Interfaces I

Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, no body).

- Interfaces specify what a class must do and not how.
- It is the blueprint of the class.
- An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move(). So it specifies a set of methods that the class has to implement.
- If a class implements an interface and does not provide method bodies for all functions specified in the interface, then the class must be declared abstract.
- A Java library example is, Comparator Interface. If a class implements this interface, then it can be used to sort a collection.

Interfaces II

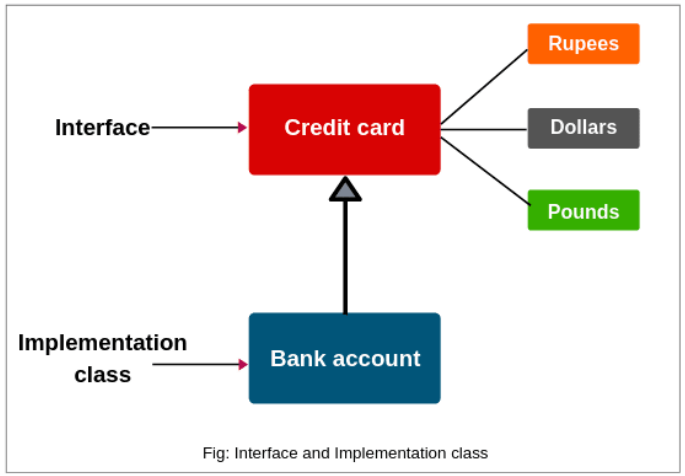

Fig: Interface and Implementation class

Interfaces III

Syntax :

```
1 interface <interface_name>
2 {
3     // declare constant fields
4     // declare methods that abstract by default.
5 }
```

✓ To declare an interface, use *interface* keyword.

✓ *It is used to provide total abstraction. That means all the methods in an interface are declared with an empty body and are public and all fields are public, static and final by default.*

✓ A class that implements an interface must implement all the methods declared in the interface. To implement interface use implements keyword.

Interfaces IV

Why do we use interface ?

- It is used to achieve total abstraction.
- Since java does not support multiple inheritance in case of class, but by using interface it can achieve multiple inheritance .
- It is also used to achieve loose coupling.
- Interfaces are used to implement abstraction. So the question arises why use interfaces when we have abstract classes?

The reason is, abstract classes may contain non-final variables, whereas variables in interface are final, public and static.

Interfaces V

```
1 // A simple interface
2 interface Player
3 {
4       final int id = 10;
5       int move();
6 }
```

Interfaces VI

```
1  public interface MyInterfaceWithState
2  {
3      int someNumber;
4      void methodA();
5  }
```

```
1  public class MyClass implements MyInterface
2  {
3      public void methodA()
4      {
5          System.out.println("Method A called!"
       );
6      }
7      public int methodB()
8      {
9          return 42;
10     }
11     public String methodC(double x, double y
       )
12     {
13         return "x = " + x + ", y = " y;
14     }
15     public static void main (String[] args)
16     {
17         MyInterface object1 = new MyClass();
18         object1.methodA(); // Guaranteed to
       work
19     }
20 }
```

Interfaces VII

✓ To implement an interface we use keyword: implements

```
1  interface Polygon
2  {
3      void getArea(int length, int breadth);
4  }
5  // implement the Polygon interface
6  class Rectangle implements Polygon
7  {
8      // implementation of abstract method
9      public void getArea(int length, int breadth
       )
10     {
11         System.out.println("The area of the
       rectangle is " + (length * breadth));
12     }
13 }
14 class Main
15 {
16     public static void main(String[] args)
17     {
18         Rectangle r1 = new Rectangle();
19         r1.getArea(5, 6);
20     }
21 }
```

Interfaces VIII

```
1  // create an interface
2  interface Language
3  {
4      void getName(String name);
5  }
6  // class implements interface
7  class ProgrammingLanguage implements
       Language
8  {
9
10     // implementation of abstract method
11       public void getName(String name)
12     {
13         System.out.println("Programming
       Language: " + name);
14
15     }
16 }
17 class Main
18 {
19     public static void main(String[] args)
20     {
21         ProgrammingLanguage language =
       new ProgrammingLanguage();
22         language.getName("Java");
23     }
24 }
```

Interfaces IX

```java
// Java program to demonstrate working of
    interface.
import java.io.*;
// A simple interface
interface In1
{
    // public, static and final
    final int a = 10;
    // public and abstract
    void display();
}
// A class that implements the interface.
class TestClass implements In1
{
    // Implementing the capabilities of interface
    .
    public void display()
    {
        System.out.println("Geek");
    }
    // Driver Code
    public static void main (String[] args)
    {
        TestClass t = new TestClass();
        t.display();
        System.out.println(a);
    }
}
```

Interfaces X

A real-world example:

Let's consider the example of vehicles like bicycle, car, bike. . . . . , they have common functionalities. So we make an interface and put all these common functionalities. And lets Bicycle, Bike, car . . . .etc implement all these functionalities in their own class in their own way.

