Core Java

Lesson 10: The Input and Output Classes

Capgemini

Lesson Objectives

- After completing this lesson, participants will be able to
  - Understand concept of Java I/O API
  - Implements byte and character streams to perform I/O
  - NIO Overview

Overview

- Most programs need to access external data.
- Data is retrieved from an input source. Program results are sent to output destination.

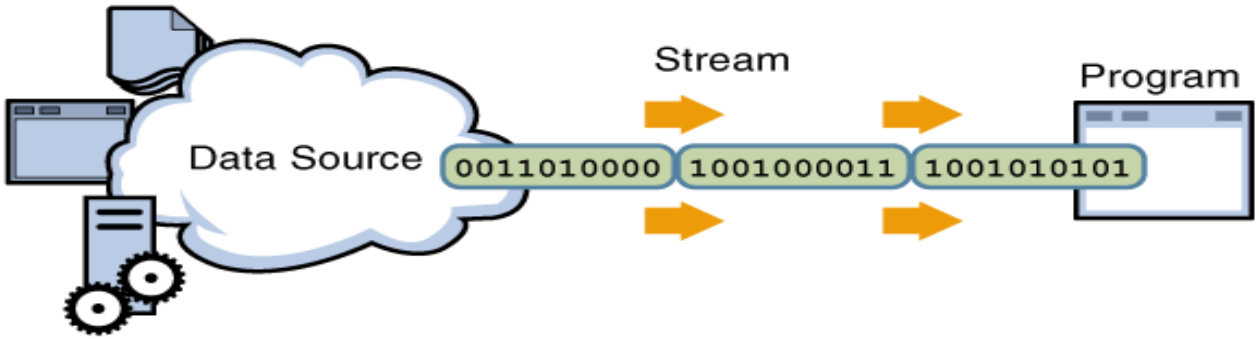


Figure 7-1: A program uses an input stream to read data from a source, one item at a time

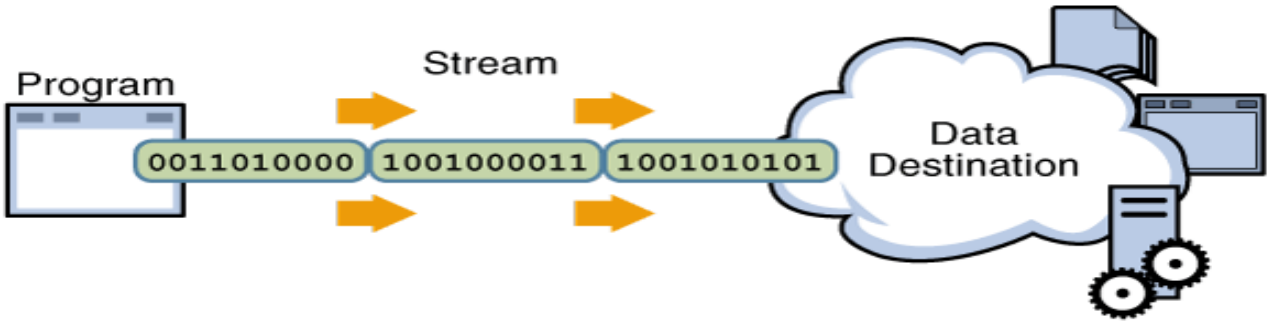


Figure 7-2:A program uses an output stream to write data to a destination, one item at time

What is a Stream?

- Stream:
  - Abstraction that consumes or produces information.
  - Linked to source and destination.
  - Implemented within class hierarchies defined in java.io package.
  - An input stream acts as a source of data.
  - An output stream acts as a destination of data.

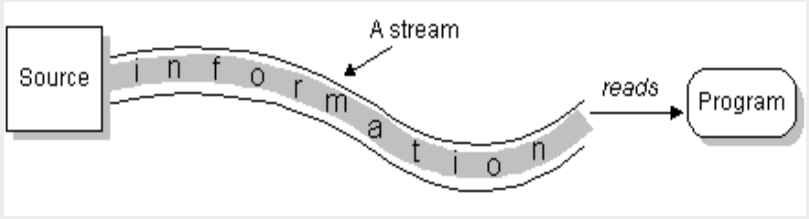


Figure 7-3: (a) Input Stream

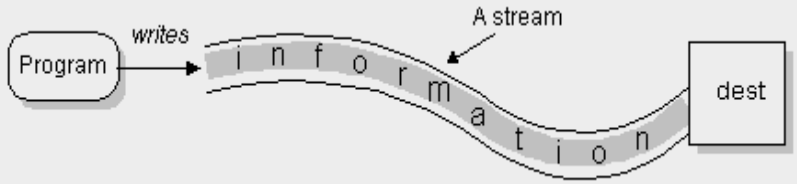


Figure 7-3:(b) Output stream

Different Types of I/O Streams

- Byte Streams: Handle I/O of raw binary data.
- Character Streams: Handle I/O of character data. Automatic translation handling to and from a local character.
- Buffered Streams: Optimize input and output with reduced number of calls to the native API.
- Data Streams: Handle binary I/O of primitive data type and String values.
- Object Streams: Handle binary I/O of objects.
- Scanning and Formatting: Allows a program to read and write formatted text.

