchical classification (e.g., Dog is an Animal).
- Reduces redundancy.
- Example:

```
class Animal {
  void sound() {
    System.out.println("Animal makes sound");
  }
}
```

```
class Dog extends Animal {
  void bark() {
    System.out.println("Dog barks");
  }
}
```

3. Polymorphism

Definition: The ability of one function or object to behave in different ways based on context.

Types:

- Compile-time (method overloading)
- Run-time (method overriding)

Uses:

- Code flexibility and maintainability.
- Improves readability and reusability.
- Example (overriding):

```
class Animal {
  void sound() {
    System.out.println("Animal sound");
  }
}

class Cat extends Animal {
  void sound() {
    System.out.println("Cat meows");
  }
}
```

4. Abstraction

Definition: Hiding complex implementation details and showing only the necessary features of an object.

Uses:

- · Reduces complexity for users.
- Focuses on essential aspects.
- Achieved via abstract classes or interfaces.

• Example:

```
abstract class Shape {
  abstract void draw(); // only declares, doesn't define
}

class Circle extends Shape {
  void draw() {
    System.out.println("Drawing Circle");
  }
}
```

Summary Table:

Concept	Purpose	Benefit
Encapsulation	Hide internal state	Protects data, improves security
<mark>Inheritance</mark>	Reuse code from parent class	Reduces redundancy, enables hierarchy
Polymorphism	One interface, many implementations	Flexibility, cleaner code
Abstraction	Show only relevant features, hide details	Reduces complexity, improves clarity

Assignment

Assignment: Create a Library Management System

Write a Java program that models a simple library management system. Your system should have the following classes:

- **Book Class**: Represents a book with attributes such as title, author, ISBN, and availability status.
- 2. **Library Class**: Represents the library itself, which contains an array of books.
- **Member Class**: Represents a library member with attributes like name, ID, and borrowed books.
- 4. **LibraryApp Class**: Main class to run the program.

Your program should allow the following functionalities:

- 1. Add a book to the library.
- 2. Display all available books.
- 3. Display all borrowed books.
- 4. Allow a member to borrow a book (if available).
- 5. Allow a member to return a book.
- 6. Display all members and the books they've borrowed.