Module – VI

Basic I/O Operations


Java
JavaScript
Python
C++
C
Objective-C
Perl
Haskell
Ruby
PHP
Pascal
Assembly
Scratch
Scheme
R
Q
PostScript
Logo
Prolog
ColdFusion
Forth
PL/SQL
Ada
Lua
Yacc
TCL
PL/I
Delphi
MATLAB
ActionScript
Fortran
Go
Lisp
Swift
Mathematica
Groovy
Tcl
Haskell
SAS
cT
Max/MSP
Scala
Dart
COBOL
Erlang
C#
D
Scala
F#
ML
OpenEdge ABL
ABAP
Visual Basic
Transact-SQL
Clarion
FoxPro

What you will Learn

- Java IO
- Types of Stream

Java I/O

- **Java I/O** (Input and Output) is used *to process the input* and *produce the output*.
- Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.
- We can perform **file handling in Java** by Java I/O API.

Stream

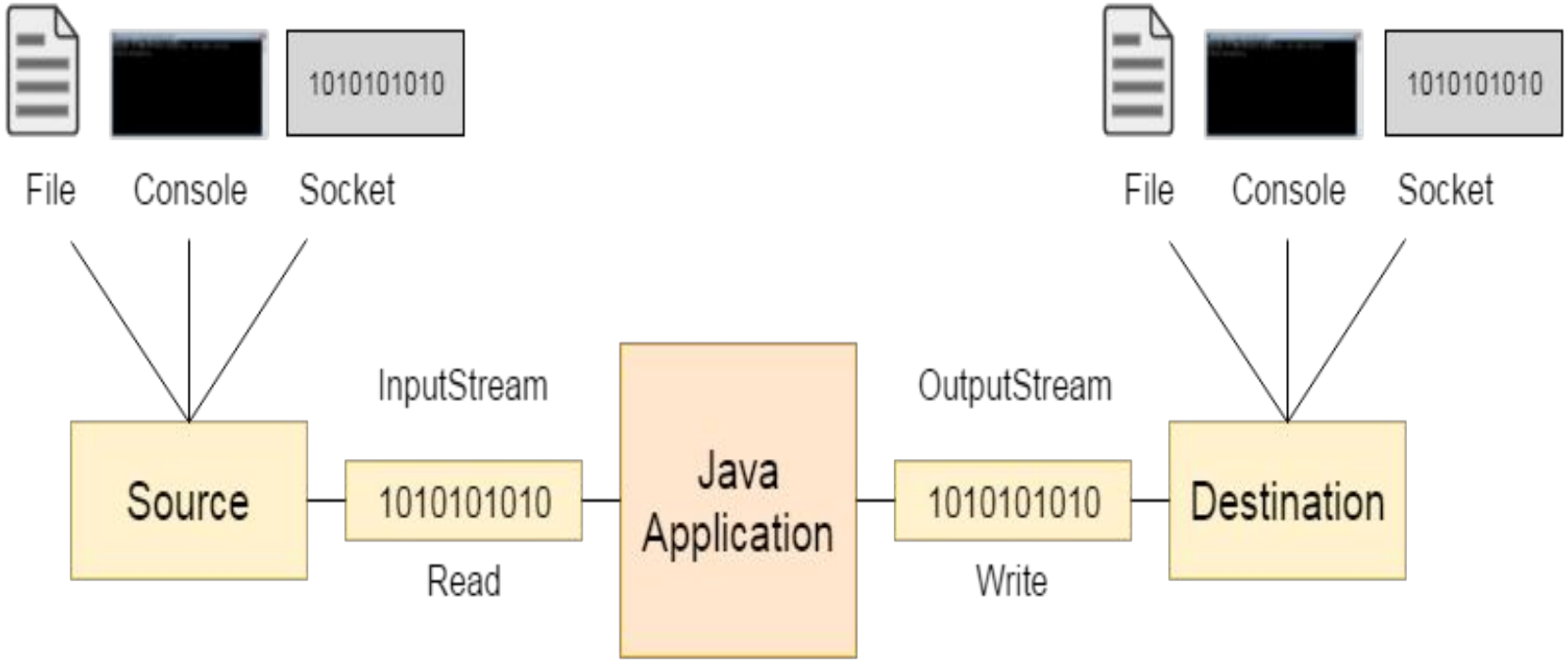
- A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow
- In Java, 3 streams are created for us automatically. All these streams are attached with the console.

