Object Oriented Programming with Java (Subject Code: BCS-403)

Unit 2

Lecture 17

Lecture 17

- Byte Streams and Character Streams
- Reading and Writing File in Java

Byte Streams and Character Streams

In Java the streams are used for input and output operations by allowing data to be read from or written to a source or destination.

Java offers two types of streams:

- character streams
- byte streams.

Character Streams

- Character streams are designed to address character based records, which includes textual records inclusive of letters, digits, symbols, and other characters.
- These streams are represented by way of training that quit with the phrase "Reader" or "Writer" of their names, inclusive of FileReader, BufferedReader, FileWriter, and BufferedWriter.
- Character streams offer a convenient manner to read and write textual content-primarily based information due to the fact they mechanically manage character encoding and decoding.

Byte Streams

- Byte streams are designed to deal with raw binary data, which includes all kinds of data, including characters, pictues, audio, and video.
- These streams are represented through cclasses that cease with the word "InputStream" or "OutputStream" of their names,along with FileInputStream,BufferedInputStream, FileOutputStream and BufferedOutputStream.
- Byte streams offer a low-stage interface for studying and writing character bytes or blocks of bytes.

- They are normally used for coping with non-textual statistics, studying and writing files of their binary form, and running with network sockets.

Difference between Character Stream and Byte Stream in Java

Aspect	Character Streams	Byte Streams
Data Handling	Handle character-based data	Handle raw binary data
Representation	Classes end with "Reader" or "Writer"	Classes end with "InputStream" or "OutputStream"
Suitable for	Textual data, strings, human-readable info	Non-textual data, binary files, multimedia
Character Encoding	Automatic encoding and decoding	No encoding or decoding
Text vs non-Text data	Text-based data, strings	Binary data, images, audio, video

Performance	Additional conversion may impact performance	Efficient for handling large binary data
Handle Large Text Files	May impact performance due to encoding	Efficient, no encoding overhead
String Operations	Convenient methods for string operations	Not specifically designed for string operations
Convenience Methods	Higher-level abstractions for text data	Low-level interface for byte data
Reading Line by Line	Convenient methods for reading lines	Byte-oriented, no built-in line-reading methods
File Handling	Read/write text files	Read/write binary files

Java I/O

- **Java I/O** (Input and Output) is used to process the input and produce the output based on the input.
- Java uses the concept of stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

Stream

- A stream is a sequence of data. In Java a stream is composed of bytes. It's called a stream because it's like a stream of water that continues to flow.

In java, 3 streams are created for us automatically.

All these streams are attached with console.

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream

Code to print **output and error** message to the console.

```
System.out.println("simple message");
System.err.println("error message");
```