Interfaces XI

```
interface Vehicle
{
        void changeGear(int a);
        void speedUp(int a);
        void applyBrakes(int a);
}
class Bicycle implements Vehicle
{
        int speed;
        int gear;
        public void changeGear(int newGear)
        {
                gear = newGear;
        }
        public void speedUp(int increment)
        {
                speed = speed + increment;
        }
        public void applyBrakes(int decrement)
        {
                speed = speed - decrement;
        }
        public void printStates()
        {
                System.out.println("speed: " + speed  + " gear: " + gear);
        }
}
class Bike implements Vehicle
{
        int speed;
        int gear;
        public void changeGear(int newGear)
                gear = newGear;
        }
        public void speedUp(int increment)
        {
                speed = speed + increment;
        }
        public void applyBrakes(int decrement)
        {
                speed = speed - decrement;
        }
        public void printStates()
        {
                System.out.println("speed: " + speed + " gear: " + gear);
        }
}
class GFG
{
        public static void main (String[] args)
        {
                Bicycle bicycle = new Bicycle();
                bicycle.changeGear(2);
                bicycle.speedUp(3);
                bicycle.applyBrakes(1);
                System.out.println("Bicycle present state :");
                bicycle.printStates();
                Bike bike = new Bike();
                bike.changeGear(1);
                bike.speedUp(4);
                bike.applyBrakes(3);
                System.out.println("Bike present state :");
                bike.printStates();
        }
}
```

Revisit Interfaces I

- Interfaces should contain only abstract methods. Interfaces should not contain a single concrete method.

```
1  interface InterfaceClass
2  {
3      abstract void abstractMethodOne();  //abstract method
4      abstract void abstractMethodTwo();  //abstract method
5      void concreteMethod()
6      {
7          //Compile Time Error.
8          //Concrete Methods are not allowed in interface
9      }
10 }
```

Revisit Interfaces II

- Interface can have two types of members. 1) Fields 2) Abstract Methods.

```
1 interface InterfaceClass
2 {
3       int i = 0; //Field
4       abstract void abstractMethodOne(); //abstract method
5       abstract void abstractMethodTwo(); //abstract method
6 }
```

Revisit Interfaces III

- By default, Every field of an interface is public, static and final . You can't use any other modifiers other than these three for a field of an interface.

```
1 interface InterfaceClass
2 {
3       int i = 0;
4       //By default, field is public, static and final
5       //Following statements give compile time errors
6       private double d = 10;
7       protected long l = 15;
8       //You can't use any other modifiers other than public, static and final
9 }
```

Revisit Interfaces IV

- You can't change the value of a field once they are initialized. Because they are static and final. Therefore, sometimes fields are called as Constants.

```
1  interface InterfaceClass
2  {
3      int i = 0;
4  }
5  class AnyClass implements InterfaceClass
6  {
7      void methodOne()
8      {
9          //Following statement gives compile time error.
10         InterfaceClass.i = 10; //final field can not be re-assigned
11     }
12 }
```

Revisit Interfaces V

- By default, All methods of an interface are public and abstract.

```
1 interface InterfaceClass
2 {
3       void abstractMethodOne();  //Abstract method
4
5       void abstractMethodTwo();  //Abstract Method
6
7       //No need to use abstract keyword,
8       //by default methods are public and abstract
9 }
```

Revisit Interfaces VI

- Like classes, for every interface .class file will be generated after compilation.
- While implementing any interface methods inside a class, that method must be declared as public. Because, according to method overriding rule, you can't reduce visibility of super class method. By default, every member of an interface is public and while implementing you should not reduce this visibility.

```
1  interface InterfaceClass
2  {
3          void methodOne();
4  }
5
6  class AnyClass implements InterfaceClass
7  {
8          void methodOne()
9          {
10                 //It gives compile time error.
11                 //Interface methods must be implemented as public
12         }
13 }
```

Revisit Interfaces VII

- By default, Interface itself is not public but by default interface itself is abstract like below,

```
1 abstract interface InterfaceClass
2 {
3       //By default interface is abstract
4       //No need to use abstract keyword
5 }
```

Revisit Interfaces VIII

- SIB – Static Initialization Block and IIB – Instance Initialization Block are not allowed in interfaces.

```
1  interface InterfaceClassOne
2  {
3      static
4      {
5          //compile time error
6          //SIB's are not allowed in interfaces
7      }
8
9      {
10         //Here also compile time error.
11         //IIB's are not allowed in interfaces
12     }
13
14     void methodOne();  //abstract method
```

Revisit Interfaces IX

```
15 }
```

Revisit Interfaces X

- As we all know that, any class in java can not extend more than one class. But class can implement more than one interfaces. This is how multiple inheritance is implemented in java.

