SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER APPLICATIONS

23CAT606 – JAVA PROGRAMMING
I YEAR II SEM

UNIT III - NETWORKING AND I/O PACKAGES

TOPIC 5 - Input Output Packages

IO Packages

Definition: **Java I/O** (Input and Output) is used *to process the input* and *produce the output*.

1. java.io package contains all the classes required for input and output operations.
2. Perform file handling in Java by Java I/O API.

Stream

1. A stream is a sequence of data.
2. In Java, a stream is composed of bytes.

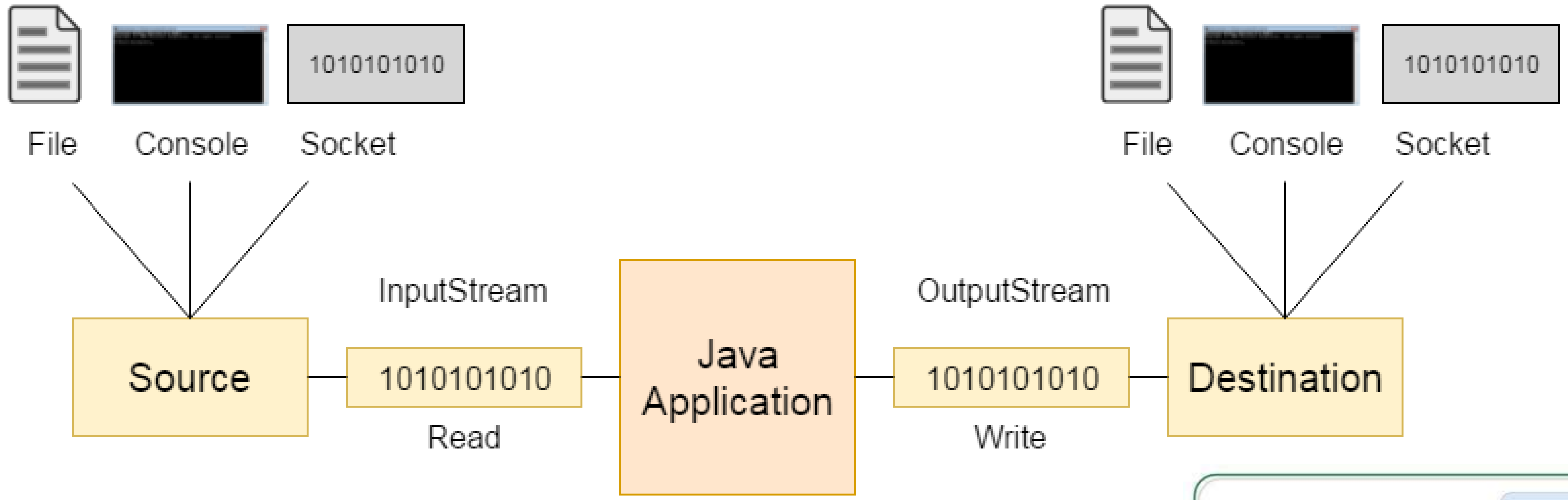
Types

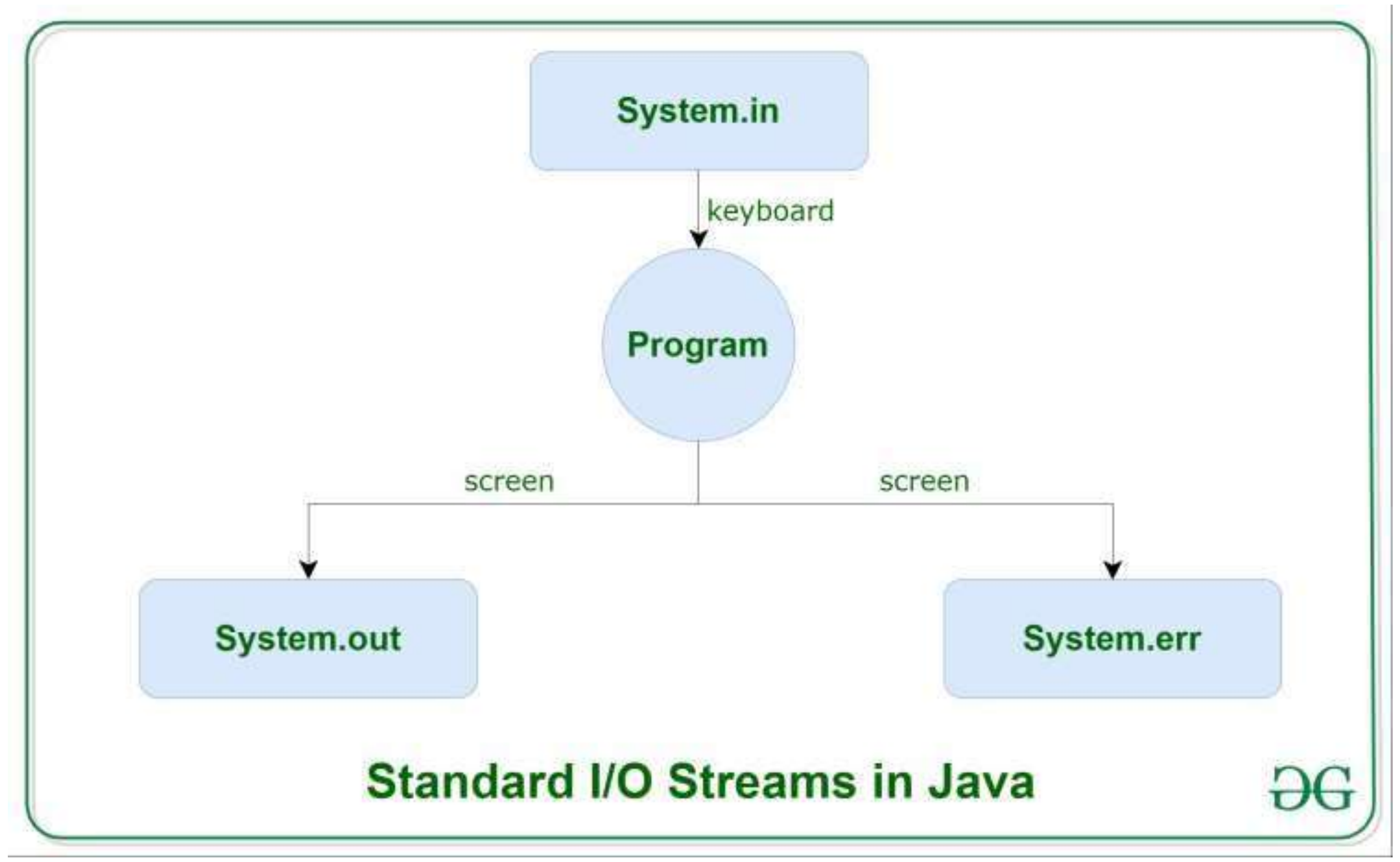
1) **System.out:** standard output stream
2) **System.in:** standard input stream
3) **System.err:** standard error stream