  - 1) System.out: standard output stream
  - 2) System.in: standard input stream
  - 3) System.err: standard error stream

OutputStream vs InputStream

- OutputStream
  - Java application uses an output stream to write data to a destination; it may be a file, an array, peripheral device or socket.
- InputStream
  - Java application uses an input stream to read data from a source; it may be a file, an array, peripheral device or socket.
-

OutputStream vs InputStream



OutputStream class

- OutputStream class is an abstract class.
- It is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.
-

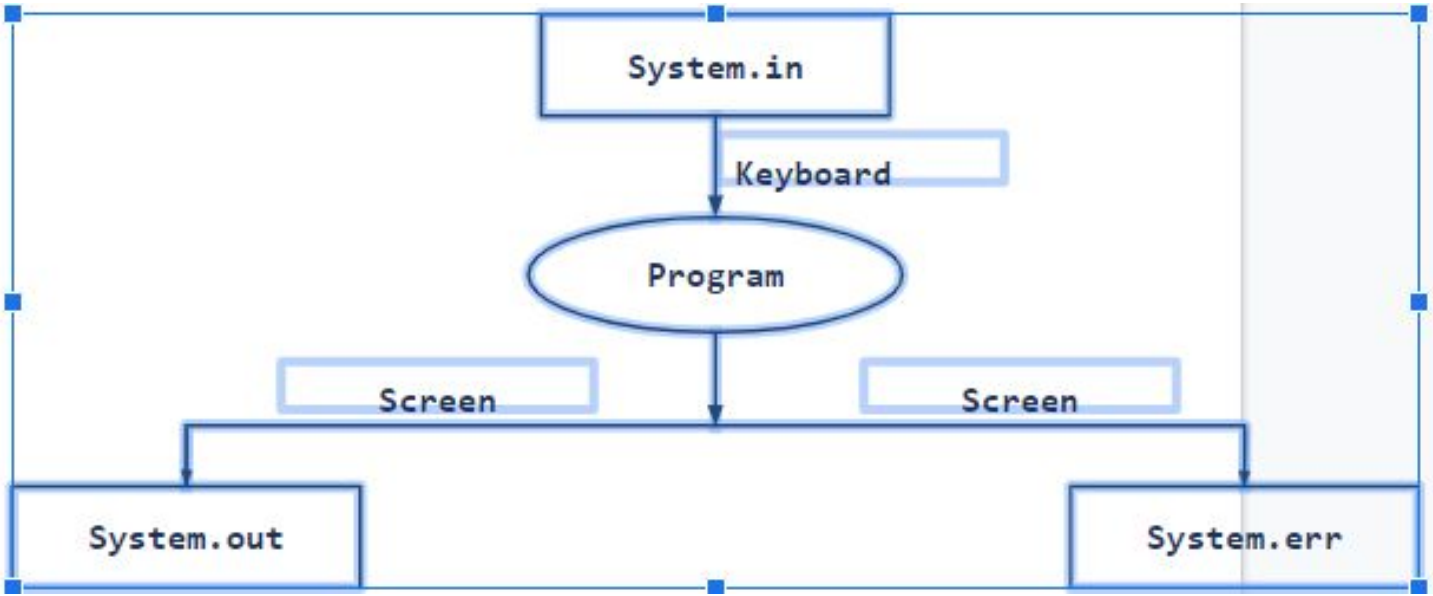
Method	Description
1) public void write(int)throws IOException	is used to write a byte to the current output stream.
2) public void write(byte[])throws IOException	is used to write an array of byte to the current output stream.
3) public void flush()throws IOException	flushes the current output stream.
4) public void close()throws IOException	is used to close the current output stream.

InputStream class

- InputStream class is an abstract class. It is the superclass of all classes representing an input stream of bytes.

Method	Description
1) public abstract int read()throws IOException	reads the next byte of data from the input stream. It returns -1 at the end of the file.
2) public int available()throws IOException	returns an estimate of the number of bytes that can be read from the current input stream.
3) public void close()throws IOException	is used to close the current input stream.

