Saraswati Vandana

या कुन्देन्दु तुषार हार धवला
या शुभ्र वस्त्रान्विता ।
या वीणा वर दंड मंडितकरा
या श्वेत पद्मासना ॥

या ब्रह्मा अच्युत शंकर प्रभ्रतिभिः
देवै सदा पूजिता ।
सा मां पातु सरस्वती भगवती
निःश्येश जाड्यापह ॥

Faculty of Engineering & Technology
Sankalchand Patel College of Engineering, Visnagar

Java Programming
(1ET1030406)

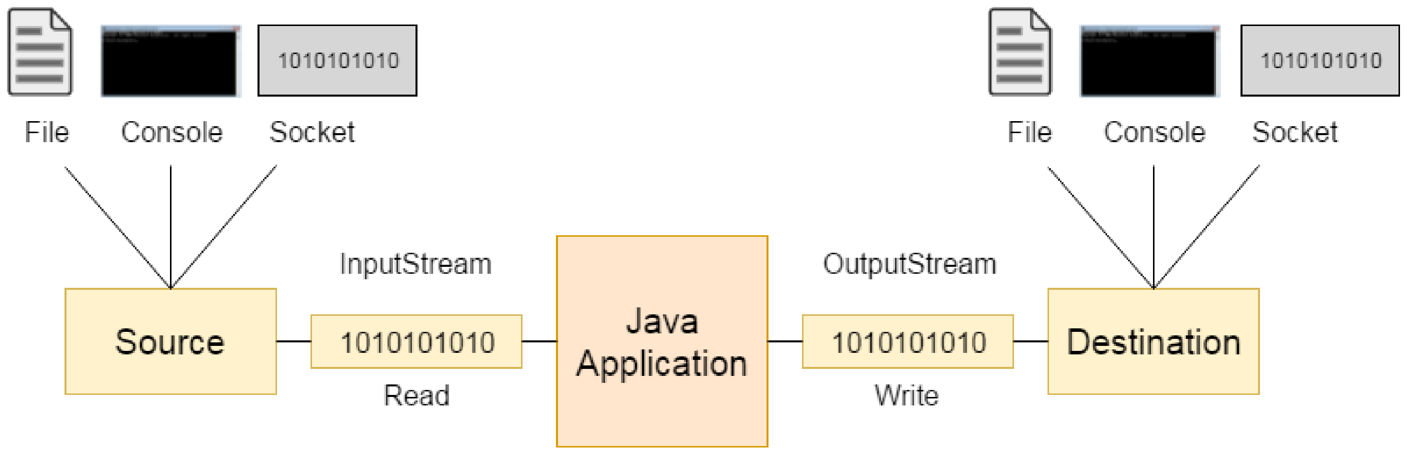
Unit-7 : I/O Programming

Prepared By
Mr. Mehul S. Patel
Department of Computer Engineering & Information Technology