Example

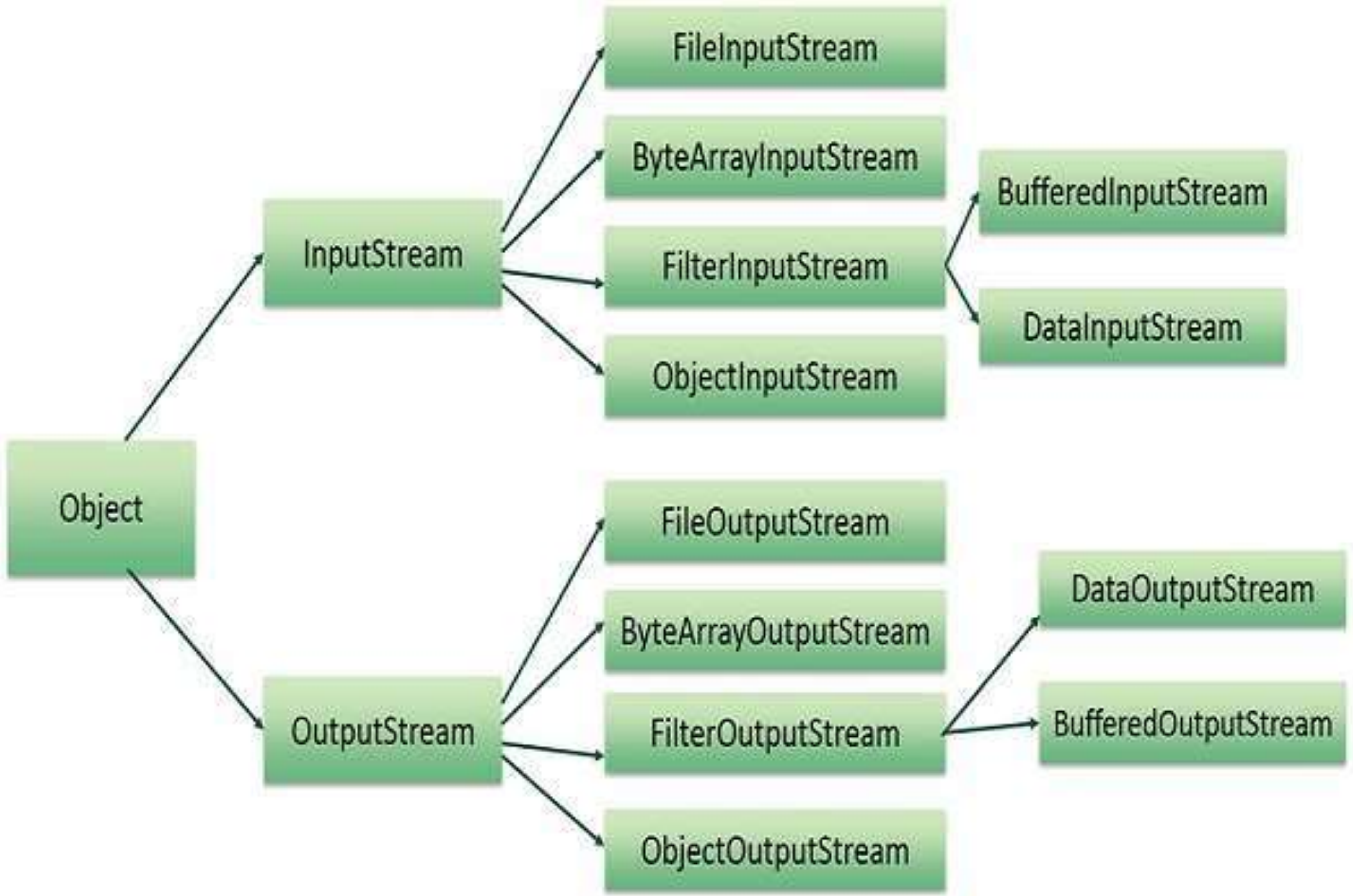
```
System.out.println("simple message");
System.err.println("error message");
int i=System.in.read();
```

IO Packages


File
Console
Socket
1010101010
File
Console
Socket
1010101010
InputStream
OutputStream
Source
1010101010
Java Application
1010101010
Destination
Read
Write


System.in
keyboard
Program
screen
screen
System.out
System.err
Standard I/O Streams in Java