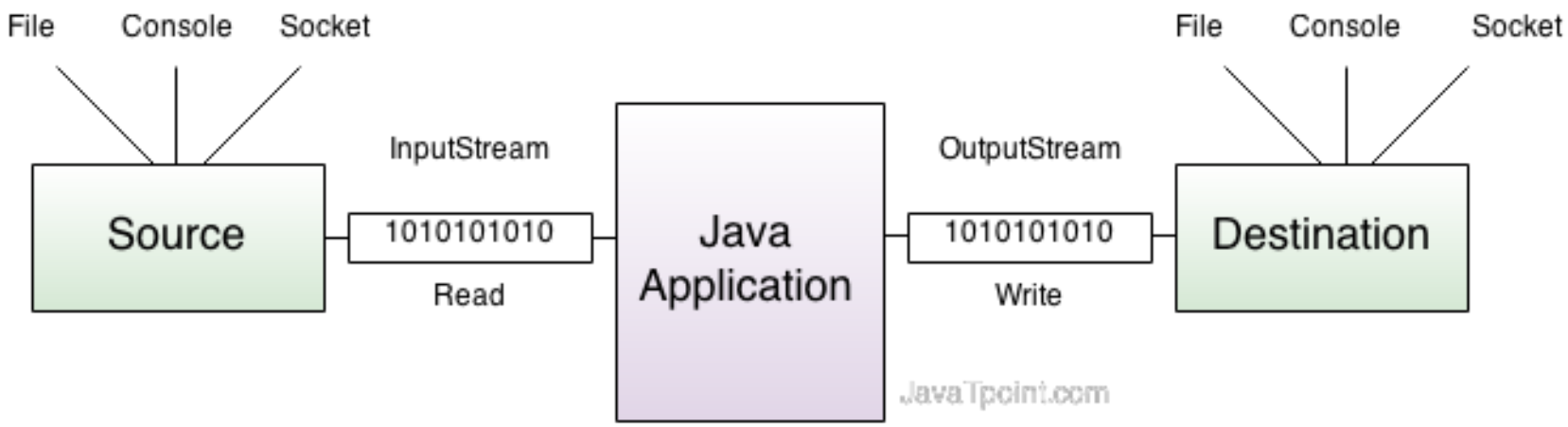
OutputStream

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.

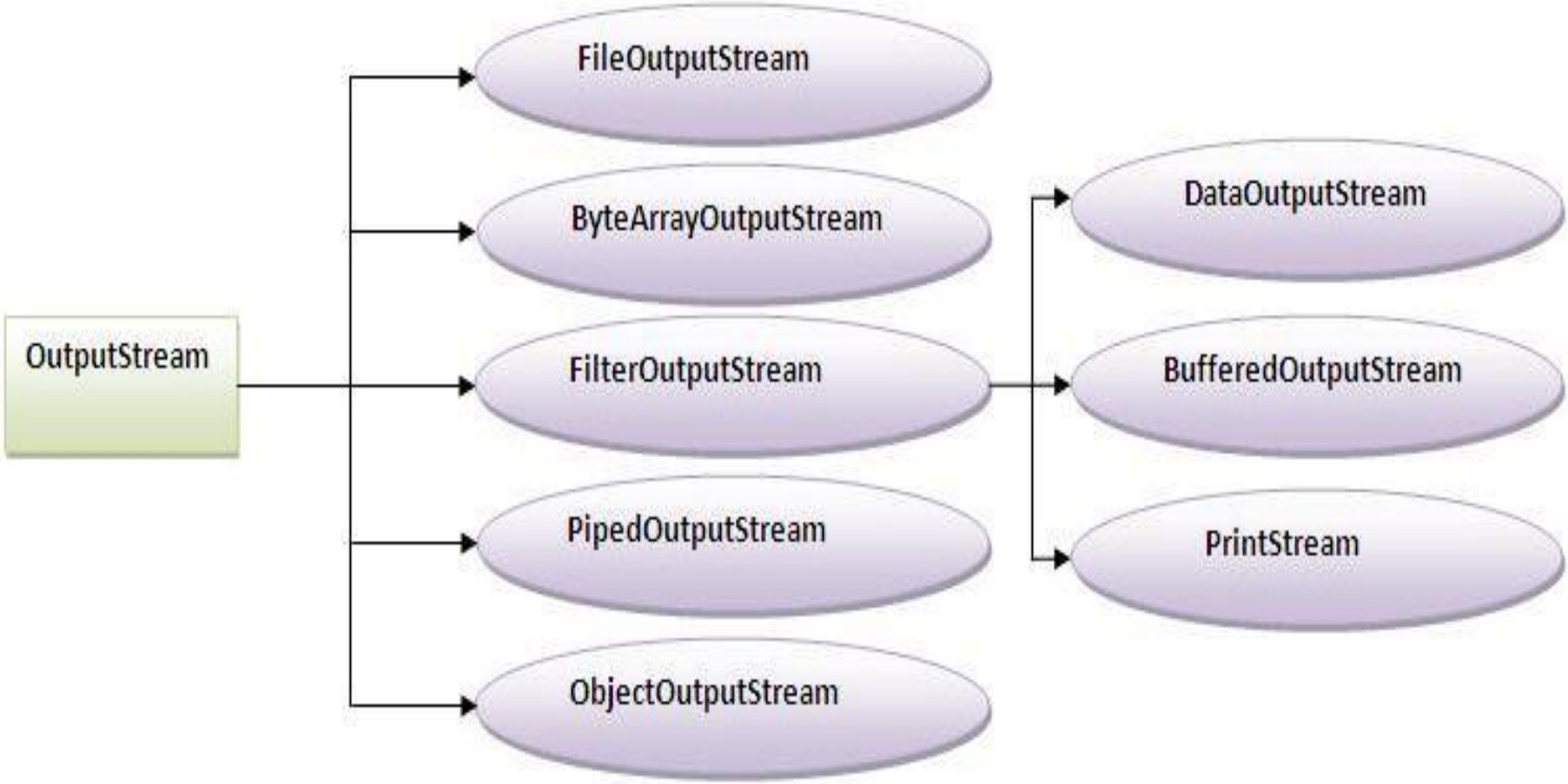
Working of Java OutputStream and InputStream



OutputStream class

- OutputStream class is an abstract class. It is the super class 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.

