FILES [java.io]

- **Non persistency programs**
  - **Non Persistency programs one which enable you to store the data temporarily during the program execution**
  - **The data which we are passing during executing the program as input and the values which we getting after executing program as an out those values will be stored in the system temporarily**

- **Persistency Programs**
  - **Persistency program one which enable you to store the data in a system memory permanently.**
    1. **Persistency programs can be done in 2 ways**
       - **Using Files**
       - **Using Databases**

- **What is File ?**
  - **Files are used to store the data permanently in the system memory.**
  - **Files are good to store limited data**
  - **Files doesn't provide any sec for confidential data through username and password**
  - **To overcome this problem then databases are used**
    1. **Oracle | sqlServer | MySql | DB2……**

- **In J2SE we will be learning about how to interact with files concepts**
- **In order to interact with files Java is provided some classes ,methods and interface in package called "java.io"**
- **java.io stream [Flow of Data] classes are classified into 2 types**

- **ByteStream**
  1. **In Byte Streams data will be maintain and manipulated in the form of bytes**
  2. **Byte Stream are used to read and write data which is 1 byte equitant in a single operation**
  3. **Byte Stream will support only "ASCII" Data**
  4. **Another name of Byte Stream "8 bit Stream"**
  5. **Really good for processing images , audio files…**

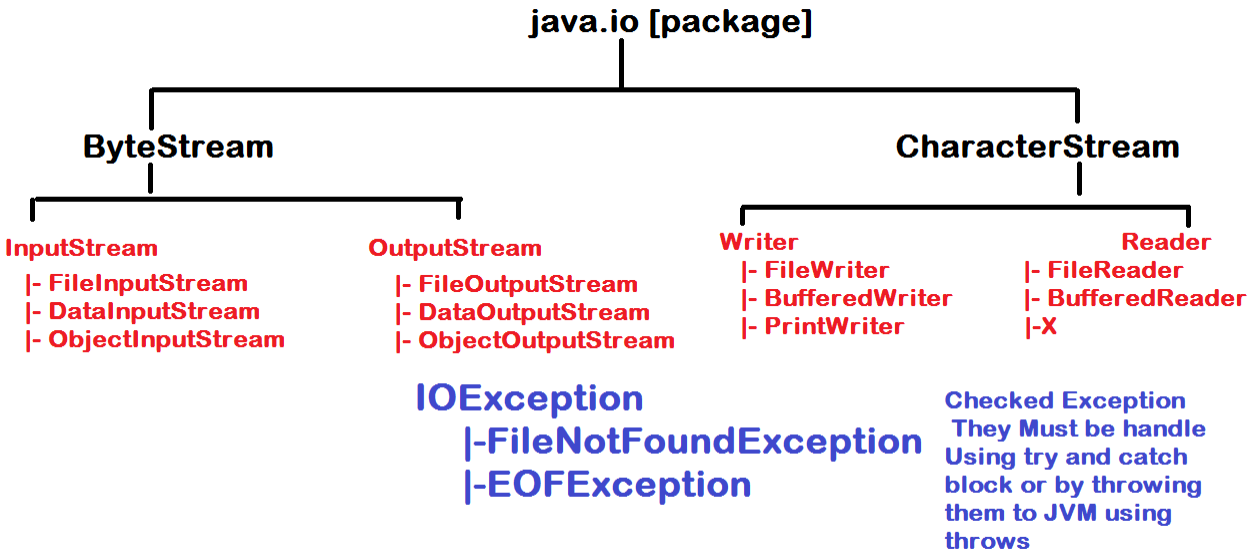
- **Character Stream**
  1. **In Character stream data will be maintain and manipulated in the form of "characters"**
  2. **Character Stream will allow to read and write data which equal to 2 bytes in single operation**
  3. **Character Streams will support "Unicode " character SET [Unicode = ASCII and Other Language characters]**
  4. **Another name for character Streams "16 bits stream"**
  5. **Character Stream good for normal text data**

Byte Stream are classified into 2 types

- **Output Stream**
  1. **Output Streams meant for writing data**
  2. **Output streams required "Target" as Object**
- **Input**
- **InputStreams**
  - **InputStream are meant for "Reading" Data**
  - **InputStream required "Source" as an Object**

CharacterStream are classified into 2 types:

- **Writer Classes**
  - **Writer classes are meant for writing the Data**
  - **Which required "Target" as an object**
- **Reader Classes**
  - **Reader classes are for reading the data**
  - **Reader classes required "source" as an Object**