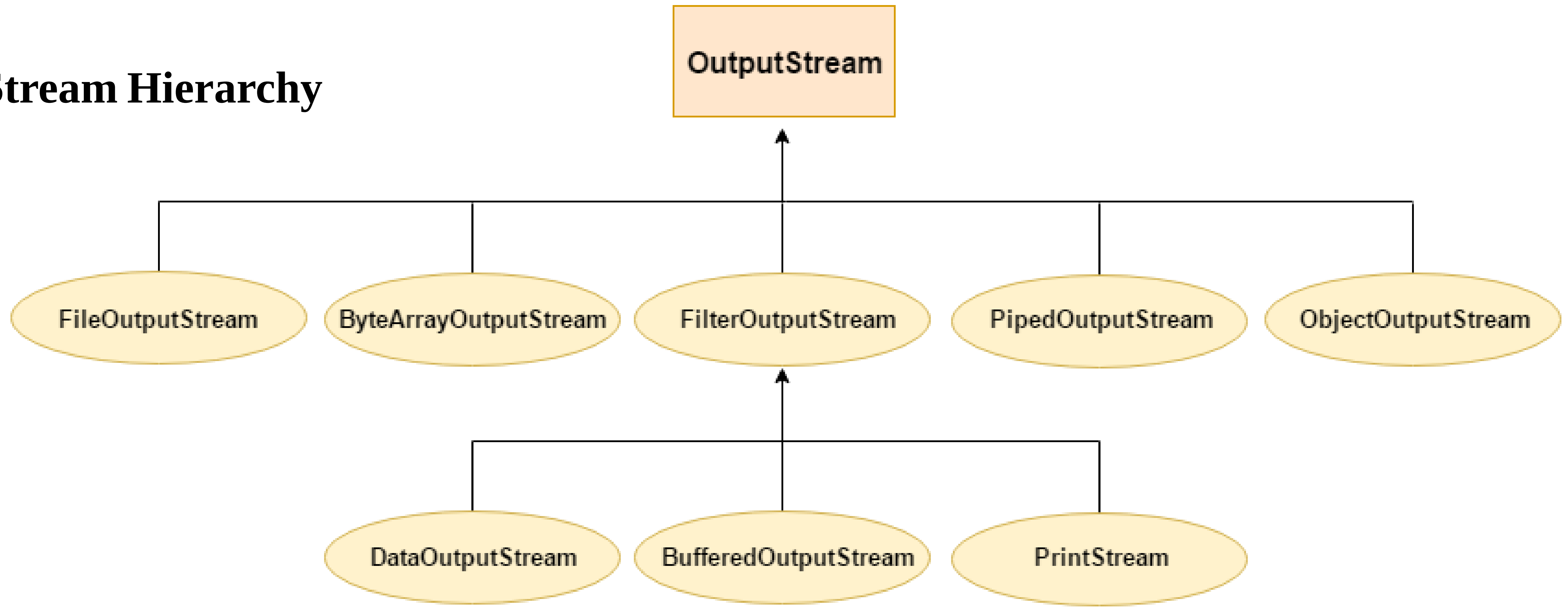
Input/OutputStream



OutputStream class

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.