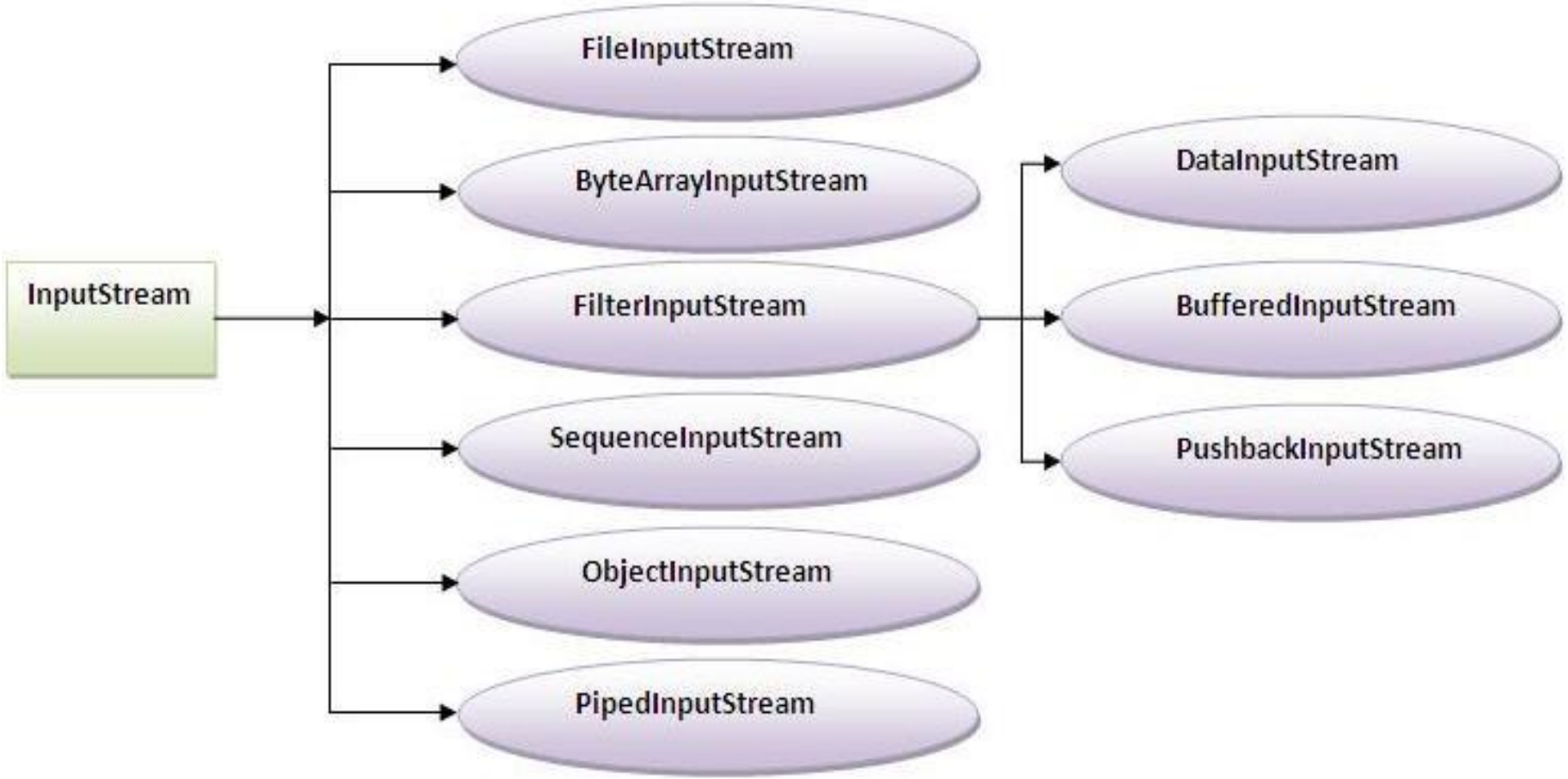

OutputStream
FileOutputStream
ByteArrayOutputStream
FilterOutputStream
PipedOutputStream
ObjectOutputStream
DataOutputStream
BufferedOutputStream
PrintStream

InputStream class

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

Commonly used methods of InputStream class

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.