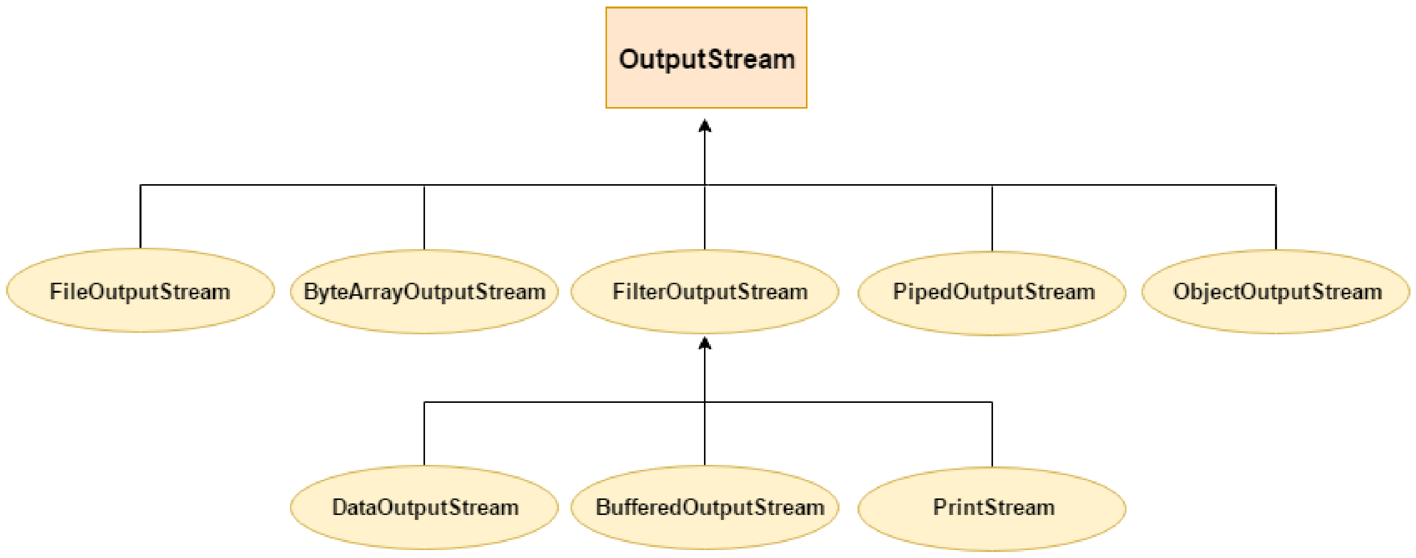
Content

Introduction to Stream

- Console Streams - System.out, System.in, System.err



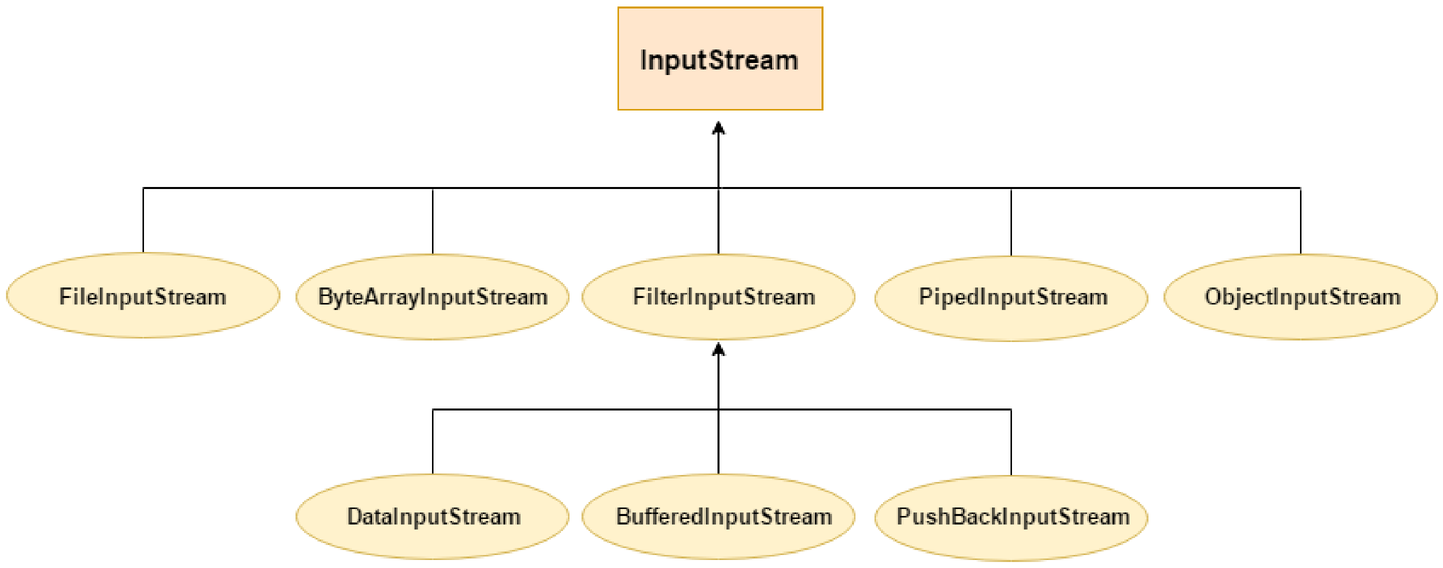
Output Stream



Output Stream – Useful methods

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.

Input Stream



Input Stream – Useful methods

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 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.

File Output Stream

Method	Description
protected void finalize()	It is sued to clean up the connection with the file output stream.
void write(byte[] ary)	It is used to write **ary.length** bytes from the byte array to the file output stream.
void write(byte[] ary, int off, int len)	It is used to write **len** bytes from the byte array starting at offset **off** to the file output stream.
void write(int b)	It is used to write the specified byte to the file output stream.
void close()	It is used to closes the file output stream.