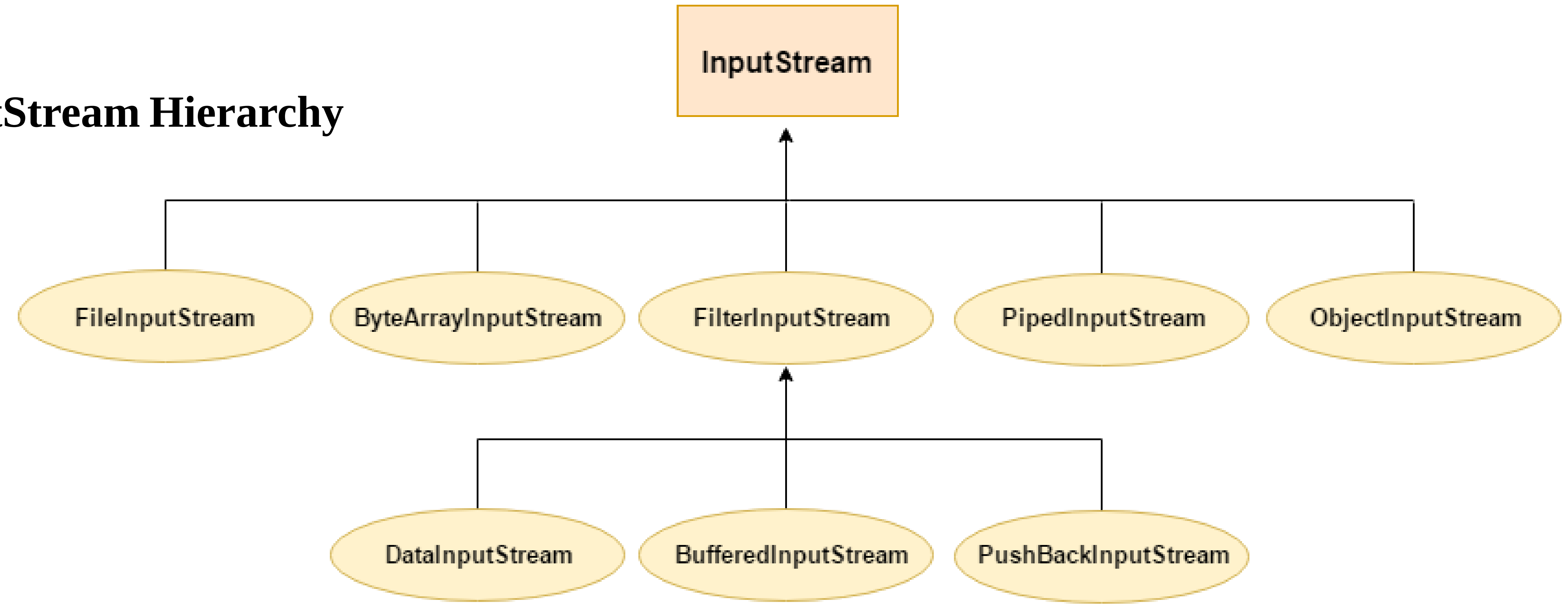
OutputStream Hierarchy



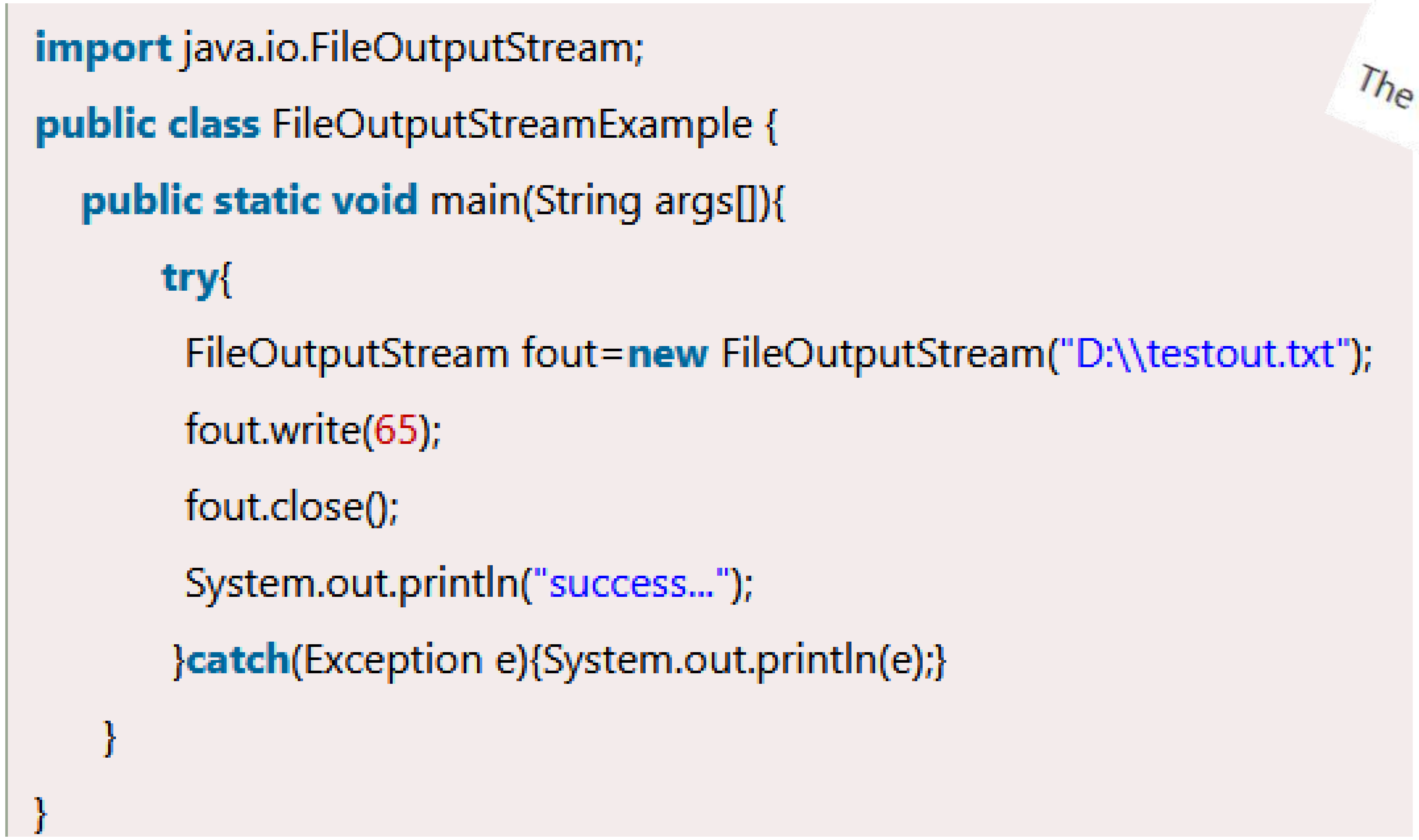
InputStream class

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 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.