Byte Stream I/O Hierarchy

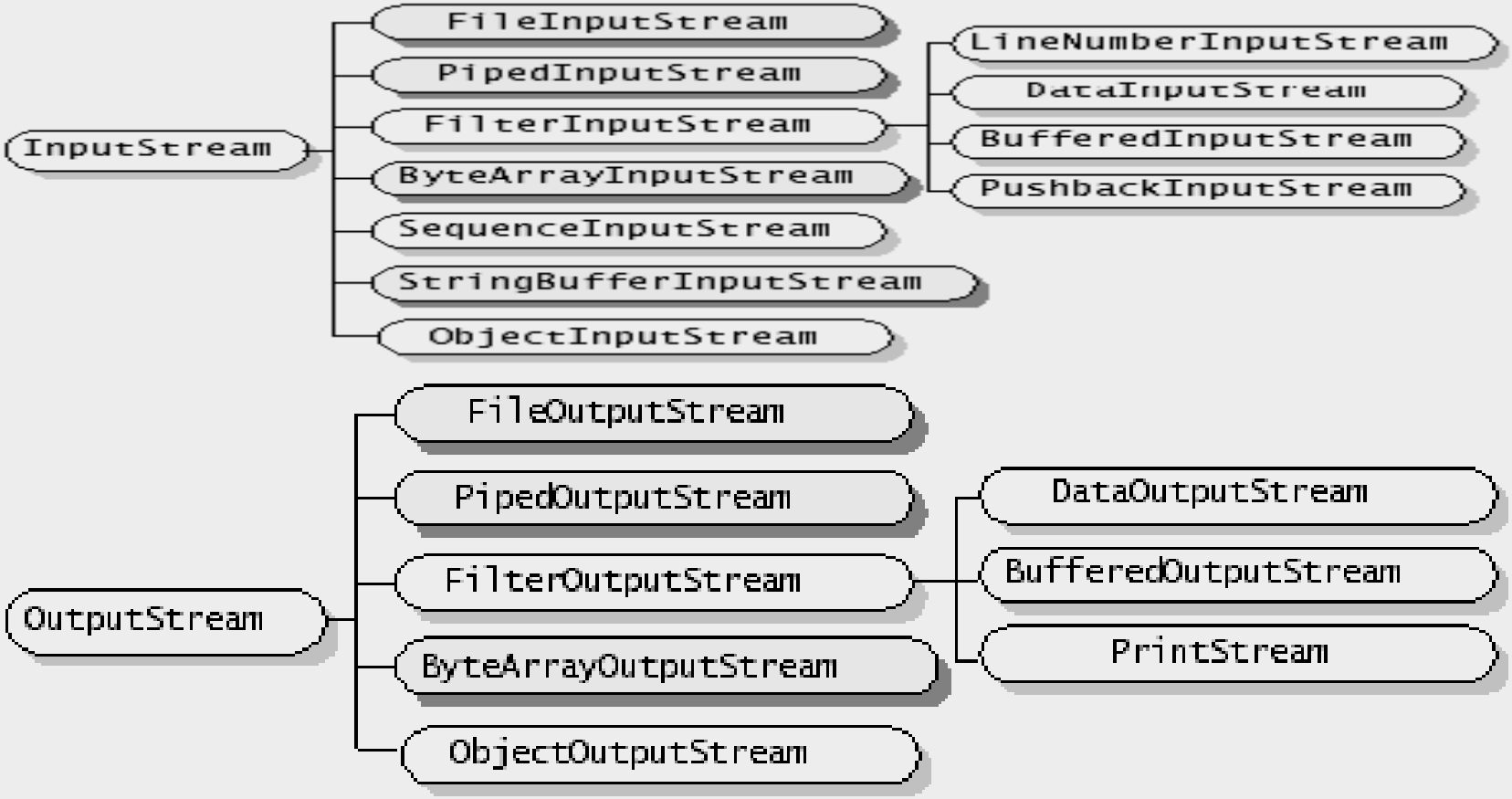


Figure 7-4:Byte-stream I/O hierarchy

Methods of InputStream Class

Method	**Description**
close()	Closes this input stream and releases any system resources associated with the stream.
int read()	Reads the next byte of data from the input stream.
int read(byte[] b)	Reads some number of bytes from the input stream and stores them into the buffer array *b*.
int read(byte[] b, int off, int len)	Reads up to *len* bytes of data from the input stream into an array of bytes.