ByteStream

- **OutputStream**
  - **OutputStreams are meant for writing the data , which required target as an "object"**
  - **Methods of OutputStreams**
    1. **public void write(int):**
       - **it will write 1 byte of the data into the file**
    2. **public void close( ):**

- it will close the outputstream object and releases any system resources associated with outputstream

➢ InputStream

- InputStreams are meant for reading the data , which required source as an "Object"

Methods of InputStream:

- public int read( ):
  1. It will read 1 byte of data from the input stream and will return its ascii value.if the file is not having any data to read then it will return -1
- public void close():
  1. It will close the input stream object and releases any system resources associated with inputStream

FileOutputStream:

➢ FileOutputStream class which enable you to store the into the file

- Constructors:
  1. FileOutputStream(String) : String rep name of the file. Actually the file will be created in the same directory

FileOutputStream fos= new FileOutputStream("Sample");

Example : 1

```
import java.io.*;

class FileOutputStreamDemo
{
        public static void main(String args[ ]) throws IOException
        {
                FileOutputStream fos=new FileOutputStream("Sample");
```

```
                fos.write(65);
                        fos.write(66);
                        fos.write(67);
                        fos.close();
                        System.out.println("File is Saved ");
        }
}
```

InputStream

- **public int available();**
  - **return No.of.Bytes are existed in the file to read**

FileInputStream:

- **FileInputStream allows an application to read the data from the file**

Example 1:

//FileInputStreamDemo.java

```
import java.io.*;

class FileInputStreamDemo
{
        public static void main(String args[ ])
        {
         try{
                FileInputStream fis=new FileInputStream("Sample");
                int nb=fis.available();
                System.out.println("No.of.Byte to read : "+nb);

                int x;
                for(int i=1; i<=nb; i++)
                    {  x=fis.read();
                       System.out.print((char)x);
                        }

         fis.close();
         }
         catch(IOException ioe)
                {
                 System.out.println("Sorry File Not Existed ....");
                }
        }
```

```
}
```

Example 2:

```
//FileInputStreamDemo2.java
import java.io.*;

class FileInputStreamDemo2
{
        public static void main(String args[ ])
        {
                try{
                        FileInputStream fis=new FileInputStream("Sample");
                        int x=fis.read();  // 65

                        while(x!=-1)
                        {  System.out.print((char)x);
                          x=fis.read();  }

                        fis.close();
                }
                catch(IOException ioe){
        System.out.println("Sorry File Not Found ");
                }
        }
}
```

Example 3: For reading the Data From the File

```
//FileInputStreamDemo3.java
import java.io.*;
import java.util.Scanner;

class FileInputStreamDemo3
{
        public static void main(String args[])
        {
                FileInputStream fis=null;  //local variable
      Scanner s=new Scanner(System.in);
                System.out.println("Enter Filename ");
                String fname=s.next();

                try{
                    fis=new FileInputStream(fname);
```

```
                int x=fis.read();
                while(x!=-1)
                  {   System.out.print((char)x);
                      x=fis.read();
                  }
            }//try

            catch(IOException ioe){
                System.out.println("Sorry File Not Found ");
            }//catch

            finally
            {
                try{

                 if(fis!=null)
                        {
                         fis.close();
                         System.out.println("\n File is Closed ");
                        }
                }
                catch(IOException ioe)
                {  System.out.println("sorry unable to continue ..."); }
            }//finally

        }//main
}//class
```

Example 4 : Reading the Data From the File Using

```
    //FileInputStreamDemo4.java
//Java 8 feature try with resource

import java.io.*;

class FileInputStreamDemo4
{
        public static void main(String args[])
        {
                java.util.Scanner s=new java.util.Scanner(System.in);
                System.out.println("Enter Filename ");
                String fn=s.next();
```

```
            try(FileInputStream fis=new FileInputStream(fn) ) //Java8
             {
                    int x=fis.read();
                    while(x!=-1)
                      { System.out.print((char)x);
                        x=fis.read();
                            }
             }
            catch(IOException ioe)
            { System.out.println("Sorry File not found "); }
        }
}               try with resource From Java 8
```

java.io.File

- By Using File class we can create File,create a directory,remove a file,list all the file….

