Topics to cover:

1. Interface
2. Further programmings

1. Interface

1.1 What is Interface in Java?

An Interface in Java programming language is defined as an abstract type used to specify the behavior of a class. An interface in Java is a blueprint of a behavior. A Java interface contains static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body. It is used to achieve abstraction and multiple inheritance in Java using Interface. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also represents the IS-A relationship.

When we decide on a type of entity by its behavior and not via attribute we should define it as an interface.

Syntax:

```
interface {
   // declare constant fields
   // declare methods that abstract
   // by default.
}
```

To declare an interface, use the 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 the interface, use the implements keyword.

Example:

```
// A simple interface
interface Player
{
   final int id = 10;
   int move();
}
```

Program 1:

```
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("I am a method in Interface 1 implemented in TestClass");
        }
```

```
	public static void main(String[] args)
	{
		TestClass t = new TestClass();
		t.display();
		System.out.println(a);
	}
}
```

Output:

```
I am a method in Interface 1 implemented in TestClass
10
```

1.2 Multiple Inheritance in Java Using Interface

Multiple Inheritance is an OOPs concept that can't be implemented in Java using classes. But we can use multiple inheritance in Java using Interface. Let us check this with an example.

Program 1:

```
// Interface 1
interface API {
	// Default method
	default void show()
	{
		// Print statement
		System.out.println("Default API");
	}
}
```

```
// Interface 2
// Extending the above interface
interface Interface2 extends API {
	// Abstract method
	void display();
}

// Interface 3
// Extending the above interface
interface Interface3 extends API {
	// Abstract method
	void print();
}

// Main class
// Implementation class code
class TestClass implements Interface2, Interface3 {
	// Overriding the abstract method from Interface2
	public void display()
	{
		System.out.println("Display from Interface2");
	}
	// Overriding the abstract method from Interface3
	public void print()
	{
		System.out.println("Print from Interface3");
	}
	// Main driver method
	public static void main(String args[])
	{
		// Creating object of this class
		// in main() method
		TestClass d = new TestClass();

		// Now calling the methods from both the interfaces
		d.show(); // Default method from API
		d.display(); // Overridden method from Interface2
		d.print(); // Overridden method from Interface3
	}
```

```
}
```

Output:

```
Default API
Display from Interface2
Print from Interface3
```

Now we will convert the above program into a new program that takes a directory and reads all files in that directory.

Program 2:

```
import java.io.*;
import java.util.*;


// Interface 1
interface API {
    default void show()
    {
            System.out.println("Default API: I am just an interface with show method\n");
    }
}

// Interface 2
// Extending the above interface
interface Interface2 extends API {
    // Abstract method
    int countFiles(String s);
}

// Interface 3
// Extending the above interface
```

```
interface Interface3 extends API {
    // Abstract method
    void showContents(String s);
}


class TestClass implements Interface2, Interface3 {
    // Overriding the abstract method from Interface2
    public int countFiles(String fileDir)
    {
            int count=0;
            File folder = new File(fileDir);
                    File[] listOfFiles = folder.listFiles();
                    for (File file : listOfFiles) {
                            if (file.isFile()){
                                    showContents(file.getAbsolutePath());
                                    count++;
                            }
                    }
            return count;
    }

    // Overriding the abstract method from Interface3
    public void showContents(String fileWithPath)
    {
            File file = new File(fileWithPath);
                    String content="";
                    try{
                            Scanner sc = new Scanner(file);
                            while (sc.hasNextLine()){
                                    String s = sc.nextLine();
                                    content=content+s+" ";
                            }
                                     System.out.println("File:            "+fileWithPath+"\nContent:
"+content.trim()+"\n");
                            sc.close();
                    }
                    catch(Exception e){
                            System.out.println(e.getMessage());
                    }
```

```
    }

    public static void main(String args[])
    {
        TestClass d = new TestClass();
        String myDir = "/home/vboxuser/AJP/data";
        d.show(); // Default method from API
        int count=d.countFiles(myDir); // Overridden method from Interface2

        System.out.println("\nTotal number of files in this directory: "+count);
    }
}
```