File Operations
Serialization and DeSerialization
JDBC
Servlet and JSP
Thymleaf

Hibernate
Spring
 a. SpringCore
 b. SpringDataJPA
 c. SpringMVC,SpringMail
 d. SpringSecurity
 e. SpringRest
 f. Microservices using SpringRest

DesignPatterns
 Structural
 Behavioural
 JEE Design Pattern

File Operations
+++++++++++++++
Agenda:
1. File
2. FileWriter
3. FileReader
4. BufferedWriter
5. BufferedReader

File:
  File f=new File("abc.txt");
 This line 1st checks whether abc.txt file is already available (or) not if it is already  available then "f" simply refers that file.
 If it is not already available then it won't create any physical file just creates a java File object represents name of the file.

Example:
```
import java.io.*;
class FileDemo{
      public static void main(String[] args)throws IOException{

            File f=new File("cricket.txt");
            System.out.println(f.exists());//false

            f.createNewFile();
            System.out.println(f.exists());//true
      }
}
```
1st run
=======
false
true

2nd run
======
true
true

=> A java File object can represent a directory also.

Example:

```
import java.io.File;
import java.io.IOException;

class FileDemo{
       public static void main(String[] args)throws IOException{

            File f=new File("cricket123");
            System.out.println(f.exists());//false

            f.mkdir();//Creates a new directory
            System.out.println(f.exists());//true
      }
}
```

1st run

```
=======
false
true
```

2nd run

```
======
true
true
```

Note: In UNIX everything is a file, java "file IO" is based on UNIX operating system
hence in java also we can represent both files and directories by File object only.

File class constructors

```
==================
```

1. File f=new File(String name);
   => Creates a java File object that represents name of the file or directory in current working directory.

```
            eg#1. File f=new File("abc.txt");
```

2. File f=new File(String subdirname,String name);
   => Creates a File object that represents name of the file or directory present in specified sub directory.

```
            eg#1. File f1=new File("abc");
                  f1.mkdir();
                  File f2=new File("abc","demo.txt");
```

3. File f=new File(File subdir,String name);

```
            eg#1.File f1=new File("abc");
                    f1.mkdir();
                 File f2=new File(f1,"demo.txt");
```

Requirement
=> Write code to create a file named with demo.txt in current working directory.

```
            cwd
             |=> demo.txt
```

eg#1.

```
import java.io.*;

//Client Code
```

```
public class Test
{
   public static void main(String[] args)throws IOException
      {
         //File(UNIX) :: file,folders
            File f= new File("demo.txt");
            f.createNewFile();
            System.out.println(f.exists());

      }
}
```

Requirement
=> Write code to create a directory named with IPLTeam in current working directory and create a  file named with abc.txt in that directory.

```
            cwd
              |=> IPLTeam
                   |=> abc.txt
```

eg#1.

```
import java.io.*;

//Client Code
public class Test
{
   public static void main(String[] args)throws IOException
      {
              //File(UNIX) :: file,folders
            File f1= new File("IPLTeam");
            f1.mkdir();

            File f2 = new File(f1,"rcb.txt");
            f2.createNewFile();

            System.out.println(f1.exists());
            System.out.println(f2.exists());

      }
}
```

Requirement:
=> Write code to create a file named with rcb.txt present in D:\IPLTeam folder.

```
                D
                 |=> IplTeam
                      |-> rcb.txt
```

eg#1.

```
import java.io.*;

//Client Code
public class Test
{
   public static void main(String[] args)throws IOException
      {
              //File(UNIX) :: file,folders
            File f= new File("D:\\OctBatchMicroservices\\IPLTeam","rcb.txt");
            f.createNewFile();
            System.out.println(f.exists());
      }
}
```

Note: If the specified String representation path doesn't exists, then JVM will throw an Exception called "FileNotFoundException"

Important methods of file class:
++++++++++++++++++++++++++++++

1. boolean exists();
     Returns true if the physical file or directory available.

2. boolean createNewFile();
      This method 1st checks whether the physical file is already available or not, if it is already available then this method simply returns
      false without creating any physical file.
      If this file is not already available then it will create a new file and returns true

3. boolean mkdir();
      This method 1st checks whether the directory is already available or not if it is already available then this method simply returns
      false without creating any directory.
      If this directory is not already available then it will create a new directory and returns true

4. boolean isFile();
      Returns true if the File object represents a physical file.

5. boolean isDirectory();
      Returns true if the File object represents a directory.

6. String[] list();
      It returns the names of all files and subdirectories present in the specified directory.

7. long length();
      Returns the no of characters present in the file.

8. boolean delete();
      To delete a file or directory.
        true -> if directory is deleted otherwise false if directory is deleted.