InputStream
FileInputStream
ByteArrayInputStream
FilterInputStream
SequenceInputStream
ObjectInputStream
PipedInputStream
DataInputStream
BufferedInputStream
PushbackInputStream

FileInputStream and FileOutputStream (File Handling)

In Java, FileInputStream and FileOutputStream classes are used to read and write data in file. In another words, they are used for file handling in java.

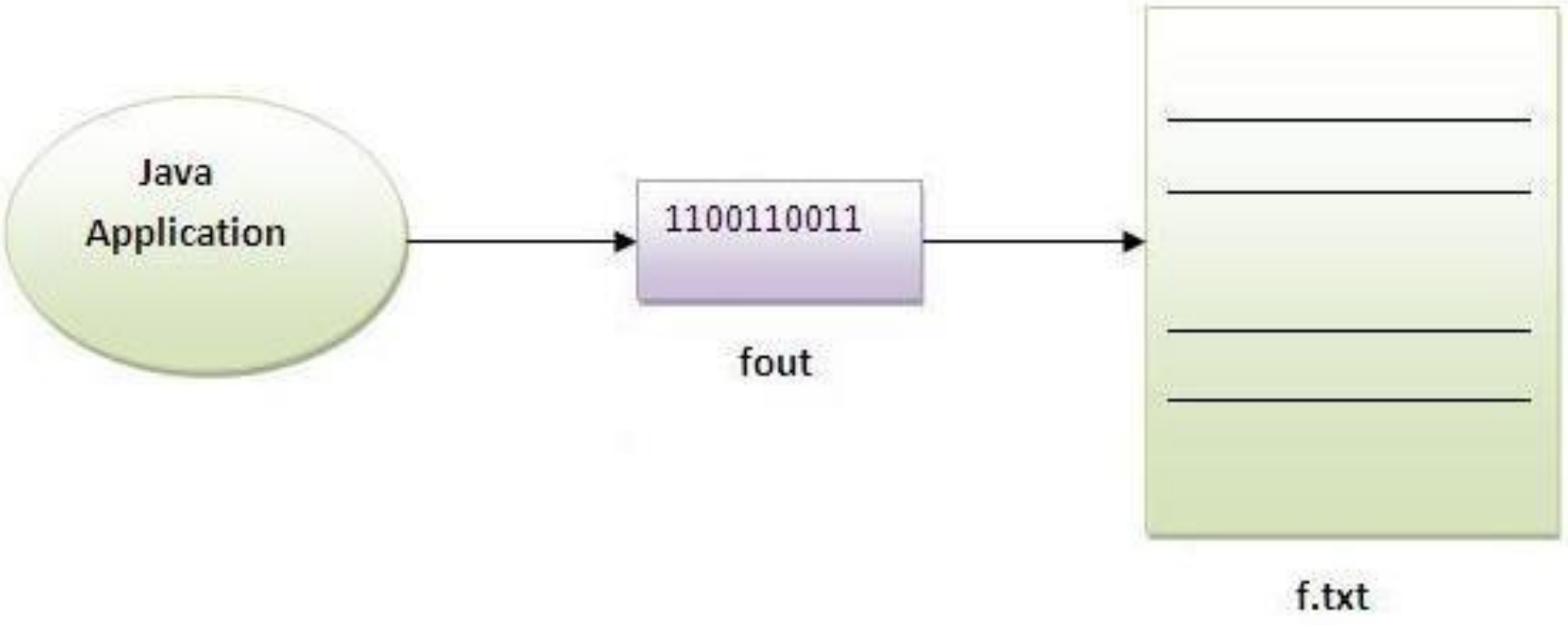
Java FileOutputStream class

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

If you have to write primitive values then use FileOutputStream. Instead, for character-oriented data, prefer FileWriter.But you can write byte-oriented as well as character-oriented data.

```
import java.io.*;
class Test{
  public static void main(String args[]){
   try{
     FileOutputstream fout=new FileOutputStream("abc.txt");
     String s="Sachin Tendulkar is my favourite player";
     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);}
  }
}
```

Output:success...