Revisit Interfaces XI

```
1  interface InterfaceClassOne
2  {
3      void methodOne();
4  }
5
6  interface InterfaceClassTwo
7  {
8      void methodTwo();
9  }
10
11 class AnyClass implements
       InterfaceClassOne, InterfaceClassTwo
12 {
13     public void methodOne()
14     {
15         //method of first interface is
           implemented
16     }
17
18     //method of Second interface must also
       be implemented.
19     //Otherwise, you have to declare this
       class as abstract.
20
21     public void methodTwo()
22     {
23         //Now, method of Second interface is
        also implemented.
24         //No need to declare this class as
       abstract
25     }
26 }
```

Practice Coding Questions I

- Can you identify the error in the below code?

```
1 interface A
2 {
3     private int i;
4 }
5
6 //Illegal modifier for field i. Only public, static and final are allowed.
```

Practice Coding Questions II

- What will be the output of the following program?

```
1  interface A
2  {
3      void myMethod();
4  }
5  class B
6  {
7      public void myMethod()
8      {
9          System.out.println("My Method");
10     }
11 }
12 class C extends B implements A
13 {
14
15 }
16 class MainClass
17 {
18     public static void main(String[] args)
19     {
20         A a = new C();
21
22         a.myMethod();
23     }
24 }
25 // My Method
```

Practice Coding Questions III

- Why the below code is showing compile time error?

```
1  interface X
2  {
3      void methodX();
4  }
5
6  class Y implements X
7  {
8      void methodX()
9      {
10         System.out.println("Method X");
11     }
12 }
13 //Interface methods must be implemented as public. Because, interface methods are public by
       default and you should not reduce the visibility of any methods while overriding.
```

Practice Coding Questions IV

- Does below code compile successfully? If not, why?

```
1  interface A
2  {
3      int i = 111;
4  }
5
6  class B implements A
7  {
8      void methodB()
9      {
10         i = 222;
11     }
12 }
13 //No, because interface fields are static and final by default and you can't change their value
       once they are initialized. In the above code, methodB() is changing value of interface
       field A.i. It shows compile time error.
```

Practice Coding Questions V

- Is the following code written correctly?

```
1  class A
2  {
3      //Class A
4  }
5
6  interface B extends A
7  {
8      //Interface B extending Class A
9  }
10 //No. An interface can extend another interface not the class.
```

Practice Coding Questions VI

- What will be the output of the following program?

Practice Coding Questions VII

```
interface P
{
    String p = "PPPP";
    String methodP();
}
interface Q extends P
{
    String q = "QQQQ";
    String methodQ();
}
class R implements P, Q
{
    public String methodP()
    {
        return q+p;
    }
    public String methodQ()
    {
        return p+q;
    }
}
public class MainClass
{
    public static void main(String[] args)
    {
        R r = new R();
        System.out.println(r.methodP());
        System.out.println(r.methodQ());
    }
}
```

Output: ??????????????

Practice Coding Questions VIII

- Is the below program written correctly? If yes, what will be the output?

```
1  class A implements B
2  {
3      public int methodB(int i)
4      {
5          return i =+ i * i;
6      }
7  }
8  interface B
9  {
10     int methodB(int i);
11 }
12 public class MainClass
13 {
14     public static void main(String[] args)
15     {
16         B b = new A();
17         System.out.println(b.methodB(2));
18     }
19 }
20 // Yes, program is written correctly. Output
       will be ?????????????
```

Practice Coding Questions IX

- Can you find out the errors in the following code?

```
interface A
{
    {
        System.out.println(”Interface A”);
    }
    static
    {
        System.out.println(”Interface A”);
    }
}
//Interfaces can’t have initializers.
```

- How do you access interface field ‘i’ in the below code?

```
class P
{
    interface Q
    {
        int i = 111;
    }
}
//P.Q.i
```

Practice Coding Questions X

- Is the following program written correctly? If yes, what will be the output?

Practice Coding Questions XI

```
1  interface ABC
2  {
3      void methodOne();
4  }
5  interface PQR extends ABC
6  {
7      void methodTwo();
8  }
9  abstract class XYZ implements PQR
10 {
11     public void methodOne()
12     {
13         methodTwo();
14     }
15 }
16 class MNO extends XYZ
17 {
18     public void methodTwo()
19     {
20         methodOne();
21     }
22 }
23 public class MainClass
24 {
25     public static void main(String[] args)
26     {
27         ABC abc = new MNO();
28         abc.methodOne();
29     }
30 }
31 Output: ????????????????????????
```

Practice Coding Questions XII

- Is the below program written correctly? If yes, what will be the output?

Practice Coding Questions XIII

```
interface X
{
    void methodX();
    interface Y
    {
        void methodY();
    }
}
class Z implements X, X.Y
{
    {
        methodX();
        System.out.println(1);
    }
    public void methodX()
    {
        methodY();
        System.out.println(2);
    }
    public void methodY()
    {
        System.out.println(3);
    }
}
public class MainClass
{
    public static void main(String[] args)
    {
        Z z = new Z();
        z.methodX(); z.methodY();
        X x = z;  x.methodX();
    }
}
// Output: ??????????????????????
```

Practice Coding Questions XIV