Constructors

- File(String)
- File(String subdir,String fname)
- File(File subdir,String fname)

Methods

- public long length( ) : it will return no.of.bytes are existed in the file
- public void createNewFile( ): it will create a new empty file
- public void mkdir( ): it will create a directory
- public String[ ] list( ) : it will read all files and directory existed in the specified path
- **public boolean exists():** it will return "true" if the specified file|directory is existed @ the specified location [default location is current directory]
- public boolean isFile( ): it will returns "true" if the given string is "File"

- **public boolean isDirectory(): it will returns "true" if the given string is "directory"**

```
//FileEx1.java

import java.io.*;
class FileEx1
{
        public static void main(String args[ ])
        {
                File f=new File("Sample");
                if( f.exists( ) )
                {
                        System.out.println("Existed ");
                }
                else
                {
                        System.out.println("Not Existed ");
                }
        }
}
```

Example 2 To Ensure the specified String is a File or Not :

```
//To Ensure the specified String is a file or not
//FileEx2.java

import java.io.*;

class FileEx2{
        public static void main(String args[ ])
        {
                File f=new File("Sample");
                if( f.exists( ) )
                 {
                        System.out.println("Yes it is Existed ");
```

```
                if( f.isFile( ) )
                        {
                                System.out.println("Yes it is a File ");
                        }
                        else if( f.isDirectory() )
                        {
                                System.out.println("Yes it is a directory ");
                        }
                    }
                else
                {
                        System.out.println("No It is not Existed ");
                }
        }
}
```

Example 3: Creating an empty file in the CWD

//To create an empty file when the specified File is not existed

//FileEx3.java

```
import java.io.*;
class FileEx3
{
  public static void main(String args[ ])  throws IOException
        {
          File f=new File("Sssit_Data");
          if( f.exists( ) )
                { System.out.println("Yes it is Existed "); }
          else
                { f.createNewFile();
            System.out.println("File is Created ....");
              }

  }
}
```

```
//Create a directory in e:\java9
//File(String subdir,String fname)

//FileEx4.java
import java.io.*;

class FileEx4
{
        public static void main(String args[]) throws IOException
        {
                File f=new File("e:\\java9","shashi");
                f.mkdir();
                System.out.println("Directory is Created ");
        }
}

//Display all item existed  in the specified directory
//e:\adv_shiva\servlets        //public String[ ] list( )

//FileEx5.java

import java.io.*;
class FileEx5
{
   public static void main(String args[ ])
        {  //File(String);
                File f=new File("e:\\adv_shiva");
                        // f--> e:\\adv_shiva

                //File(File subdir,String name);
                File f2=new File(f,"servlets");
                        // f2---> e:\\adv_shiva\\servlets

                String names[]=f2.list();
                for(String name:names)
```

```
                { System.out.println(name);}

                System.out.print("No.of.Items : "+names.length);
        }
}

//FileWriterDemo.java
//FileWriter(String)  --> creating the file in the same directory
//FileWriter(File);  --> creating the file in sepcified location




import java.io.*;
class FileWriterDemo
{
        public static void main(String args[ ]) throws IOException
        {
      FileWriter fw=new FileWriter("data1");
                fw.write(65);   //write(int)
                fw.write("\n");

                char x[]={'w','e','l','c','o','m','e'};
                fw.write(x);   //write(char[ ])
                fw.write("\n");

                String s="Have a nice day";
                fw.write(s);  //write(String);
                fw.write("\n");

                fw.close();
                System.out.println("Data is Saved ");
        }
}
```

java.io.FileReader

- this class allow an application to read the data from the file

Constructor

1.FileReader(String)

it will read data from the file which is existed CWD

2.FileReader(File) --> it will read data from the specified location

```
import java.io.*;

class FileReaderDemo {
        public static void main(String args[ ]) throws IOException
       {
                File f=new File("e:\\j2se_super\\Applets\\Myapplet.html");
                FileReader fr=new FileReader(f);   //int read( )
                int x=fr.read( );

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

          fr.close();
        }
}
```

BufferedWriter:

- **while working with Filewriter we have to put "\n" new line character after each data.**
- **The new line character "\n" may or may not be reflected in the different file environments**
- **To Overcome the above problem then we have to use BufferedWriter**
- **BufferedWriter doesn't allow to write the data into the file directly which required any "Writer" class Object**

Constructor:

BufferedWriter(Writer):

