JAVA I/O

Prepared by

Dr. Rajesh Kumar Ojha
Asst. Prof., CSE, Silicon University

STREAM CLASSES

Introduction

- **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.
- We can perform **file handling in java** by java IO API.

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.

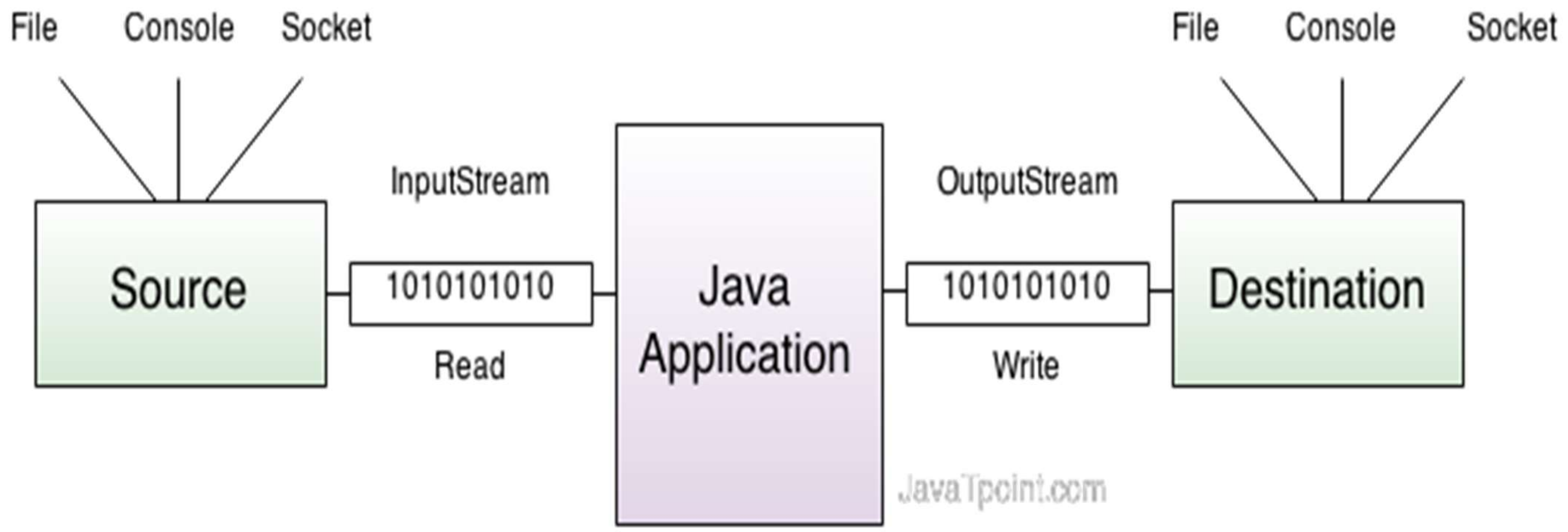
1) **System.out:** standard output stream
2) **System.in:** standard input stream
3) **System.err:** standard error stream

Output Stream

- Java application uses an output stream to write data to a destination, it may be a file,an array,peripheral device or socket.