Default Stream



Standard I/O

- **System.in –** This is the **standard input stream** that is used to read characters from the keyboard or any other standard input device

- **System.out –** This is the **standard output stream** that is used to produce the result of a program on an output device like the computer screen

- **print() –** This method in Java is used to display a text on the console. This text is passed as the parameter to this method in the form of String

Example


```
class Main{
  public static void main(String[] args){
    System.out.print(“ICT Academy");
    System.out.print(“Java");
  }
}
```

Example

- **println():** This method prints the text on the console and the cursor moves to the start of the next line at the console

```
class Main{
  public static void main(String[] args){
    System.out.println(“ICT Academy");
    System.out.println(“Java");
  }
}
```

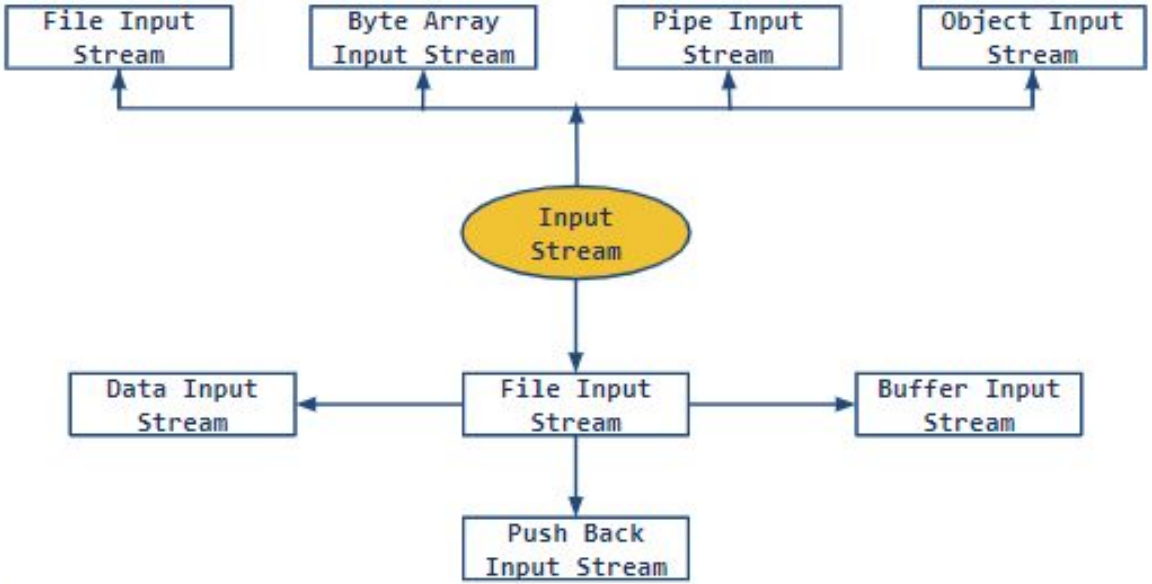
System.err

- This is the **standard error stream** that is used to output all the error data that a program might throw, on a computer screen or any standard output device