File Output Stream - Example

```
import java.io.FileOutputStream;
public class FileOutputStreamExample {
    public static void main(String args[]){
            try{
        FileOutputStream fout=new FileOutputStream("D:\\testout.txt");
            fout.write(65);
            fout.close();
            System.out.println("success...");
           }catch(Exception e){System.out.println(e);}
     }
}
```

File Output Stream - Example

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

File Input Stream

Method	Description
int available()	It is used to return the estimated number of bytes that can be read from the input stream.
int read()	It is used to read the byte of data from the input stream.
int read(byte[] b)	It is used to read up to **b.length** bytes of data from the input stream.
int read(byte[] b, int off, int len)	It is used to read up to **len** bytes of data from the input stream.
long skip(long x)	It is used to skip over and discards x bytes of data from the input stream.
void close()	It is used to closes the stream.

File Input Stream - Example

```
import java.io.FileInputStream;
public class DataStreamExample {
     public static void main(String args[]){
          try{
            FileInputStream fin=new FileInputStream("D:\\testout.txt");
            int i=fin.read();
            System.out.print((char)i);
            fin.close();
          }catch(Exception e){System.out.println(e);}
         }
        }
```

File Input Stream - Example

```
import java.io.FileInputStream;
public class DataStreamExample {
     public static void main(String args[]){
          try{
            FileInputStream fin=new FileInputStream("D:\\testout.txt");
            int i=0;
            while((i=fin.read())!=-1){
             System.out.print((char)i);
            }
            fin.close();
          }catch(Exception e){System.out.println(e);}
         }
        }
```

Buffered Output Stream

Constructor	Description
BufferedOutputStream(OutputStream os)	It creates the new buffered output stream which is used for writing the data to the specified output stream.
BufferedOutputStream(OutputStream os, int size)	It creates the new buffered output stream which is used for writing the data to the specified output stream with a specified buffer size.

Buffered Output Stream

Method	Description
void write(int b)	It writes the specified byte to the buffered output stream.
void write(byte[] b, int off, int len)	It write the bytes from the specified byte-input stream into a specified byte array, starting with the given offset
void flush()	It flushes the buffered output stream.

Buffered Output Stream - Example

```
import java.io.*;
public class BufferedOutputStreamExample{
public static void main(String args[])throws Exception{
    FileOutputStream fout=new FileOutputStream("D:\\testout.txt");
    BufferedOutputStream bout=new BufferedOutputStream(fout);
    String s="Welcome to javaTpoint.";
    byte b[]=s.getBytes();
    bout.write(b);
    bout.flush();
    bout.close();
    fout.close();
    System.out.println("success");
}
}
```

Buffered Input Stream

Constructor	Description
BufferedInputStream(InputStream IS)	It creates the BufferedInputStream and saves it argument, the input stream IS, for later use.
BufferedInputStream(InputStream IS, int size)	It creates the BufferedInputStream with a specified buffer size and saves it argument, the input stream IS, for later use.

Buffered Input Stream

Method	Description
int available()	It returns an estimate number of bytes that can be read from the input stream without blocking by the next invocation method for the input stream.
int read()	It read the next byte of data from the input stream.
int read(byte[] b, int off, int ln)	It read the bytes from the specified byte-input stream into a specified byte array, starting with the given offset.
void close()	It closes the input stream and releases any of the system resources associated with the stream.
void reset()	It repositions the stream at a position the mark method was last called on this input stream.
void mark(int readlimit)	It sees the general contract of the mark method for the input stream.
long skip(long x)	It skips over and discards x bytes of data from the input stream.
boolean markSupported()	It tests for the input stream to support the mark and reset methods.

Buffered Input Stream - Example

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

Data Output Stream

Method	Description
int size()	It is used to return the number of bytes written to the data output stream.
void write(int b)	It is used to write the specified byte to the underlying output stream.
void write(byte[] b, int off, int len)	It is used to write len bytes of data to the output stream.
void writeBoolean(boolean v)	It is used to write Boolean to the output stream as a 1-byte value.
void writeChar(int v)	It is used to write char to the output stream as a 2-byte value.
void writeChars(String s)	It is used to write string to the output stream as a sequence of characters.
void writeByte(int v)	It is used to write a byte to the output stream as a 1-byte value.
void writeBytes(String s)	It is used to write string to the output stream as a sequence of bytes.
void writeInt(int v)	It is used to write an int to the output stream
void writeShort(int v)	It is used to write a short to the output stream.
void writeShort(int v)	It is used to write a short to the output stream.
void writeLong(long v)	It is used to write a long to the output stream.
void writeUTF(String str)	It is used to write a string to the output stream using UTF-8 encoding in portable manner.
void flush()	It is used to flushes the data output stream.