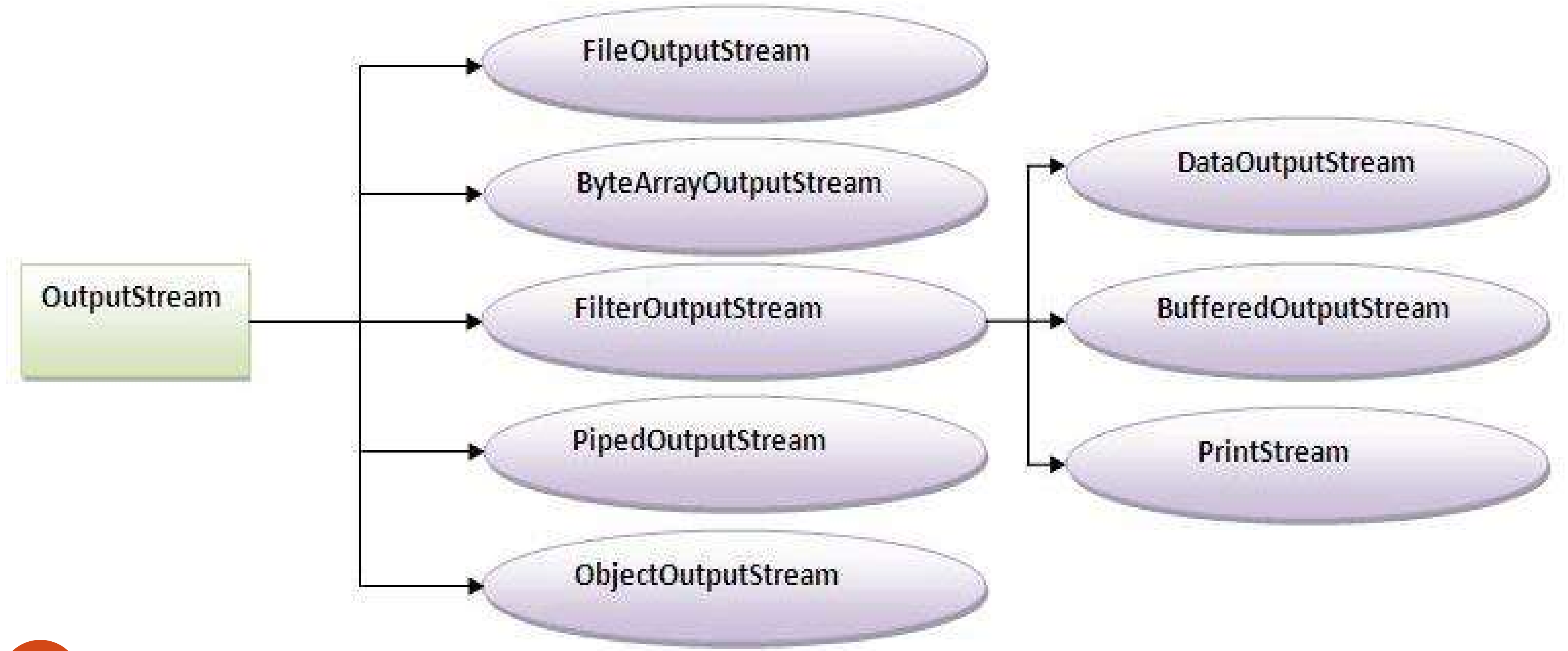
InputStream

- Java application uses an input stream to read data from a source, it may be a file,an array,peripheral device or socket.
- Let's understand working of Java OutputStream and InputStream by the figure given below.



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.