Table 7-1: Methods of class InputStream

Methods of OutputStream Class

Method	Description
close()	Closes this output stream and releases any system resources associated with this stream.
flush()	Flushes this output stream and forces any buffered output bytes to be written out.
write(byte[] b)	Writes *b.length* bytes from the specified byte array to this output stream.
write(byte[] b, int off, int len)	Writes *len* bytes from the specified byte array starting at offset off to this output stream.
write(int b)	Writes the specified byte to this output stream.

Table 7-2: Methods of class OutputStream

Input Stream Subclasses

Classname	Description
DataInputStream	A filter that allows the binary representation of java primitive values to be read from an underlying inputstream
BufferedInputStream	A filter that buffers the bytes read from an underlying input stream. The buffer size can be specified optionally.
FilterInputStream	Superclass of all input stream filters. An input filter must be chained to an underlying inputstream.
ByteArrayInputStream	Data is read from a byte array that must be specified
FileInputStream	Data is read as bytes from a file. The file acting as the input stream can be specified by File object, or as a String
PushBackInputStream	A filter that allows bytes to be "unread " from an underlying stream. The number of bytes to be unread can be optionally specified.
ObjectInputStream	Allows binary representation of java objects and java primitives to be read from a specified inputstream.
PipedInputStream	It reads many bytes from PipedOutputStream to which it must be connected.
SequenceInputStream	Allows bytes to be read sequentially from two or more input streams consecutively.

The predefined streams

- The java.lang.System class encapsulates several aspects of the run-time environment.
- Contains three predefined stream variables: in, out & err.
- These fields are declared as public and static within System.
  - *System.out* :refers to the standard output stream
  - *System.err* :refers to standard error stream
  - *System.in* : refers to standard input

Example : Reading Console input

```
import java.io.*;
class ReadKeys {
 public static void main (String args[]) {
       StringBuffer sb = new StringBuffer();
       char c;
     System.out.println("Enter a String:");
       try {
            while((c =(char)System.in.read()) != '\n')
                 sb.append(c);
      }catch(Exception e){
           System.out.println(“Error while reading” +
e.getMessage()); }
  String s = new String(sb);
  System.out.println("You entered : " + s);      }}
```

Example: FileInputStream & FileOutputStream

```
class CopyFile  {
      FileInputStream fromFile;  FileOutputStream toFile;
      public void init(String arg1, String arg2) {       //pass file names
            try{
               fromFile = new FileInputStream(arg1);
               toFile = new FileOutputStream(arg2);
            } catch (Exception fnfe) {…}
     }
    public void copyContents() {// copy bytes
      try {
            int i = fromFile.read();
            while ( i != -1) {                              //check the end of file
                  toFile.write(i);
                  i = fromFile.read(); }
     } catch (IOException ioe) { System.out.println("Exception: " + ioe);}
  }
```