Data Output Stream - Example

```
import java.io.*;
public class OutputExample {
    public static void main(String[] args) throws IOException {
        FileOutputStream file = new FileOutputStream(D:\\testout.txt);
        DataOutputStream data = new DataOutputStream(file);
        data.writeInt(65);
        data.flush();
        data.close();
        System.out.println("Succcess...");
    }
}
```

Data Input Stream

Method	Description
int read(byte[] b)	It is used to read the number of bytes from the input stream.
int read(byte[] b, int off, int len)	It is used to read **len** bytes of data from the input stream.
int readInt()	It is used to read input bytes and return an int value.
byte readByte()	It is used to read and return the one input byte.
char readChar()	It is used to read two input bytes and returns a char value.
double readDouble()	It is used to read eight input bytes and returns a double value.
boolean readBoolean()	It is used to read one input byte and return true if byte is non zero, false if byte is zero.
int skipBytes(int x)	It is used to skip over x bytes of data from the input stream.
String readUTF()	It is used to read a string that has been encoded using the UTF-8 format.
void readFully(byte[] b)	It is used to read bytes from the input stream and store them into the buffer array.
void readFully(byte[] b, int off, int len)	It is used to read **len** bytes from the input stream.

Data Input Stream - Example

```
import java.io.*;
public class DataStreamExample {
  public static void main(String[] args) throws IOException {
    InputStream input = new FileInputStream("D:\\testout.txt");
    DataInputStream inst = new DataInputStream(input);
    int count = input.available();
    byte[] ary = new byte[count];
    inst.read(ary);
    for (byte bt : ary) {
      char k = (char) bt;
      System.out.print(k+"-");
    }
  }
}
```

File Writer

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.

Method	Description
void write(String text)	It is used to write the string into FileWriter.
void write(char c)	It is used to write the char into FileWriter.
void write(char[] c)	It is used to write char array into FileWriter.
void flush()	It is used to flushes the data of FileWriter.
void close()	It is used to close the FileWriter.

File Writer - Example

```
import java.io.FileWriter;
public class FileWriterExample {
    public static void main(String args[]){
         try{
           FileWriter fw=new FileWriter("D:\\testout.txt");
           fw.write("Welcome to javaTpoint.");
           fw.close();
          }catch(Exception e){System.out.println(e);}
          System.out.println("Success...");
     }
}
```

File Reader

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.

Method	Description
int read()	It is used to return a character in ASCII form. It returns -1 at the end of file.
void close()	It is used to close the FileReader class.

File Reader - Example

```
import java.io.FileReader;
public class FileReaderExample {
    public static void main(String args[])throws Exception{

        FileReader fr=new FileReader("D:\\testout.txt");
        int i;
        while((i=fr.read())!=-1)
        System.out.print((char)i);
        fr.close();
    }
}
```

Buffered Writer

Constructor	Description
BufferedWriter(Writer wrt)	It is used to create a buffered character output stream that uses the default size for an output buffer.
BufferedWriter(Writer wrt, int size)	It is used to create a buffered character output stream that uses the specified size for an output buffer.

Buffered Writer

Method	Description
void newLine()	It is used to add a new line by writing a line separator.
void write(int c)	It is used to write a single character.
void write(char[] cbuf, int off, int len)	It is used to write a portion of an array of characters.
void write(String s, int off, int len)	It is used to write a portion of a string.
void flush()	It is used to flushes the input stream.
void close()	It is used to closes the input stream

Buffered Writer - Example

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

    FileWriter writer = new FileWriter("D:\\testout.txt");
    BufferedWriter buffer = new BufferedWriter(writer);
    buffer.write("Welcome to javaTpoint.");
    buffer.close();
    System.out.println("Success");
    }
}
```

Buffered Reader

Constructor	Description
BufferedReader(Reader rd)	It is used to create a buffered character input stream that uses the default size for an input buffer.
BufferedReader(Reader rd, int size)	It is used to create a buffered character input stream that uses the specified size for an input buffer.

Buffered Reader

Method	Description
int read()	It is used for reading a single character.
int read(char[] cbuf, int off, int len)	It is used for reading characters into a portion of an array.
boolean markSupported()	It is used to test the input stream support for the mark and reset method.
String readLine()	It is used for reading a line of text.
boolean ready()	It is used to test whether the input stream is ready to be read.
long skip(long n)	It is used for skipping the characters.
void reset()	It repositions the stream at a position the mark method was last called on this input stream.
void mark(int readAheadLimit)	It is used for marking the present position in a stream.
void close()	It closes the input stream and releases any of the system resources associated with the stream.

Buffered Reader - Example