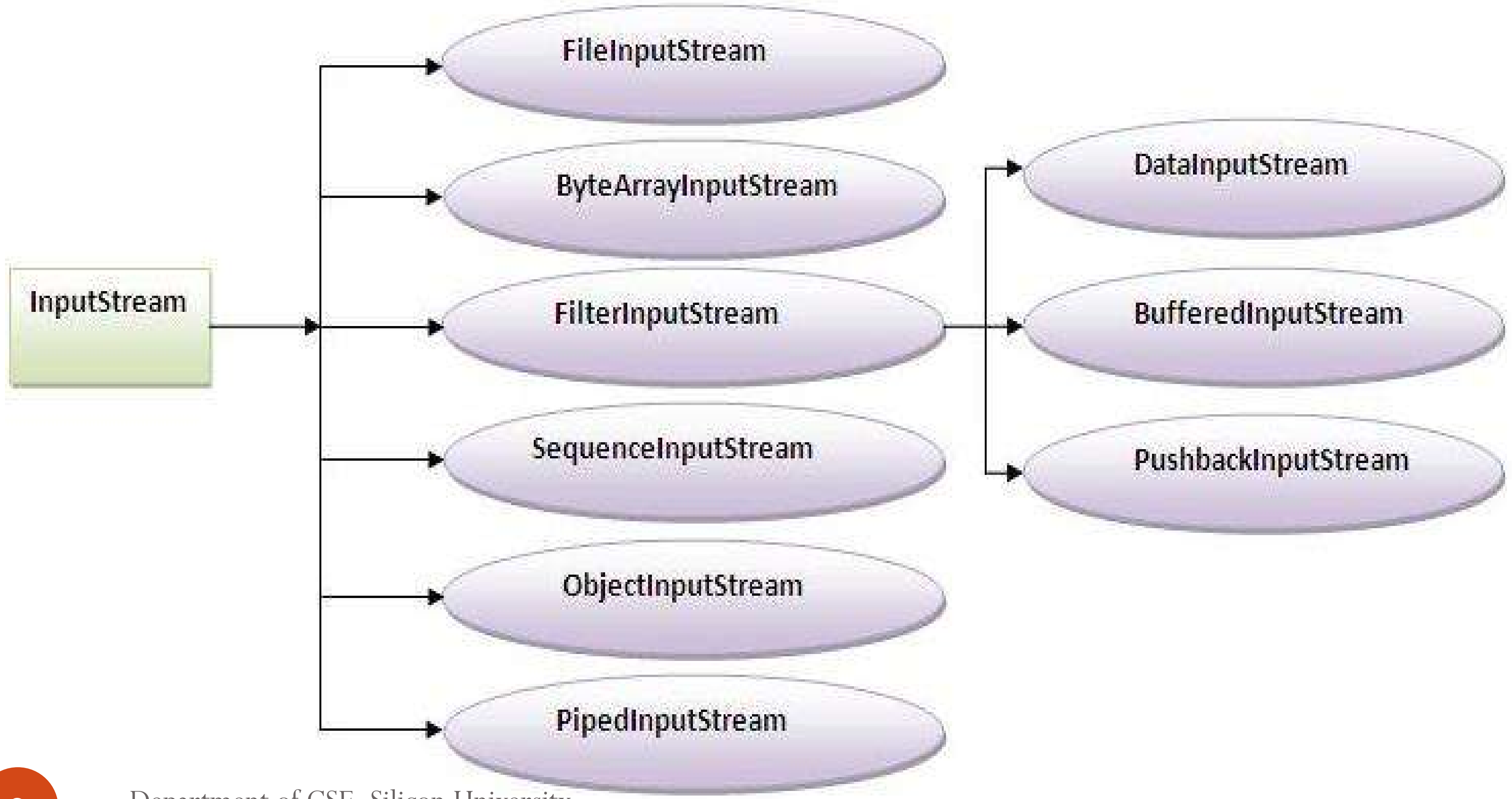


Methods of OutputStream class

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



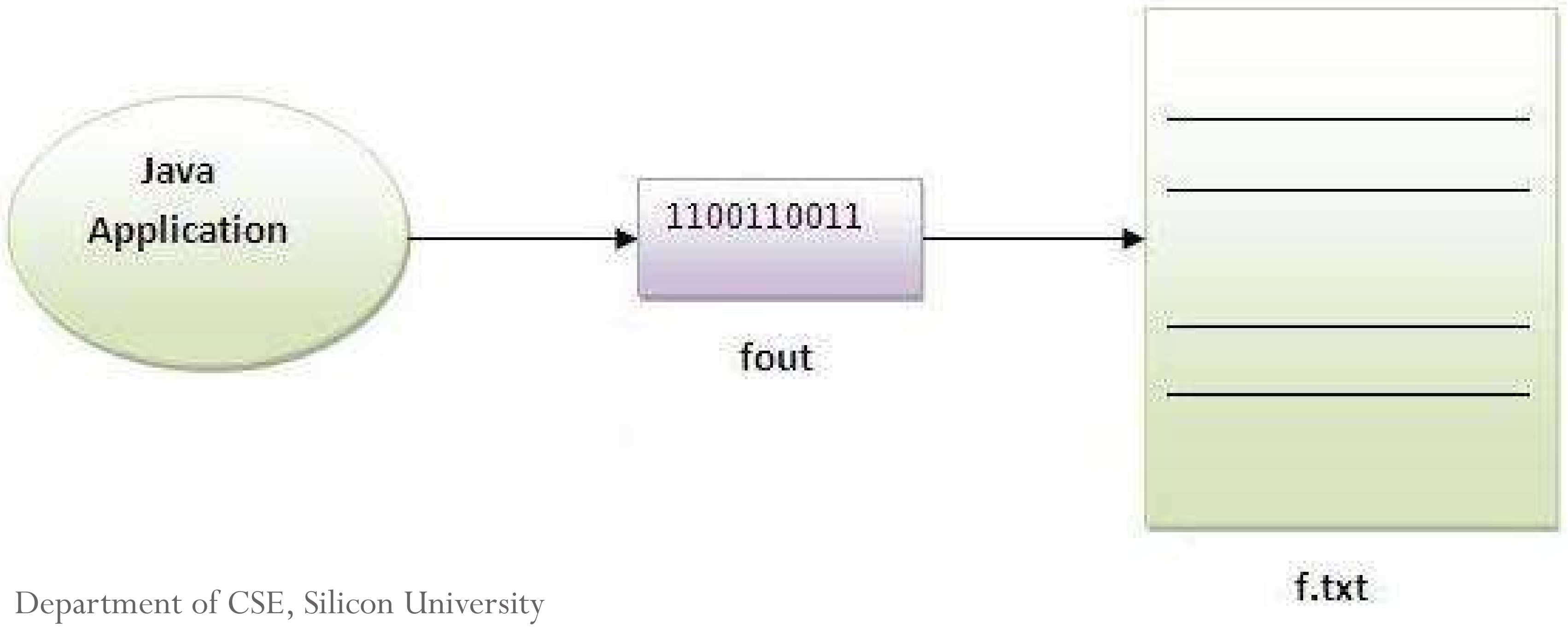
Methods of InputStream class

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

FileInputStream and FileOutputStream

FileOutputStream class

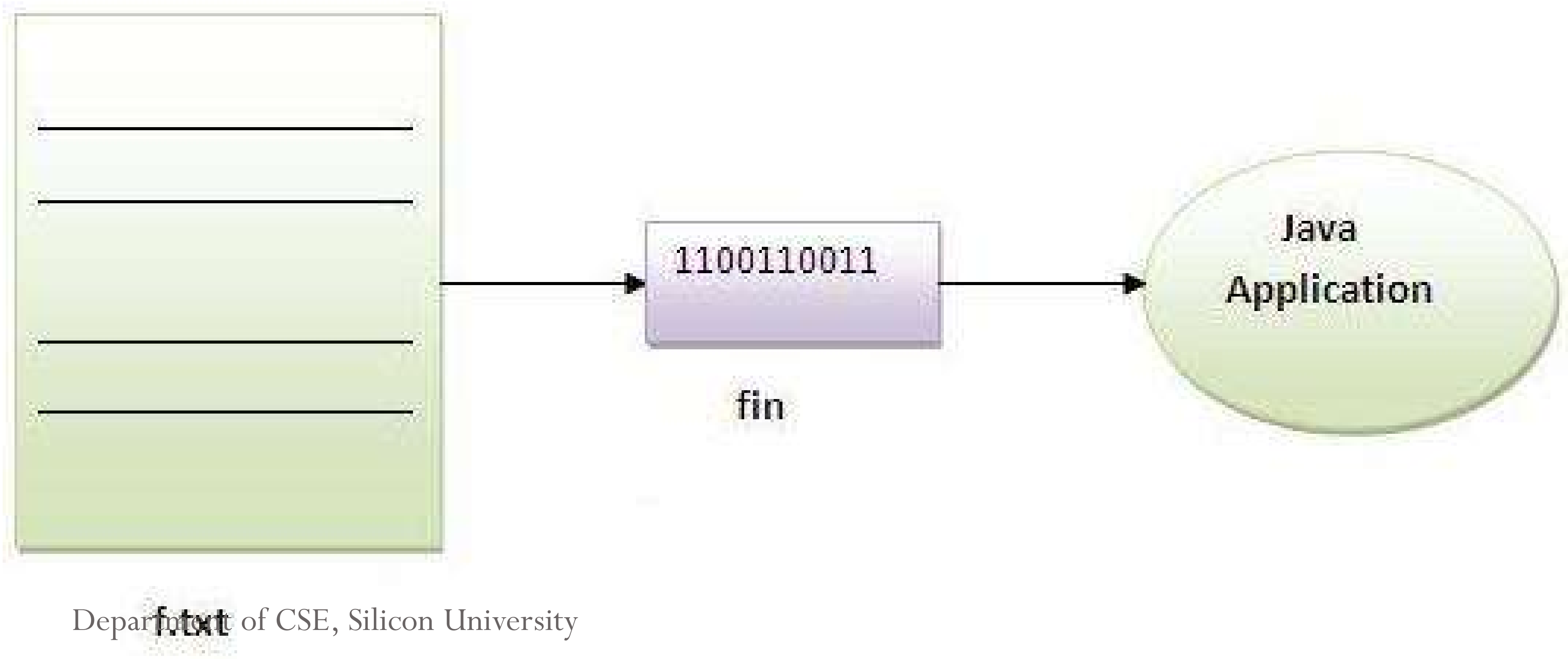
- Java FileOutputStream is an output stream for writing data to a file.
- If you have to write primitive values then use FileOutputStream. Instead, for character-oriented data, prefer FileWriter.
- But you can write byte-oriented as well as character-oriented data.



```
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);}
 }
}
O/P: success...
```

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);}
 }
}
```

O/P: Sachin is my favourite player.

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();