Demo : FileInput/OutputStream

- Execute:
  - ReadKeys.java
  - CopyFile.java program

Character Stream Hierarchy

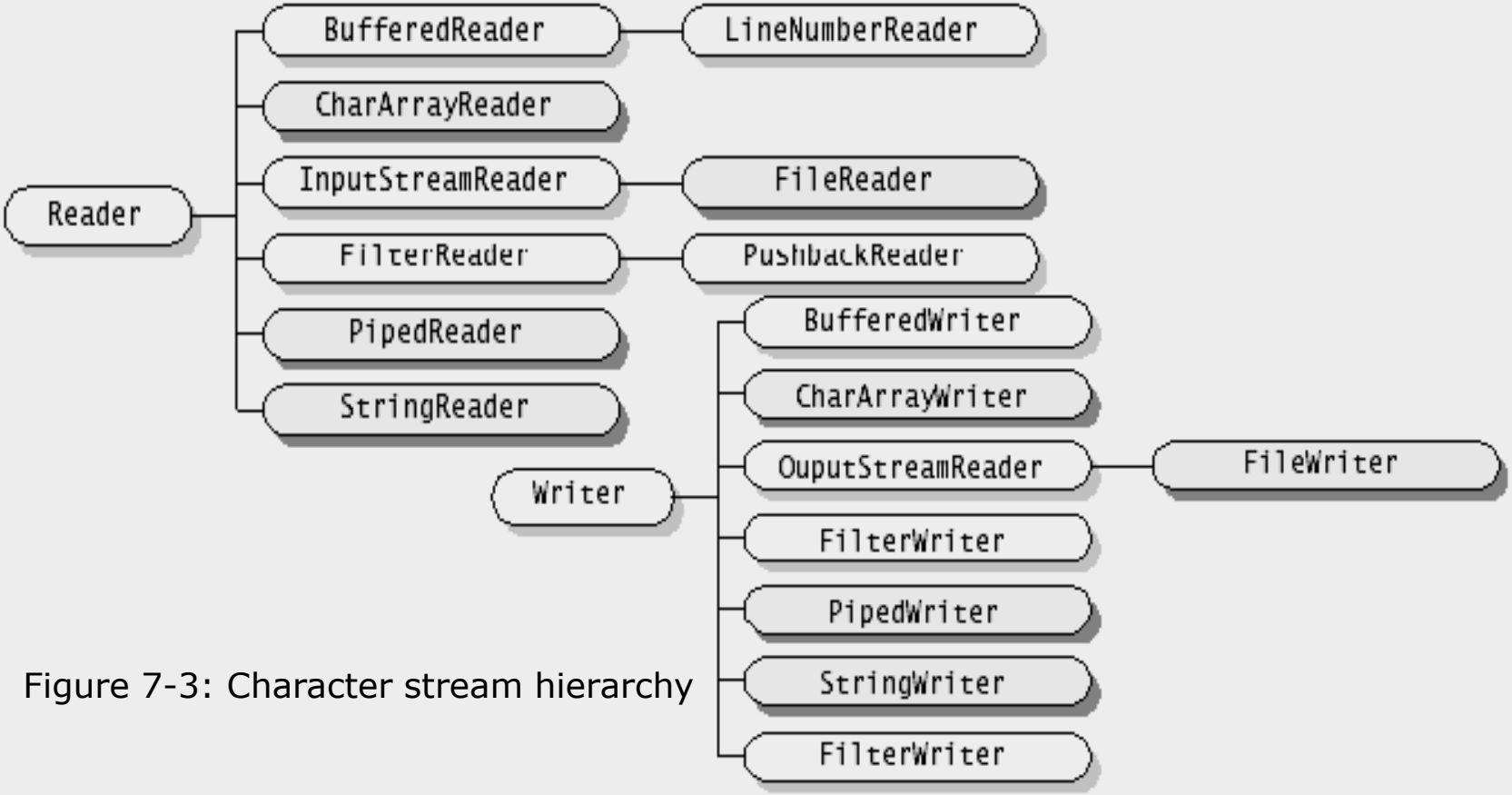


Figure 7-3: Character stream hierarchy

Reader Class Methods

Method	**Description**
int read() throws IOException	reads a byte and returns as an int
int read(char b[])throws IOException	reads into an array of chars b
int read(char b[], int off, int len) throws IOException	reads *len* number of characters into char array *b*, starting from offset *off*
long skip(long n) throws IOException	Can skip n characters.

Table 7-4: Reader Methods

Writer Class Methods

Method	Description
void write(int c)throws IOException	writes a byte.
void write(char b[])throws IOException	writes from an array of chars b
void write(char b[], int off, int len) throws IOException	writes len number of characters from char array b, starting from offset off
void write(String b, int off, int len) throws IOException	writes len number of characters from string b, starting from offset off

Table 7-5: Writer Methods

Example: FileReader, FileWriter Classes

```
public class CopyCharacters {
    public static void main(String[] args) throws IOException {
 try(FileReader inputStream = new FileReader("sampleinput.txt");
        FileWriter outputStream = newFileWriter("sampleoutput.txt"))
{
            int c;
            while ((c = inputStream.read()) != -1) {
                outputStream.write(c);
            }
  } catch(IOException ex) {
              System.out.println(ex.getMessage());
   }
} }
```

PrintWriter

- The **Java.io.PrintWriter** class prints formatted representations of objects to a text-output stream.
- This class implements all of the print methods found in PrintStream. It does not contain methods for writing raw bytes, for which a program should use unencoded byte streams.
- Unlike the PrintStream class, if automatic flushing is enabled it will be done only when one of the println() methods is invoked, rather than whenever a newline character happens to be output. The println() methods use the platform's own notion of line separator rather than the newline character.