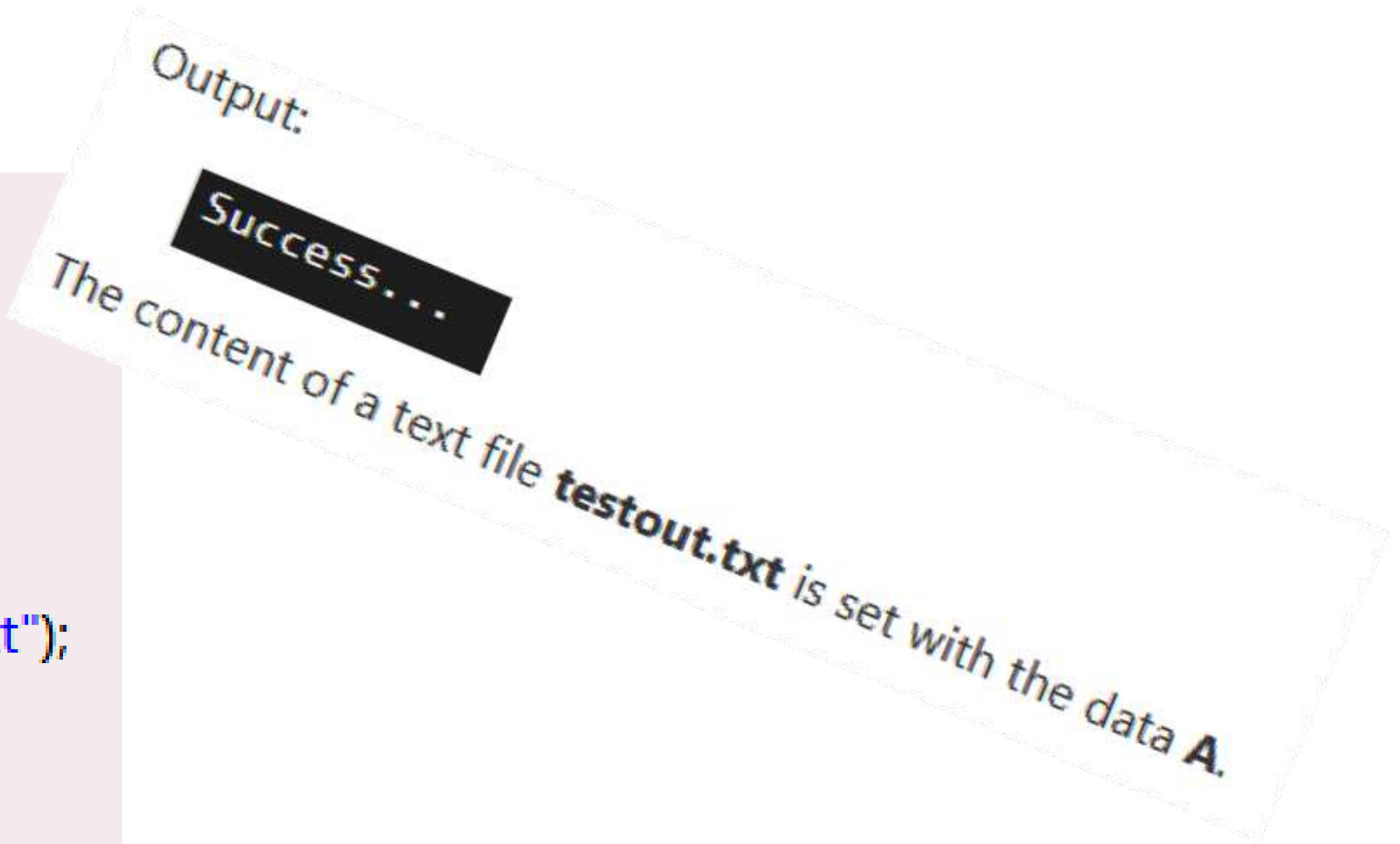
InputStream Hierarchy



Example: Java FileOutputStream Example 1: write byte



```
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);}
    }
}
```



Output:

```
Success...
```

The content of a text file **testout.txt** is set with the data **A**.

Java FileOutputStream example 2: write string

```
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);}
    }
}
```

Output:

