**Java .io package**

Java.io is a package which contains number of classes by using that classes we are able to send the data from one place to another place.

In java language we are transferring the data in the form of two ways:-

1. Byte format

2. Character format

**Stream/channel:-**

It is acting as medium by using steam or channel we are able to send particular data from one place to the another place.

Streams are two types:-

1. Byte oriented streams.(supports byte formatted data to transfer)

2. Character oriented stream.(supports character formatted data to transfer)

**Byte oriented streams:-**

**Java.io.FileInputStream**

To read the data from the destination file to the java application we have to use FileInputSream class.

To read the data from the .txt file we have to read() method.

**Java.io.FileOutputStream:-**

To write the data to the destination file we have to use the FileOutputStream.

To write the data to the destination file we have to use write() method.

Ex:- it will supports one character at a time.

import java.io.\*;

class Test

{

static FileInputStream fis;

static FileOutputStream fos;

public static void main(String[] args)

{

try{

fis=new FileInputStream("get.txt");

fos=new FileOutputStream("set.txt",true);

int c;

while ((c=fis.read())!=-1)

{ fos.write(c);

}

fis.close();

fos.close();

}

catch(IOException io)

{ System.out.println("getting IOException");

}

}

}

Ex:-it will support one character at a time.

import java.io.\*;

class Test

{

static FileReader fr;

static FileWriter fw;

public static void main(String[] args)

{

try{

fr=new FileReader("get.txt");

fw=new FileWriter("set.txt",true);

int c;

while ((c=fr.read())!=-1)

{

fw.write(c);

}

fr.close();

fw.close();

}

catch(IOException io)

{

System.out.println("getting IOException"); }

}

}

**Line oriented I/O:-**

Character oriented streams supports single character and line oriented streams supports single line data.

**BufferedReader**:- to read the data line by line format and we have to use readLine() to read the data.

**PrintWriter :-** to write the data line by line format and we have to use println() to write the data.

import java.io.\*;

class Test

{

static BufferedReader br;

static PrintWriter pw;

public static void main(String[] args)

{

try{

br=new BufferedReader(new FileReader("get.txt"));

pw=new PrintWriter(new FileWriter("set.txt"));

String line;

while ((line=br.readLine())!=null)

{

pw.println(line);

}

br.close();

pw.close();

}

catch(IOException io)

{

System.out.println("getting IOException");

}

} }

**Buffered Streams:-**

Up to we are working with non buffered streams these are providing less performance because these are interact with the hard disk, network.

Now we have to work with Buffered Streams

BufferedInputStream read the data from memory area known as Buffer.

We are having four buffered Stream classes

1. BufferedInputStream

2. BufferedOutputStream

3. BufferedReader

4. BufferedWriter

Ex:-

import java.io.\*;

class Test

{

static BufferedReader br;

static BufferedWriter bw;

public static void main(String[] args)

{

try{

br=new BufferedReader(new FileReader("Test1.java"));

bw=new BufferedWriter(new FileWriter("States.java"));

String str;

while ((str=br.readLine())!=null)

{

bw.write(str);

}

br.close();

bw.close();

}

catch(Exception e)

{

System.out.println("getting Exception");

}

}

}

Ex:-

import java.io.\*;

class Test

{

static BufferedInputStream bis;

static BufferedOutputStream bos;

public static void main(String[] args)

{

try{

bis=new BufferedInputStream(new FileInputStream("abc.txt"));

bos=new BufferedOutputStream(new FileOutputStream("xyz.txt"));

int str;

while ((str=bis.read())!=-1)

{

bos.write(str);

}

bis.close();

bos.close();

}

catch(Exception e)

{

System.out.println(e);

System.out.println("getting Exception");

}

}

}

**Ex:-**

import java.io.\*;

class Test

{

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

{

BufferedReader br=new BufferedReader(new FileReader("abc.txt"));

String str;

while ((str=br.readLine())!=null)

{

System.out.println(str);

}

}

}

**Java.util.Scanner:-**

By using Scanner class we are able to divide the String into the number of tokens.

