What is persistency?
  It is a mechanism of storing the data permanently on to the file.
  In java persistency can be achieved through an API's available inside package called "java.io".

Input and Output 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("IPLTeams");
            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
             |=> abc.txt

Program:
import java.io.*;
class FileDemo{
      public static void main(String[] args)throws IOException{
            File f=new File("demo.txt");
            f.createNewFile();
      }
}
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

Program:
import java.io.*;
class FileDemo{
```

```
  public static void main(String[] args)throws IOException{
      File f1=new File("IPLTeam");
      f1.mkdir();
      File f2=new File("IPLTeam","abc.txt");
      f2.createNewFile();
  }
}
```

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

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

Program:

```
import java.io.*;
class FileDemo{
      public static void main(String[] args)throws IOException{
            File f=new File("c:\\IPLTeam","rcb.txt");
            f.createNewFile();
      }
}
```

Assuming C:\\IPLTeam should be already available otherwise it would result in "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

Requirement:
Requirement: Write a program to display the names of all files and directories present in D:\\Java Job Guarantee Batch
Requirement: Write a program to display only file names.
Requirement: Write a program to display only directory names.

```
import java.io.*;
class Test
{
      public static void main(String[] args)throws Exception
      {
            int dirCount     = 0;
            int jpgFileCount = 0;
            int txtFileCount = 0;
            int zipFileCount = 0;

            String location = "D:\\Java Job Guarantee Batch";
            File f= new File(location);

            String[] names = f.list();

            for(String name : names){
                  // D:\\Java Job Guarantee Batch
                  // all files we are iterating
                  File f1 = new File(f,name);

                  if (f1.isDirectory())
                        dirCount++;

                  if(f1.isFile()){

                        if (name.endsWith(".png"))
                              jpgFileCount++;
                        if(name.endsWith(".txt"))
                              txtFileCount++;
                        if(name.endsWith(".zip"))
                              zipFileCount++;

                  }
                  System.out.println(name);
            }
            System.out.println("Total no of JPGfiles  is :: "+jpgFileCount);
            System.out.println("Total no of txtfiles  is :: "+txtFileCount);
            System.out.println("Total no of Zipfiles  is :: "+zipFileCount);
            System.out.println("Total no of Directory is :: "+dirCount);

      }
      //JVM shutdown now
}
```

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.FileWriter;
import java.io.IOException;

public class TestApp {
	public static void main(String[] args)throws IOException {
			FileWriter fw=new FileWriter("abc.txt");
			fw.write(73);
			fw.write("neuron\nTechnology\nPrivate\nLimited");
			fw.write("\n");
			char ch[] ={'a','b','c'};
			fw.write(ch);
			fw.flush();
			fw.close();
	}
}
```

A new file will be created automatically

abc.txt
=======
Ineuron
Technology
Private
Limited
abc

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.FileReader;
import java.io.IOException;

public class TestApp {
      public static void main(String[] args)throws IOException {
                  FileReader fr=new FileReader("abc.txt");
                  int i=fr.read();
                  while(i!=-1){
                        System.out.println((char)i);
                        i=fr.read();
                  }
      }
```