}
}
```

ByteArrayOutputStream class

- Java ByteArrayOutputStream class is used to write data into multiple files.
- The data is written into a byte array that can be written to multiple stream.
- The ByteArrayOutputStream holds a copy of data and forwards it to multiple streams.
- The buffer of ByteArrayOutputStream automatically grows according to data.

Constructors of ByteArrayOutputStream

Constructor	Description
ByteArrayOutputStream ()	creates a new byte array output stream with the initial capacity of 32 bytes, though its size increases if necessary.
ByteArrayOutputStream (int size)	creates a new byte array output stream, with a buffer capacity of the specified size, in bytes.

Methods of ByteArrayOutputStream class

Method	Description
public synchronized void writeTo(OutputStream out) throws IOException	writes the complete contents of this byte array output stream to the specified output stream.
public void write(byte b) throws IOException	writes byte into this stream.
public void write(byte[] b) throws IOException	writes byte array into this stream.
public void flush()	flushes this stream.
public void close()	has no affect, it doesn't closes the bytearrayoutputstream.

ByteArrayOutputStream Example

```
import java.io.*;
class S{
 public static void main(String args[])throws Exception{
  FileOutputStream fout1=new FileOutputStream("f1.txt");
  FileOutputStream fout2=new FileOutputStream("f2.txt");

  ByteArrayOutputStream bout=new ByteArrayOutputStream();
  bout.write(139);
  bout.writeTo(fout1);
  bout.writeTo(fout2);

  bout.flush();
  bout.close();//has no effect
  System.out.println("success...");
 }
}
```

```
O/P: success...
```

Thank you