Requirement: Write a program to display the names of all files and directories present in D:\EnterpriseJava

eg#1.
```
import java.io.*;

//Client Code
public class Test
{
   public static void main(String[] args)throws IOException
      {
          //File Object of java
            File f= new File("D:\\EnterpriseJava");

            //Get the information of that location
            String[] info = f.list();
```

```
            int count = 0;

            //read the array and display the information
            for (String data:info )
            {
                  count++;
                  System.out.println(data);
            }
            System.out.println("Total no of files and directories is :: "+count);
            System.out.println("Total no of files and directories is ::
"+info.length);
      }
}
Output
Total no of files and directories is :: 153


Requirement: Write a program to display only file names.
Requirement: Write a program to display only directory names.

eg#1
import java.io.*;

//Client Code
public class Test
{
   public static void main(String[] args)throws IOException
      {
          //File Object of java
            File f= new File("D:\\EnterpriseJava");

            //Get the information of that location
            String[] info = f.list();

            //Count of Files
            int countOfFiles = 0;
            int countOfDirectories = 0;

            //read the array and display the information
            for (String data:info )
            {
                  //Creating a File Object for "D:\EnterpriseJava"
                  File f1 = new File(f,data);

                  //Checking whether File Object is "FileType"
                  if (f1.isFile()){
                        countOfFiles++;
                        System.out.println(data);
                  }

                  //Checking whether File Object is "DirectoryType"
                  if(f1.isDirectory())
                  {
                        countOfDirectories++;
                        System.out.println(data);
                  }
            }
            System.out.println("Total no of files is       :: "+countOfFiles);
```

```
            System.out.println("Total no of Directories is ::
"+countOfDirectories);
      }
}
Total no of files is       :: 145
Total no of directories is :: 8
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++
```

FileWriter:
   By using FileWriter object we can write character data to the file.

Constructors:

```
      FileWriter fw=new FileWriter(String name);
      FileWriter fw=new FileWriter(File f);
```

    The above 2 constructors meant for overriding the data to the file.

Instead of overriding if we want append operation then we should go for the following 2 constructors.

```
      FileWriter  fw=new FileWriter(String name,boolean append);
      FileWriter  fw=new FileWriter(File f,boolean append);
```

If the specified physical file is not already available then these constructors will create that file.

Methods:

1. write(int ch);
      To write a single character to the file.
2. write(char[] ch);
      To write an array of characters to the file.
3. write(String s);
      To write a String to the file.
4. flush();
      To give the guarantee the total data include last character also written to the file.
5. close();
      To close the stream.

eg#1.

```
import java.io.*;

public class Test
{
   public static void main(String[] args)throws IOException
      {
         //Writing character data to a file
         FileWriter fw = new FileWriter("cricketer.txt",true);
         fw.write(97);
         fw.write("PWIOI");

         char[] arr = {'P','W','S','K','I','L','L','S'};
         fw.write(arr);

         fw.flush();//Gaurantee that data is written to a file
         fw.close();//Closing the Stream
      }
}
```

Note:
 => The main problem with FileWriter is we have to insert line separator manually,which is  difficult to the programmer. ('\n')
 => And even line separator varing from system to system.

FileReader:
 => By using FileReader object we can read character data from the file.

Constructors:
  FileReader fr=new FileReader(String name);
  FileReader fr=new FileReader (File f);

Methods
=======
1. int read();
      It attempts to read next character from the file and return its Unicode value. If
      the next character is not available then we will get -1.

2. int i=fr.read();

3. System.out.println((char)i);
      As this method returns unicodevalue , while printing we have to perform type casting.

4. int read(char[] ch);
      It attempts to read enough characters from the file into char[] array and returns
      the no of characters copied from the file into char[] array.

5. File f=new File("abc.txt");

6. Char[] ch=new Char[(int)f.length()];

7. void close();

eg#1.
```
import java.io.*;

//Client Code
public class Test
{
   public static void main(String[] args)throws IOException
      {
         //Reader character data from a file
         File file = new File("sample.txt");

         FileReader fr = new FileReader(file);

         //FirstAppraoch :: reading character data at a time
          int i =fr.read();
            while (i!=-1)
            {
                  System.out.print((char)i);
                  i = fr.read();
```

```
            }

         System.out.println();


         //SecondApproach :: reading whole file data into character array
         char[] ch = new char[(int)file.length()];
         System.out.println("No of characters in a file :: "+file.length());
         int noOfCharactersRead = fr.read(ch);
         System.out.println("OneTimeReadOperation count :: "+noOfCharactersRead);

         for (char data: ch)
         {
               System.out.print(data);
         }
         System.out.println();
         fr.close();//Closing the Stream
      }
}
```

Usage of FileWriter and FileReader is not recommended because of following reason

1. While writing data by FileWriter compulsory we should insert line separator(\n) manually which is a bigger headache to the
programmer.

2. While reading data by FileReader we have to read character by character instead of line by line which is not convenient to the
programmer.

Assume we need to search for a 10 digit mobile no present in a file called "mobile.txt"
=>Since we can read only character just to search one mobile no 10 searching and to search 10,000 mobile no we need to read 1cr times,
so performance is very low.

3. To overcome these limitations we should go for BufferedWriter and BufferedReader concepts.