//BufferedWriterDemo.java

```
import java.io.*;

 class BufferedWriterDemo {
        public static void main(String args[ ]) throws IOException
       {
              FileWriter fw=new FileWriter("data2");
              BufferedWriter bw=new BufferedWriter(fw);

              bw.write(65); //write(int)
              bw.newLine();

              char[ ] x={'w','e','l','c','o','m','e'};
              bw.write(x);  //write(char[ ])
              bw.newLine( );

              String s="Have a Good Day....!!!";
              bw.write(s);  //write(String)
              bw.close();
              fw.close();
              System.out.println("Data is Saved ");
       }
 }
```

java.io.BufferedReader :

FileReader for reading data from the file char by char
public int read( ) return -1 when file is not having data

BufferedReader for Reading data From the file line by line
BufferedReader(Reader) FileReader | BufferedReader
public String readLine();
- **return null when file not having data to read**

```
//BufferedReaderDemo.java
import java.io.*;

class BufferedReaderDemo
```

```
{
        public static void main(String args[]) throws Exception
        {
                File f=new File("e:\\j2se_shiva\\class\\Employee.java");
                FileReader fr=new FileReader(f);
                BufferedReader br=new BufferedReader(fr);

      String line=br.readLine();

                while(line!=null)
                { System.out.println(line);
                  line=br.readLine();
                  Thread.sleep(1000);  //multi-threading
                }

                br.close();
                fr.close();

        }
}
```

PrintWriter

- **While working with print writer need not type "\n" character or newLine( ) to bring the cursor to the next line in the file**
- **PrintWriter allows to write any type of data. Into the file**

Constructor:

PrintWriter(String) -->Create a file In the CWD
PrintWriter(File) --> Create a File in the specified location

```
  //public void println(xxx) int | short | byte | float | double
  //public void print(xxx)

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

```
        {
                PrintWriter pw=new PrintWriter("data3");
                pw.println(10);
                pw.println('s');
                pw.println(3.14f);
                pw.println("Shashi Java ");
                pw.close();
                System.out.println("Data is Saved ");
        }
}
```

DynamicData

```
//DynamicData.java
import java.io.*;

class DynamicData
{
        public static void main(String args[ ]) throws IOException
        {
//To read the Data From the user We have to use "DataInputStream"
        DataInputStream dis=new DataInputStream(System.in);
        FileOutputStream fos=new FileOutputStream("info.txt");
      System.out.println("Enter Data Press * for Exit ");
           int x=dis.read( );

                while( x!='*' )
                { fos.write(x);
                   x=dis.read( );
                 }

                System.out.println("Data is Saved ");
                dis.close();
                fos.close();
        }
}
```

Serialization:

- It is the process of storing the state of object in the file
- In order to achieve "Serialization" then we have to make use of "ObjectOutputStream"
- To Create an Object for "ObjectOutputStream" then we have to use the following constructor "ObjectOutputStream(OutputStream)"

```
Eg:  FileOutputStream fos=new
                        FileOutputStream("MyEino");
     ObjectOutputStream oos=new ObjectOutputStream(fos);
```

- To write the Object into the file then we have to the following method

  java.io.ObjectOutputStream
  public final void writeObject(Object) throws IOException

  Note: If you want write any class object into the file then corresponding class must be implemented by "java.io.Serializable" interface Otherwise we will get "java.io.NotSerializableException".

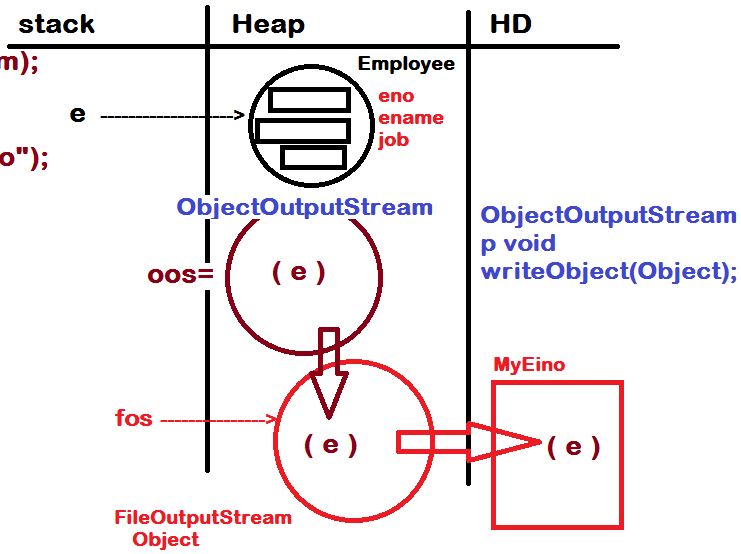
- Serializable is marker interface which is not any having abstract methods override just it as empty interface

- If any class implemented by Serializable interface then that corresponding class Object cab be persisted and it can transferred from one location to another via network

