Week 3

08 July 2023 11:00 PM

8th July 2023

Abstraction- Using only essential knowledge without in detail knowledge
- Essential knowledge- Name of method, its parameters, return type
- In detail knowledge- Code in method's body
- Abstraction can be achieved through- i) interface ii) abstract class

API- rules and guidelines(like name of method, its parameters, return type)
3 type of API in java-
1. Inbuilt API- rules and guidelines- by java community, implementation- by java community
   - e.g. Multithreading, Collection framework
2. Open specification API- rules and guidelines- by java community, implementation- by different vendors
   - e.g. JDBC API (written by vendors like oracle, mysql), Servlet API, JSP API
3. Third party API- rules and guidelines- by different vendors, implementation- by different vendors
   - e.g. Spring, Hibernate

interface is a set of rules and guidelines.

Rules of interface-
1. interface keyword
2. In interface, all variables are public, static and final by default
3. interface variables must be initialized (as they're final)
4. **Till JDK 1.7, all interface methods are abstract**
5. We cannot create object of interface.
6. **We can create reference of interface by using its implemented class.**

```
public interface Inter{
    //error- int x;
    int x= 10;
    void m1();
    void m2();
}

public class T implements Inter{
    public void m1(){
        System.out.println("m1--T");
    }
    public void m2(){
        System.out.println("m2--T");
    }

}
```

```
public class Test{
    public static void main(String[] args){
        Inter i= new T();
        i.m1();
        i.m2();
        T t1= new T();
        System.out.println(Inter.x); //preferred
        System.out.println(i.x);
        System.out.println(t1.x);
    }
}
```

```
interface Connection{
    void commit();
    void rollback();
}

class Mysql implements Connection{
```

```
    void rollback();
}

class Oracle implements Connection{
    public void commit(){
        System.out.println("Commit-Oracle");
    }
    public void rollback(){
        System.out.println("Rollback-Oracle");
    }
}

class Mysql implements Connection{
    public void commit(){
        System.out.println("Commit-Mysql");
    }
    public void rollback(){
        System.out.println("Rollback-Mysql");
    }
}

public class Test{
    public static void main(String[] args){
        Connection c1= new Oracle();
        c1.commit();
        c1.rollback();
    }
}
```

Instead of writing separate classes for the databases and using their method names, the methods are declared in interface and are implemented in classes which implement that interface.
If we were to use Mysql database instead of oracle in the above code, we'll have to change only-
Connection c1= **new Mysql();**
But if this change needs to be done many times is not efficient. This is overcome by Spring IOC.

interface extends interface
inteface extends multiple interfaces
interface I2 extends I, I1

interface supports multiple inheritance because it has abstract methods.
Question- **Why is multiple inheritance not supported by Java classes?**
Answer: If 2 classes have same method and if a class would be extending both classes, then while calling the method from child class, there's confusion about which class's method to call in Java.

9th July 2023

Notes given by sir--

File Handling (I/O) :
==========

=> To store large amount of data we can use dataabse Concept.
=> There are various types of databases are avialbale in market.
    a) Relational databses .(MYSQL , Oracle ,PostGress ...)
    b) Non-Relational Databse. (MongoDB).
=> So connect our java Application to the database.We have to follow some steps:-

1) load third party jar files to Project.(MYSQL,Oracle)
2) provide user name and password.
3) provide the public url of the database Instance.
5) perfrom the CRUD opeartion.(using Query)

File IO :
=========

I/O = Input /Output

=> If we want to store small amount of data then we can use File Io concept.
=> To save the data or retrive the data from the file,java provides various
classes,and methods.
=> All the classes and method are avaiable in java.io package.
=> If we want to use this classes then we have to first import java.io package.
=> Various classes related to File Io Concept

1) File (C)
2) FileWriter(c)
3) BufferedWriter(c)
4) PrintWriter(c)
5) FileReader(c)
6) BufferedReader(c)

==========

(Here, for the changes to get reflected in java Project- Project in Eclipse-> Right click-> Refresh)
classes and methods to retrieve data from file- java.io package

1) How to create File In java?

=>

```
File f =new File("ABC.txt");
f.createNewFile();
System.out.println("File created  ....!");
```

---

2) How to create Folder In java?

=>

```
File f=new File("Core Java");
f.mkdir();
System.out.println("Done ...");
```

---

3) How to create File Inside the Folder?

=>

```
File f=new File("Core Java", "Roll.java");
f.createNewFile();
System.out.println("Done...!");
```

---

4) How to create Folder in the Drive?

=>

```
File f=new File("E:\\","May Batch");
f.mkdir();
System.out.println("done ...!");
```

---

5) How to create File in the Drive?

=>

```
File f=new File("E:\\","Rollno.pdf");
f.createNewFile();
System.out.println("Ok...");
```

---

6) How to create File Inside the Folder in Drive?