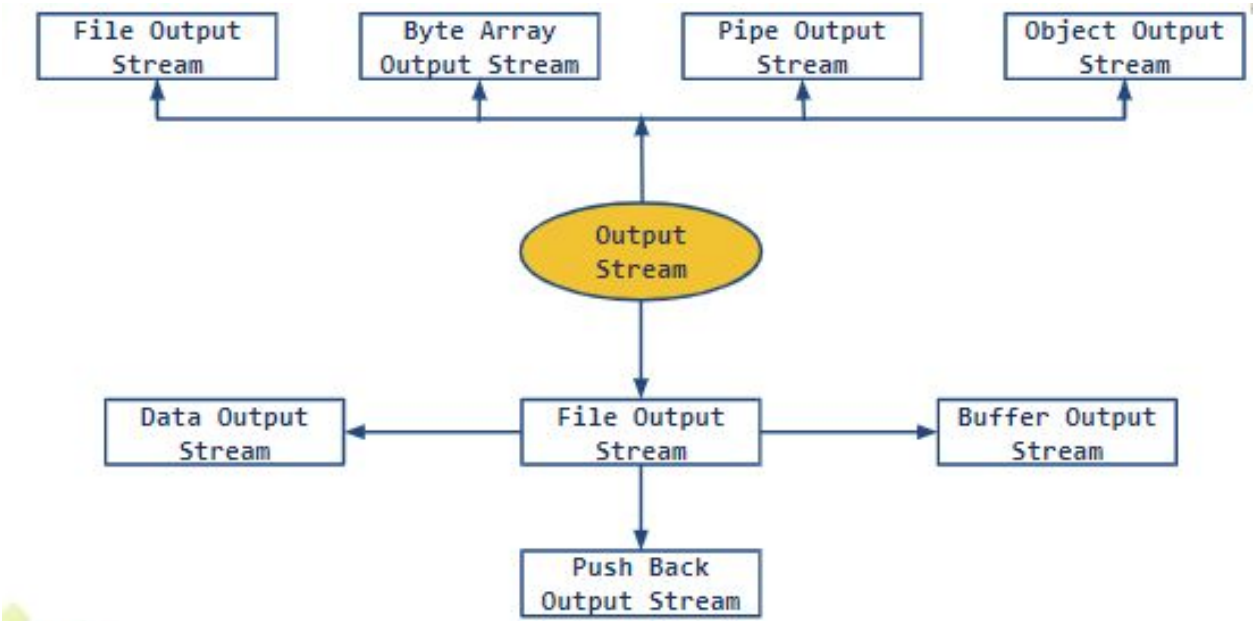
- This stream also uses all the 3 above-mentioned functions to output the error data:

    - print()
    - println()
    - printf()

Types of Stream– Input Stream



Types of Stream– Output Stream



Stream Classification

- **ByteStream:** This is used to process data byte by byte (8 bits). Though it has many classes, the FileInputStream and the FileOutputStream are the most popular ones

- **Character Stream:** In Java, characters are stored using Unicode conventions. Character stream automatically allows us to read/write data character by character. Though it has many classes, the FileReader and the FileWriter are the most popular ones

Stream Classification

Stream class	Description
• BufferedInputStream	• It is used for Buffered Input Stream
• DataInputStream	• It contains method for reading java standard data types
• FileInputStream	• This is used to reads from a file
• InputStream	• This is an abstract class that describes stream input
• PrintStream	• This contains the most used print() and println() method
• BufferedOutputStream	• This is used for Buffered Output Stream
• DataOutputStream	• This contains method for writing java standard data types
• FileOutputStream	• This is used to write to a file
• OutputStream	• This is an abstract class that describe stream output

File Output steam & File Input stream in Java I/O

ICTACADEMY

- In Java, FileInputStream and FileOutputStream classes are used to read and write data in file

- Java FileOutputStream is an output stream for writing data to a file

• **public void write(int)throws IOException:**	is used to write a byte to the current output stream
• **public void close()throws IOException:**	is used to close the current output stream

File Output steam & File Input stream in Java I/O

Java FileInputStream class obtains input bytes from a file

It is used for reading streams of raw bytes such as image data.

• **public abstract int read()throws IOException:**	The methods returns the next byte of data, or -1 if the end of the file is reached
• **public void close()throws IOException:**	Used to close the current input stream

Example

ICTACADEMY

```
import java.io.FileInputStream;
 public class Fileoutputstream {
   public static void main(String args[]){
      try{
         FileOutputStream fout=new
FileOutputStream("test2.txt");
         String s = "Welcome to ICT Academy";
         byte b[] = s.getBytes();
         //converting string into byte array
         System.out.println();
         fout.write(b);
         fout.close();
         System.out.println("success...");
       }catch(Exception e){
         System.out.println(e);
       }
```

```
   try{
        FileInputStream fin=new
FileInputStream("test2.txt");
        int i = 0;
        while((i = fin.read())! = -1){
          System.out.print((char)i);
        }
        fin.close();
    }catch(Exception e){
        System.out.println(e);
     }

    }
}
```

Data Output steam & Data Input stream in Java I/O

ICTACADEMY

- Java **DataOutputStream** class allows an application to write primitive Java data types to the output stream in a machine-independent way

- **UTF-8 is 8**-bit Unicode Transformation Format

• **java.io.DataOuputStream.writeUTF(String str)**	writes a string to the underlying output stream using modified UTF-8 encoding.

