I/O package

Input — meant for read
output — meant for write.

The java.io package provides classes and interfaces which allows to read from and write to data the console, file, network socket etc.

- Java provides I/O operations through stream.
- A stream linked to a physical layer by java I/O system to make input and output operations
- A stream is continuous flow of data.
- Java ~~has~~ uses two types of streams.

> Byte stream — It handles input and output of bytes.

> Character stream — It handles input and output of characters.
> - It uses unicode system

Byte Stream Classes

] Abstract class

InputStream
(used to read data from console (keyboard), file, socket.)

OutputStream.
(used to write data to monitor, printer, file, socket)

> These two classes are abstract classes which are inherited by other classes

Other classes of Stream. Class.

All are subclasses of InputStream and OutputStream

1. FileInputStream	— ~~Reads~~ provides methods to read data from file in form of byte.
2. FileOutputStream	— provides methods to write data to file in form of byte.
3. DataInputStream	— provides methods to read from file using standard datatypes (reads int type, double type etc)
4. DataOutputStream	— write data to file using standard datatype.
5. BufferedInputStream	— Uses buffer for reading
6. BufferedOutputStream	— uses buffer for writing
7. ObjectInputStream	— Reads object from file
8. ObjectOutputStream	— Writes object to file.
9. PrintStream	— Output stream that contains print() and println() method.

All the classes have two common methods of InputStream and OutputStream class.

1. byte read() — Reads ~~to~~ data in byte datatype.
2. void write(byte b) — Write data in byte datatype.

~~Character St~~

~~Exa How to read data from keyboard using InputStream.~~

~~InputStr~~

Character Stream classes

Reader   Writer

Two classes are abstract classes which are inherited by other classes which provides methods to handle unicode characters.

Other classes

~~1. Buffer~~

1. InputStreamReader — ~~it~~ provides methods which translates bytes to character type.
2. OutputStreamReader — provides methods which translates character to byte.

3. PrintWriter —— contains print() and println() method.

4. FileReader —— Reads data from file in character type.

5. FileWriter — Writes data to file.

6. BufferedReader — Handles inputStreamReader which uses buffer.

7 BufferedWriter :— Handles outputStreamReader which uses buffer.

Q How to read data from keyboard.?

// input io package.

// The io operation always throws IOException
// so write the code in try-catch block.

```
import java.io.*;
class readkeyboard
{ public static void main(String[] s)
 {
   try
   { String st;
   InputStreamReader ir = ~~new InputStream~~
              new InputStreamReader(System.in);
   BufferedReader br =
              new BufferedReader(ir);
                        ↓
              object of InputStreamReader)
st = br.readLine(); // Reads string.
System.out.println(st);
}
catch(IOException e)
{ }
}
}
```

Q program to write data to a file.

```
import java.io.*;
class writeFile
{ public static void main(String[] s)
 { try
   { // create a file object carrying filename
    File f=new File("d:/myFile.txt");
    FileWriter fw =new FileWriter(f);
    String st = "Welcome to File handling";
     fw.write(st);
     fw.close();
     f.close();
   }
  catch(IOException e)
   { }
 }
}
```

Q Reading data from myfile.txt file.

```
import java.io.*;
class readFile
{ public static void main(String[] s)
  {
   try
   { File f=new File("d:\myfile.txt");
  [ BufferedReader br = new BufferedReader(
          new FileReader(f));

            or

  [ FileReader fr=new FileReader(f);
   BufferedReader br=new
          BufferedReader(fr);

   String st;
   while((st =br.readLine()) !=null)
   { System.out.println(st);
   }
   br.close();
   fb.close();
   fr.close();
  }
 catch(IOException e) {}
  } }
```

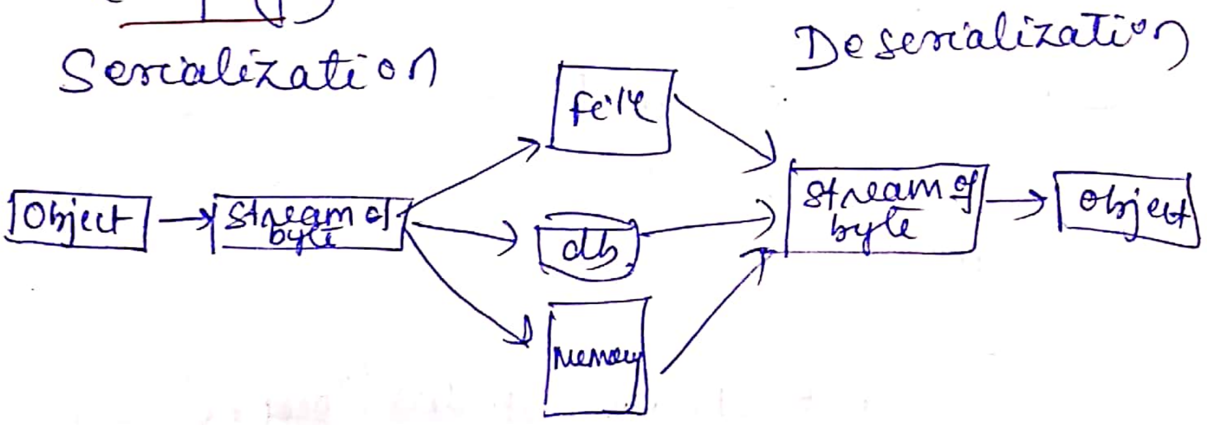
Serialization and Deserialization

Serialization is the process of converting an object into sequence of bytes which can be stored or write into a file or database which can be sent through stream.

Deserialization is the process of converting sequence of bytes into object.

To write an object into a file the corresponding class of that object must implement Serializable interface present in java.io package.

Serializable interface is a marker interface which has no ~~met~~ abstract methods (empty)



ObjectOutputStream provides methods to write object to the file
void writeObject(Object)

ObjectInputStream provides methods to read object from a file

Object readObject()

Ex 4 Write a student object into a file

```
import java.io.*;
Class Student implements Serializable.
{ String name;
  int roll;
  Student(String name, int roll)
  { this.name = name;
    this.roll = roll;
  }
}

Class Serialize
{ public static void main(String[] s)
  {
    try
    {
    Student s1 = new Student("Deepak", 1).
  // create file object.
FileOutputStream FO = new FileOutputStream(
                        "std.txt").
```

you can pass file object like previous page

```
ObjectOutputStream oo = new
        ObjectOutputStream(fo);
    oo. writeObject(s1);

    oo. close();
    fo. close();
  }
  catch(IOException e){ }
   }
  }
```

Read the Student object from std.txt.

```
import java.io.*;
  Class Deserial
  {public static void main(String[] s)
  {
    Student st = null;
  try
  {FileInputStream fi = new FileInputStream("std.txt");
  ObjectInputStream oi = new ObjectInputStream(fi);
    st = (Student) oi. readObject();
             ↓ typecasting since the method returns
                                  Object type.
    System.out.println(st. name+"  "+ st.roll);
   }
~~Catch(Exception e)~~
Catch(IOException e)
  { }
 }
}
```

transient data

If you don't want to write any data of a class to file then declare it transient.

Ex:

transient ~~static~~ String collegename;

```
Class Student
{ int roll;
  String name;
  transient String college="CIME";
}
```

not serialized i.e. not written to file.