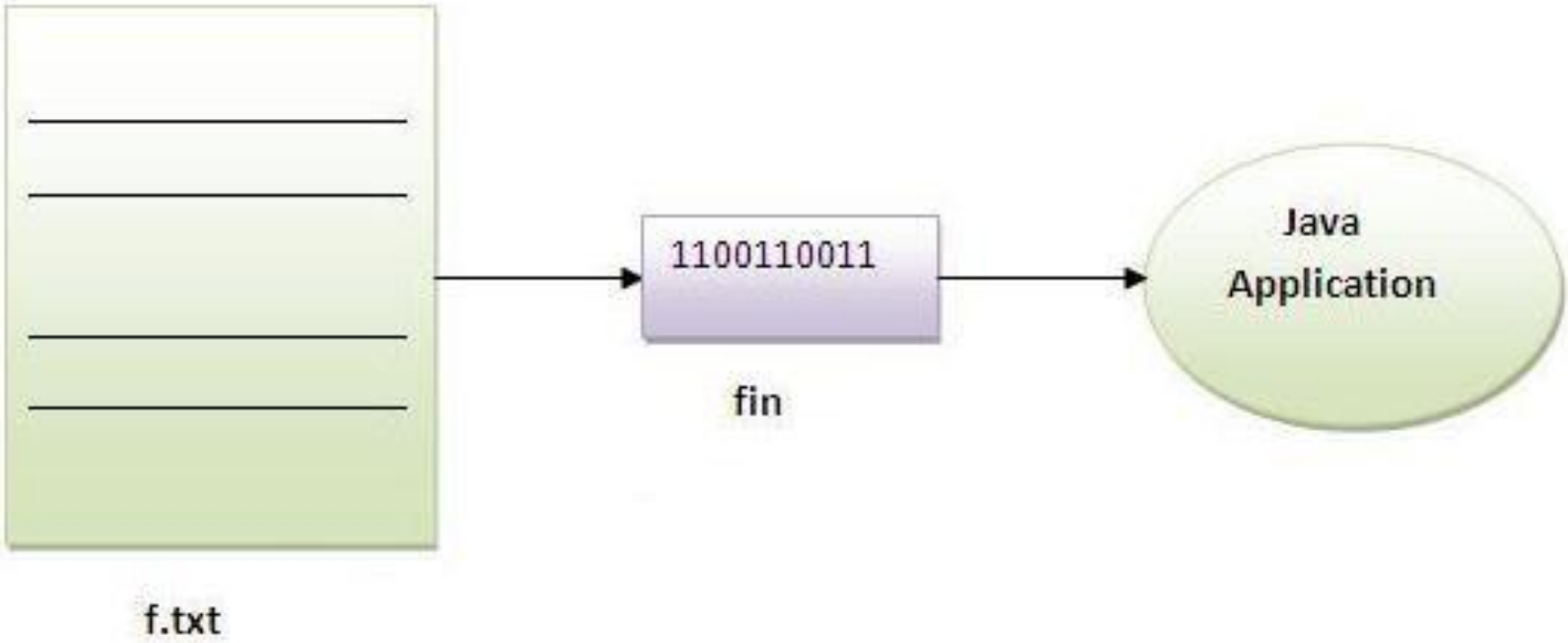

Java
Application
1100110011
fout
f.txt

Java FileInputStream class

Java FileInputStream class obtains input bytes from a file.It is used for reading streams of raw bytes such as image data. For reading streams of characters, consider using FileReader.It should be used to read byte-oriented data for example to read image, audio, video etc.

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

Output:Sachin is my favourite player.


1100110011
Java
Application
fin
f.txt

Reading the data of current java file and writing it into another file

```
import java.io.*;
class C{
public static void main(String args[])throws Exception{
FileInputStream fin=new FileInputStream("C.java");
FileOutputStream fout=new FileOutputStream("M.java");
int i=0;
while((i=fin.read())!=-1){
fout.write((byte)i);
}
fin.close();
}
}
```

We can read the data of any file using the FileInputStream class whether it is java file, image file, video file etc