```
import java.io.*;
public class BufferedReaderExample {
    public static void main(String args[])throws Exception{
          FileReader fr=new FileReader("D:\\testout.txt");
          BufferedReader br=new BufferedReader(fr);
          int i;
          while((i=br.read())!=-1){
          System.out.print((char)i);
          }
          br.close();
          fr.close();
    }
}
```

File

Constructor	Description
File(File parent, String child)	It creates a new File instance from a parent abstract pathname and a child pathname string.
File(String pathname)	It creates a new File instance by converting the given pathname string into an abstract pathname.
File(String parent, String child)	It creates a new File instance from a parent pathname string and a child pathname string.
File(URI uri)	It creates a new File instance by converting the given file: URI into an abstract pathname.

File

Modifier and Type	Method	Description
static File	createTempFile(String prefix, String suffix)	It creates an empty file in the default temporary-file directory, using the given prefix and suffix to generate its name.
boolean	createNewFile()	It atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.
boolean	canWrite()	It tests whether the application can modify the file denoted by this abstract pathname.String[]
boolean	canExecute()	It tests whether the application can execute the file denoted by this abstract pathname.
boolean	canRead()	It tests whether the application can read the file denoted by this abstract pathname.

File

Modifier and Type	Method	Description
boolean	isAbsolute()	It tests whether this abstract pathname is absolute.
boolean	isDirectory()	It tests whether the file denoted by this abstract pathname is a directory.
boolean	isFile()	It tests whether the file denoted by this abstract pathname is a normal file.
String	getName()	It returns the name of the file or directory denoted by this abstract pathname.
String	getParent()	It returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.
Path	toPath()	It returns a java.nio.file.Path object constructed from the this abstract path.
URI	toURI()	It constructs a file: URI that represents this abstract pathname.

File

Modifier and Type	Method	Description
File[]	listFiles()	It returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname
long	getFreeSpace()	It returns the number of unallocated bytes in the partition named by this abstract path name.
String[]	list(FilenameFilter filter)	It returns an array of strings naming the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
boolean	mkdir()	It creates the directory named by this abstract pathname.

File - Example

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

        try {
            File file = new File("javaFile123.txt");
            if (file.createNewFile()) {
                System.out.println("New File is created!");
            } else {
                System.out.println("File already exists.");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

File - Example

```java
import java.io.*;
public class FileDemo2 {
    public static void main(String[] args) {
        String path = "";
        boolean bool = false;
        try {
            // createing  new files
            File file  = new File("testFile1.txt");
            file.createNewFile();
            System.out.println(file);
            // createing new canonical from file object
            File file2 = file.getCanonicalFile();
            // returns true if the file exists
            System.out.println(file2);
            bool = file2.exists();
            // returns absolute pathname
            path = file2.getAbsolutePath();
            System.out.println(bool);
            // if file exists
            if (bool) {
                // prints
                System.out.print(path + " Exists? " + bool);
            }
        } catch (Exception e) {
            // if any error occurs
            e.printStackTrace();
        }
    }
}
```

File - Example

```
import java.io.*;
public class FileExample {
public static void main(String[] args) {
    File f=new File("/Users/sonoojaiswal/Documents");
    String filenames[]=f.list();
    for(String filename:filenames){
        System.out.println(filename);
    }
}
}
```

File - Example

```
import java.io.*;
public class FileExample {
public static void main(String[] args) {
    File dir=new File("/Users/sonoojaiswal/Documents");
    File files[]=dir.listFiles();
    for(File file:files){
        System.out.println(file.getName()+" Can Write: "+file.canWrite()+
"
       Is Hidden: "+file.isHidden()+" Length: "+file.length()+" bytes");
    }
}
}
```

Scanner

Method	Description
public String next()	it returns the next token from the scanner.
public String nextLine()	it moves the scanner position to the next line and returns the value as a string.
public byte nextByte()	it scans the next token as a byte.
public short nextShort()	it scans the next token as a short value.
public int nextInt()	it scans the next token as an int value.
public long nextLong()	it scans the next token as a long value.
public float nextFloat()	it scans the next token as a float value.
public double nextDouble()	it scans the next token as a double value.

Scanner - Example

```
import java.util.Scanner;
class ScannerTest{
 public static void main(String args[]){
   Scanner sc=new Scanner(System.in);
   System.out.println("Enter your rollno");
   int rollno=sc.nextInt();
   System.out.println("Enter your name");
   String name=sc.next();
   System.out.println("Enter your fee");
   double fee=sc.nextDouble();
   System.out.println("Rollno:"+rollno+" name:"+name+" fee:"+fee);
   sc.close();
 }
}
```

References:

- https://www.javatpoint.com/java-io
- http://www.geeksforgeeks.org/java-io-packag/

Questions/Comments

THANK
YOU