To get the integer value from the keyboard-------:-s.nextInt()

To get the String value from the keyboard---------:-s.next()

To get the floating values from the keyboard------:-s.nextFloat ();

i mport java.util.\*;

class Test

{

public static void main(String[] args)

{

while (true)

{

Scanner s=new Scanner(System.in);

System.out.println("enter emp no");

int eno=s.nextInt();

System.out.println("enter emp name");

String ename=s.next();

System.out.println("enter emp salary");

float esal=s.nextFloat();

System.out.println("emp no----->"+eno);

System.out.println("emp name---->"+ename);

System.out.println("emp sal------>"+esal);

System.out.println("do u want one more record(yes/no)");

String option=s.next();

if (option.equals("no"))

{

break;

}

}

}

} Note :-

hasNext() method return true if the Scanner has another token as its input the return type is boolean.

next() is return next complete token from the Scanner

Ex:-

import java.io.\*;

import java.util.\*;

class Test

{

public static void main(String[] args)

{ try{

Scanner s=new Scanner(new BufferedReader(new FileReader("abc.txt")));

while (s.hasNext())

{

System.out.println(s.next());

}

}

catch(Exception e)

{

System.out.println(e);

System.out.println("getting Exception");

}

}

}

**Serialization:-**

The process of saving an object to a file (or) the process of sending an object across the network is called serialization.

But strictly speaking the process of converting the object from java supported form to the network supported form of file supported form.

To do the serialization we required fallowing classes

1. FileOutputStream

2. ObjectOutputStream

**Deserialization:-**

The process of reading the object from file supported form or network supported form to the java supported form is called deserialization.

We can achieve the deserialization by using fallowing classes.

1. FileInputStream

2. ObjectInputStream

Ex:-Student.java

import java.io.Serializable;

public class Student implements Serializable

{

int id;

String name;

int marks;

public Student(int id, String name,int marks)

{

this.id = id;

this.name = name;

this.marks=marks;

}

}

**To perform serialization :- we are writing the object data to the file called abc.txt file we are transferring that file across the network.**

import java.io.\*;

class Serializable1

{

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

{

Student s1 =new Student(211,"ravi",100);

FileOutputStream fos=new FileOutputStream("abc.txt",true);

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(s1);

oos.flush();

System.out.println("Serializable process success");

}

}

**To perform deserialization:- in the network the file is available with java data to read the data we have to go for deserialization.**

import java.io.\*;

class Deserialization

{

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

{

//deserialization process

FileInputStream fis=new FileInputStream("abc.txt");

ObjectInputStream ois=new ObjectInputStream(fis);

Student s=(Student)ois.readObject();

System.out.println("the student name is:"+s.name);

System.out.println("the stuent id is:"+s.id);

System.out.println("the student marks:"+s.marks);

System.out.println("deserialization success"); } }

**Transient Modifier:-**

Transient modifier is the modifier applicable for only variables and we can’t apply for methods and classes.

At the time of serialization, if we don’t want to save the values of a particular variable to meet security constraints then we should go for transient modifier.

At the time of serialization JVM ignores the original value of transient variable and default value will be serialized.

import java.io.\*;

import java.io.Serializable;

class Student implements Serializable

{ transient int id=100;

transient String name="ravi";

}

class Serializable1

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

{ Student s1=new Student();

System.out.println("the stuent id is:"+s1.id);

System.out.println("the student name is:"+s1.name);

FileOutputStream fos=new FileOutputStream("chandu.txt",true);

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(s1);

FileInputStream fis=new FileInputStream("chandu.txt");

ObjectInputStream ois=new ObjectInputStream(fis);

Student s=(Student)ois.readObject();

System.out.println("the stuent id is:"+s.id);

System.out.println("the student name is:"+s.name);

}

}