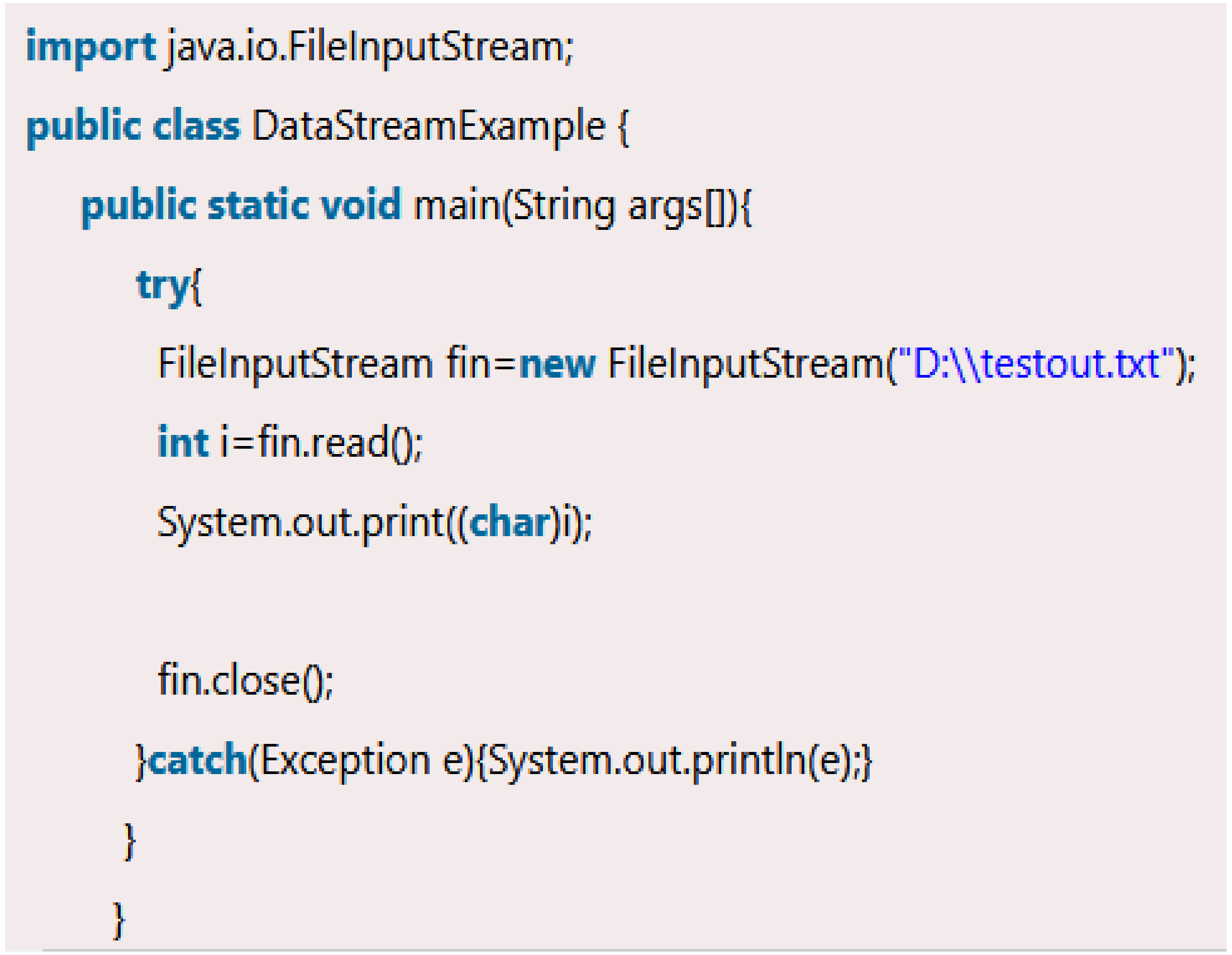
```
Success...
```

The content of a text file **testout.txt** is set with the data **Welcome to javaTpoint.**

testout.txt

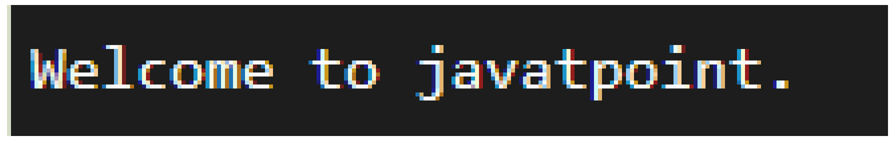
```
Welcome to javaTpoint.
```

Java FileInputStream example 1: read single character



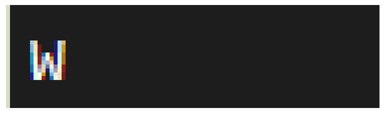
```java
import java.io.FileInputStream;
public class DataStreamExample {
    public static void main(String args[]){
        try{
          FileInputStream fin=new FileInputStream("D:\\testout.txt");
          int i=fin.read();
          System.out.print((char)i);

          fin.close();
        }catch(Exception e){System.out.println(e);}
       }
      }
```

a text file named as **"testout.txt"** is required to be created.