```
ObjectOutputStream(OutputStream);

FileOutputStream fos=
    new FileOutputStream("MyEinfo");

ObjectOutputStream oos=
  new ObjectOutputStream( fos );
        oos.writeObject( e );
```



Example:

```
//OOSDemo.java
import java.io.*;

class Employee implements Serializable
{
        int eno;
  String ename,job;  //instance fields

  void setEmployee(int no,String name,String job)
        { eno=no; ename=name; this.job=job; }

  void getEmployee()
        { System.out.println("Eno  is : "+eno);
     System.out.println("Ename is : "+ename);
          System.out.println("Job is : "+job); }
}

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

```
                Employee e=new Employee();
                e.setEmployee(101,"Shashi","DigitalCoach");

                FileOutputStream fos=new FileOutputStream("MyEinfo");
                ObjectOutputStream oos=new ObjectOutputStream(fos);

                        oos.writeObject(e);

                System.out.println("Object is Saved ");
                oos.close( );
                fos.close();
        }
}
```

Deserialization :

Process of reading the Objects from the File . in-order to achieve Deserialization then we have use "ObjectInputStream" Class

- To create an Object For "ObjectInputStream" then we have to use the following constructor

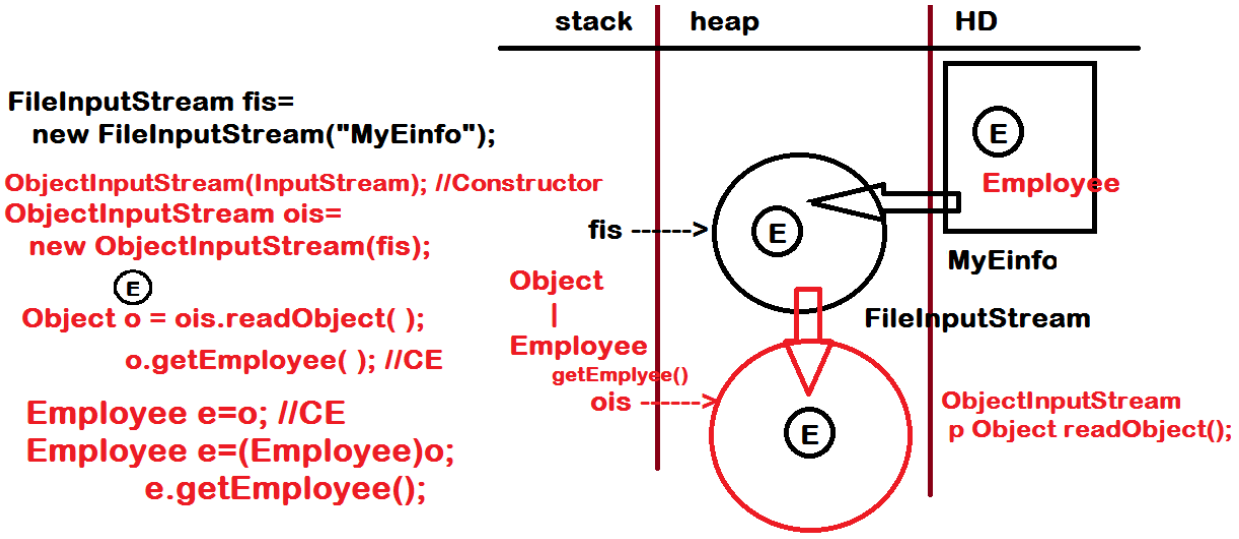
  ObjectInputStream(InputStream);

Eg: FileInputStream fis=new FileInputStream("MyEinfo");
ObjectInputStream ois=new ObjectInputStream(fis);

- To read an object then we have to use

java.io.ObjectInputStream

public final java.lang.Object readObject() throws java.io.IOException,ClassNotFoundException



Example:

```
//OISDemo.java
import java.io.*;

class OISDemo{

        public static void main(String args[ ]) throws Exception{

                FileInputStream fis=new FileInputStream("MyEinfo");

                ObjectInputStream ois=new ObjectInputStream(fis);

                Object o = ois.readObject();

                Employee e=(Employee)o;

                e.getEmployee();

                fis.close();

                ois.close();

        }

}
```