NIO Overview

- NIO is widely used in file handling.
- NIO was developed to allow Java programmers to implement high-speed I/O without using the custom native code
- NIO moves the time-taking I/O activities like filling, namely and draining buffers, etc back into the operating system, thus allows for great increase in operational speed

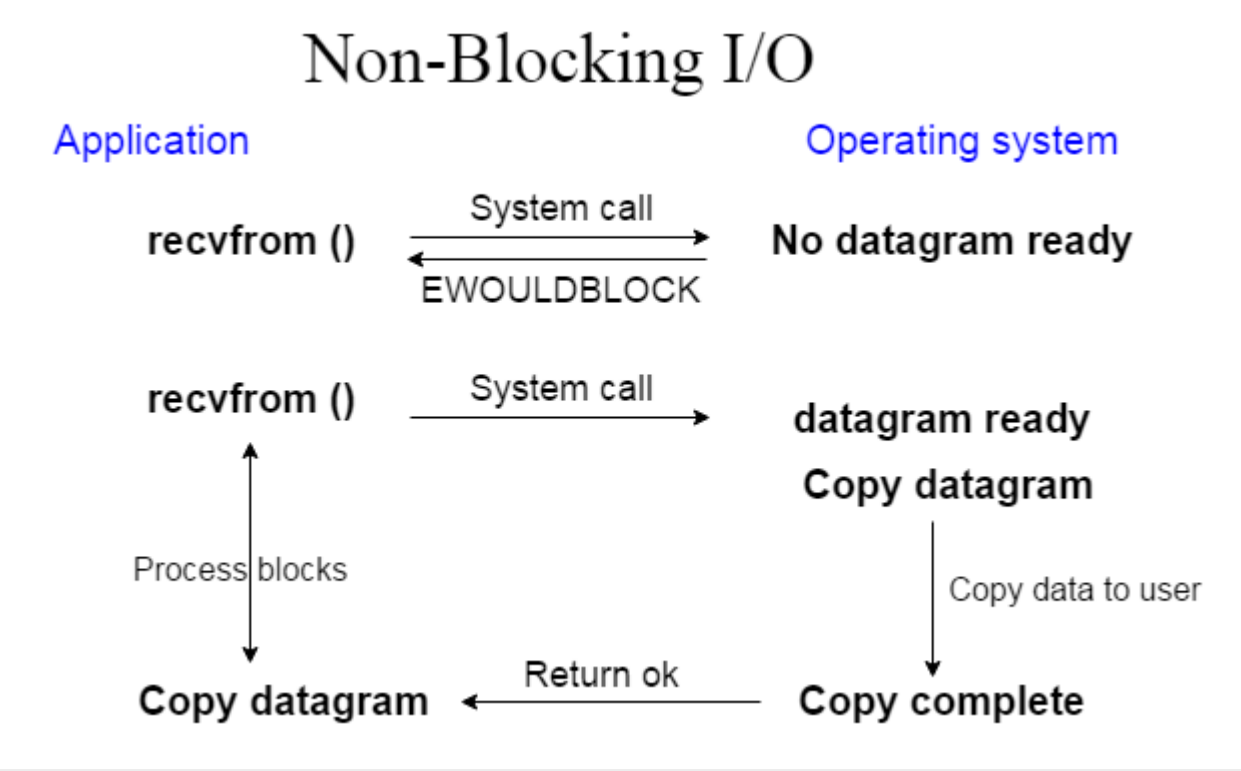
➢ **Java NIO fundamental components are given below:**

NIO Overview

- **Channels and Buffers:** In standard I/O API the character streams and byte streams are used. In NIO we work with channels and buffers. Data is always written from a buffer to a channel and read from a channel to a buffer.
- **Selectors:** Java NIO provides the concept of "selectors". It is an object that can be used for monitoring the multiple channels for events like data arrived, connection opened etc. Therefore single thread can monitor the multiple channels for data.
- **Non-blocking I/O:** Java NIO provides the feature of Non-blocking I/O. Here the application returns immediately whatever the data available and application should have pooling mechanism to find out when more data is ready.

NIO Overview

- Application and Operating system interface for Non-blocking I/O:



Input Output Classes

- Lab 7:Input Output Classes

Summary

- In this lesson you have learnt:
  - Different types of I/O Streams supported by Java
  - Important classes in java.io package
  - NIO overview

Review Question

- Question 1: What is a buffer?
  - **Option 1 :** Section of memory used as a staging area for input or output data.
  - **Option 2 :** Cable that connects a data source to the bus.
  - **Option 3 :** Any stream that deals with character IO.
  - **Option 4 :** A file that contains binary data.
- Question 2: Can data flow through a given stream in both directions?
  - True
  - False
- Question 3: _____ is the name of the abstract base class for streams dealing with *character input*