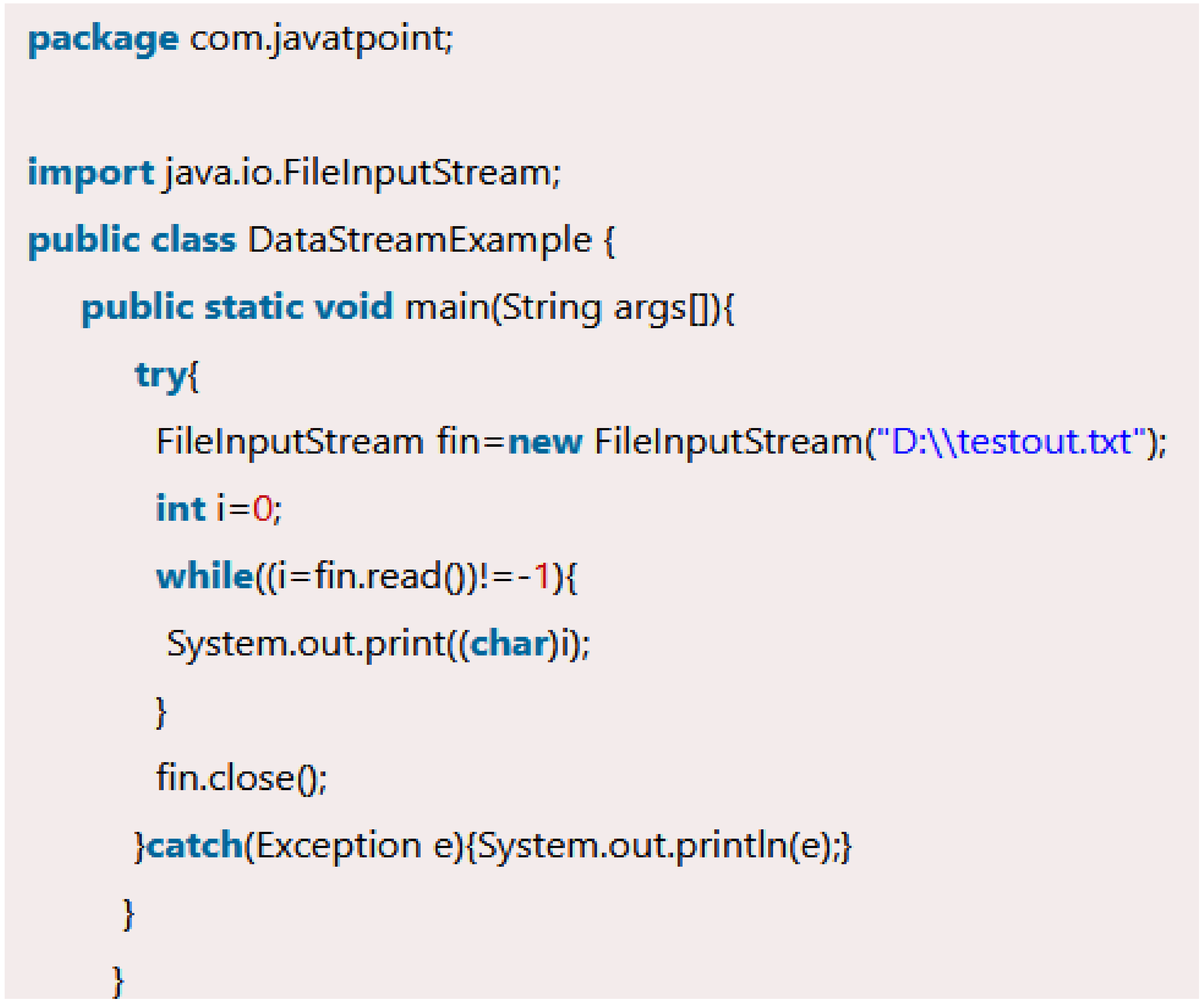
```
Welcome to javatpoint.
```

Output:



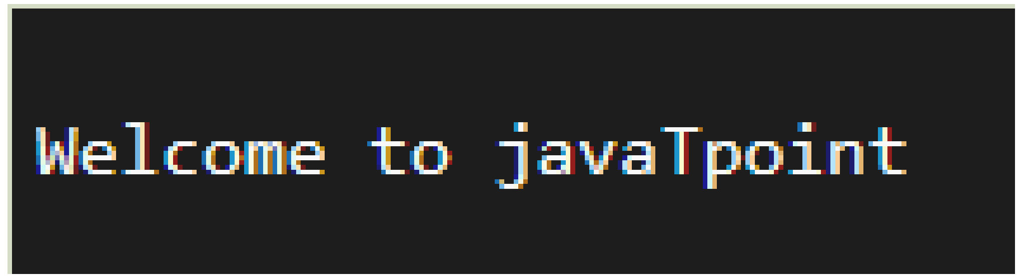
```
W
```

Java FileInputStream example 2: read all characters



```java
package com.javatpoint;

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);}
         }
        }
```

Output:



```
Welcome to javaTpoint
```

Thank you