=>

```
File f=new File("E:\\May Batch","Student.java");
f.createNewFile();
System.out.println("Ok...");
```

---

7) **File Class Important methods? **:

==========

```
        File f = new File("PQR.txt");

        System.out.println(f.createNewFile()); // false
        System.out.println(f.isFile()); // true
        System.out.println(f.isDirectory()); // false
        System.out.println(f.canRead());// true
        System.out.println(f.getAbsolutePath());//
        System.out.println(f.exists()); // true
        System.out.println(f.delete()); //true
        System.out.println(f.exists()); // false

        f.renameTo(new File("ABCD.txt"));
```

==========

8) How to Write the Data inside the File?

=>

=>To Write the Data inside the File java provides various classes.

a) FileWriter(c)

b) BufferedWriter(C)

c) PrintWriter(c)

9)How to Write data into the File Using FileWriter?

=>

```
FileWriter fw=new FileWriter("XYZ.txt");

fw.write("Python");
fw.write("\n");
fw.write("Java");
fw.write("\n");
fw.write('A');
fw.write("\n");
fw.write(97);

fw.flush();
System.out.println("Done ...!");
```

---

10) How to append the data into the File ?

=>

```
FileWriter fw=new FileWriter("XYZ.txt", true);  // new FileWriter(String filename, boolean append)

fw.write("\n");
fw.write("Automation Testing");
fw.write("\n");  //to go to next line
fw.write("Nayan");
fw.write("\n");
fw.write(859655);

fw.flush();
System.out.println("done...!");
```

---

11) How to Write the Data in File Using BufferedWriter ?

=>

```
        newline();  => to Shift cursor to next Line.
```

```
        File f =new File("XYZ.txt");

        FileWriter fw=new FileWriter(f);

        BufferedWriter bw=new BufferedWriter(fw);

        bw.write("Nayan");
        bw.newLine();
        bw.write("Ankit");
        bw.newLine();
        bw.write("Unnati");
        bw.newLine();
        bw.write("Sandesh");

        bw.flush();

        System.out.println("done");
-------------------------------------------------------------------
12) How to append the Data in File Using BufferedWriter?
=>
        FileWriter fw=new FileWriter("XYZ.txt", true);
        BufferedWriter bw=new BufferedWriter(fw);

        bw.write("Swarali");
        bw.newLine();
        bw.write("Fatima");
        bw.newLine();
        bw.write("Raj");
        bw.newLine();
        bw.write("Sonali");
        bw.newLine();
        bw.write("Saurabh");

        bw.flush();
        System.out.println("done");
=====================================================
13) PrintWriter: It is smart write in all Writer.
================
            How to write the Data inside File using PrintWriter ?
            =>
            PrintWriter pw = new PrintWriter("XYZ.txt");

            pw.write("Java");
            pw.write("\n");
            pw.write('A');
            pw.write("\n");
            pw.write(100);
            pw.write("\n");

            pw.println(false);
            pw.println('B');
            pw.println(12.256d);
            pw.println(100);
            pw.print(true);
```

```
            pw.flush();

            System.out.println("done ...!");
```
==========

14) How to read the Data From File?
=> To read the Data from File Java provides Various classes .
  a) FileReader(C)
  b)BufferedReader(c)
15) How to read the data From File Using FileReader?
=>
  Note : read() method will retrun -1 value , no data present inside File.

```
            File f = new File("XYZ.txt");

            FileReader fr = new FileReader(f);

            int i = fr.read();

            while (i != -1) {
                  char ch = (char) i;
                  System.out.print(ch);
                  i = fr.read();
            }
```

---
16) How to read the Data From File using Bufferedreader?
=>
  Note : readLine() method will return null value if no data present inside File.

```
            FileReader fr=new FileReader("XYZ.txt");

            BufferedReader br=new BufferedReader(fr);

            String val=br.readLine();

            while(val!=null)
            {
                  System.out.println(val);
                  val =br.readLine();
            }
```
==========End==========
My notes-
In above examples, data won't be written without flush().
FileWriter fw= new FileWriter("XYZ.txt");
fw.write(100); //In the file, this will be seen as a character (100 is the ascii value of d)

File f= new File("PQR.text");
boolean b= f.createNewFile();
Return type of createNewFile() is boolean. It'll return false if the file is already available.

BufferedWrite-
newLine() - to shift the cursor to next line

The println(String) method of PrintWriter Class in Java is used to print the specified String on the stream and then break the line. This String is taken as a parameter.

read() - reads a single character from the reader

read(char[] array) - reads the characters from the reader and stores in the specified array

You can use the readLine() method from java.io.BufferedReader to read a file line-by-line to String. This method returns null when the end of the file is reached.