Data Output steam & Data Input stream in Java I/O

ICTACADEMY

- **Data Input Stream** allows an application to read primitive data from the input stream in a machine independent way

• **java.io.DataInputStream.readUTF()**	Reads in a string that has been encoded using a modified UTF-8 format and then the string of character is decoded from the UTF and returned as **String**.

Example


```
import java.io.*;
public class datastream{

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

// Writing data to the file

   DataOutputStream dataOut = new
DataOutputStream(new
FileOutputStream("file.txt"));
   dataOut.writeUTF("Hii,Here is Java");
```

```
  // Reading data from the same file
   DataInputStream dataIn = new
DataInputStream(new
FileInputStream("file.txt"));

   while(dataIn.available() > 0){
      String k = dataIn.readUTF();
      System.out.println(k+" ");
   }
   }
}
```

Buffered Output steam in Java I/O

- Java **BufferedOutputStream** class uses an internal buffer to store data

- It adds more efficiency than to write data directly into a stream. So, it makes the performance fast

Example

- we are writing the textual information in the BufferedOutputStream object which is connected to the FileOutputStream object
- The flush() flushes the data of one stream and send it into another
- It is required if you have connected the one stream with another

Example


```
import java.io.*;
class Write{
 public static void main(String args[])throws Exception{
   FileOutputStream fout=new FileOutputStream("file.txt");
   BufferedOutputStream bout=new BufferedOutputStream(fout);
   String s="MS Dhoni is my favourite player";
   byte b[]=s.getBytes();
   bout.write(b);

   bout.flush();
   bout.close();
   fout.close();
   System.out.println("success");
 }
}
```

Buffered Input stream in Java I/O

- Java BufferedInputStream class is used to read information from stream

- It internally uses buffer mechanism to make the performance fast

Example- Reading data of a file


```
import java.io.*;
class Read{
 public static void main(String args[]){
  try{
    FileInputStream fin=new FileInputStream("file.txt");
    BufferedInputStream bin=new BufferedInputStream(fin);
    int i;
    while((i=bin.read())!=-1){
     System.out.println((char)i);
    }
    bin.close();
    fin.close();
  }catch(Exception e){system.out.println(e);}
 }
}
```

Java FileWriter and FileReader (File Handling in java)

- Java **FileWriter** and **FileReader** classes are used to write and read data from text files

- These are character-oriented classes, used for file handling in java

- Java has suggested not to use the FileInputStream and FileOutputStream classes if you have to read and write the textual information

Java FileWriter class

- Java FileWriter class is used to write character-oriented data to the file

Constructors of FileWriter

Constructor	Description
• FileWriter(String file)	creates a new file. It gets file name in string.
• FileWriter(File file)	creates a new file. It gets file name in File object.

Methods of FileWriter class

Method	Description
• public void write(String text)	writes the string into FileWriter.
• public void write(char c)	writes the char into FileWriter.
• public void write(char[] c)	writes char array into FileWriter.
• public void flush()	flushes the data of FileWriter.
• public void close()	closes FileWriter.

Example


```
import java.io.*;
class Writer{
 public static void main(String args[]){
  try{
   FileWriter fw=new FileWriter(“new.txt");
   fw.write("my name is Rohan");
   fw.close();
  }catch(Exception e){System.out.println(e);}
  System.out.println("success");
 }
}
```

Java FileReader class

- Java **FileReader** class is used to read data from the file. It returns data in byte format like FileInputStream class

Constructors of FileReader

Constructor	Description
• FileReader(String file)	It gets filename in string. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException
• FileReader(File file)	It gets filename in file instance. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException

Methods of FileReader class

Method	Description
• public int read()	returns a character in ASCII form. It returns –1 at the end of file.
• public void close()	closes FileReader.

Example

```
import java.io.*;
class Reader{
 public static void main(String args[])throws Exception{
  FileReader fr=new FileReader(“new.txt");
  int i;
  while((i=fr.read())!=-1)
  System.out.println((char)i);

  fr.close();
 }
}
```