UNIT 9 : I/O IN JAVA

UNIT STRUCTURE

9.1 Learning Objectives

9.2 Introduction

9.3 I/O Basics: Streams

9.4 The Stream Classes

9.5 The Predefined Streams

9.6 Reading Console Input

9.7 Writing Console Output

9.8 Reading and Writing Files

9.9 Let Us Sum Up

9.10 Further Readings

9.11 Answers to Check Your Progress

9.12 Model Questions

9.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- describe I/O packages in Java
- define predefined streams
- use console input and output
- describe how to read and write data to and from files

9.2 INTRODUCTION

In this unit, we will learn one of the most important Java I/o. package We will learn about streams and different stream classes. Besides this how to read data from input and how to write data into output devices. We also give a brief introduction of file handling in Java. In the next unit, we will learn about Java applets.

The I/O package supports Java's basic I/O (Input/Output) system including file I/O. Java program perform I/O through streams. A stream is linked to a physical device by the Java I/O system. Java define two types of

stream, *byte* and *character*. In Java 1.0, the only way to perform console input was to use a byte stream. The preferred method of reading console input for Java 2 is use a character oriented stream, which makes the program easier to internationalize and maintain.

Java provides a number of classes and methods that allow to read and write files. In Java all files are byte oriented, and Java provides method to read and write bytes from and to a file.

9.3 I/O BASICS : STREAMS

A stream can be defined as a sequence of data. There are two kinds of Streams.

InputStream - The InputStream is used to read data from a source.

OutputStream - The OutputStream is used for writing data to a destination.

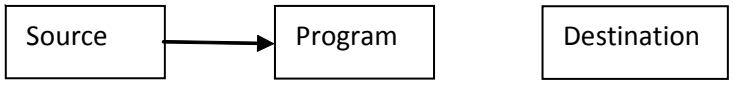


Figure 9.1:Data flow in Java Program

Java views each file as a sequential stream of bytes. Each operating system provides a mechanism to determine the end of a file, such as an end-of-file marker or count of the total bytes in the file that is recorded in a system-maintained administrative data structure.

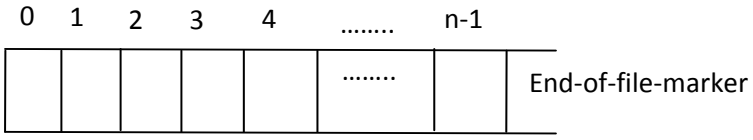


Figure 9.2: Java's view of a file

A Java program process a stream of bytes after receiving an indication from the operating system that the end of the stream reaches. The program does not need to know how the underlying platform represents files or streams.Now, we will discuss some notion of streams.

Byte streams: Programs use byte streams to perform input and output of 8-bit bytes.

Character streams: Java character streams are used to perform input and output for 16-bit unicode.

Standard Streams: All the programming languages provide support for standard I/O, where the user's program can take input from keyboard and then produce an output on the computer screen. Java provides the following three standard streams

Standard Input " This is used to input the data to user's program, a keyboard is used as standard input stream and represented as System.in.

Standard Output " This is used to output the data produced by the user's program and a computer screen is used for standard output stream and represented as System.out.

Standard error " This is used to output the error data produced by the user's program and generally a computer screen is used for standard error stream and represented as System.err

Working of Files and Streams- Java uses streams to handle I/O operations through which the data is followed from one location to another. For example, an InputStream can flow the data from a disk file to the internal memory and an OutputStream can flow the data from the internal memory to a disk file. The disk-file may be a text file or a binary file. When we work with a text file, we use a character stream where one character is treated as per byte on disk. When we work with a binary file, we use a binary stream.

The working process of the I/O streams can be shown in figure 9.3 below.

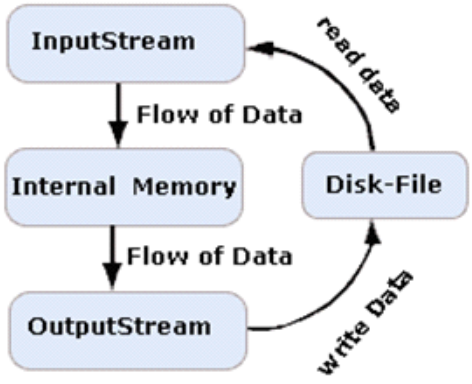


Figure 9.3:Input-Output Stream

9.4 THE STREAM CLASSES

The Java ***Input/Output (I/O)*** is a part of ***java.io*** package. The java.io package contains a relatively large number of classes that support input and output operations. The classes in the package are primarily abstract classes and stream-oriented that define methods and subclasses which allow bytes to be read from and written to files or other input and output sources. The InputStream and OutputStream are central classes in the package which are used for reading from and writing to byte streams, respectively.To use the stream classes, it is required to import java.io.

The ***java.io*** package can be categorized along with its stream classes in a hierarchy structure as shown in Fig 9.4.

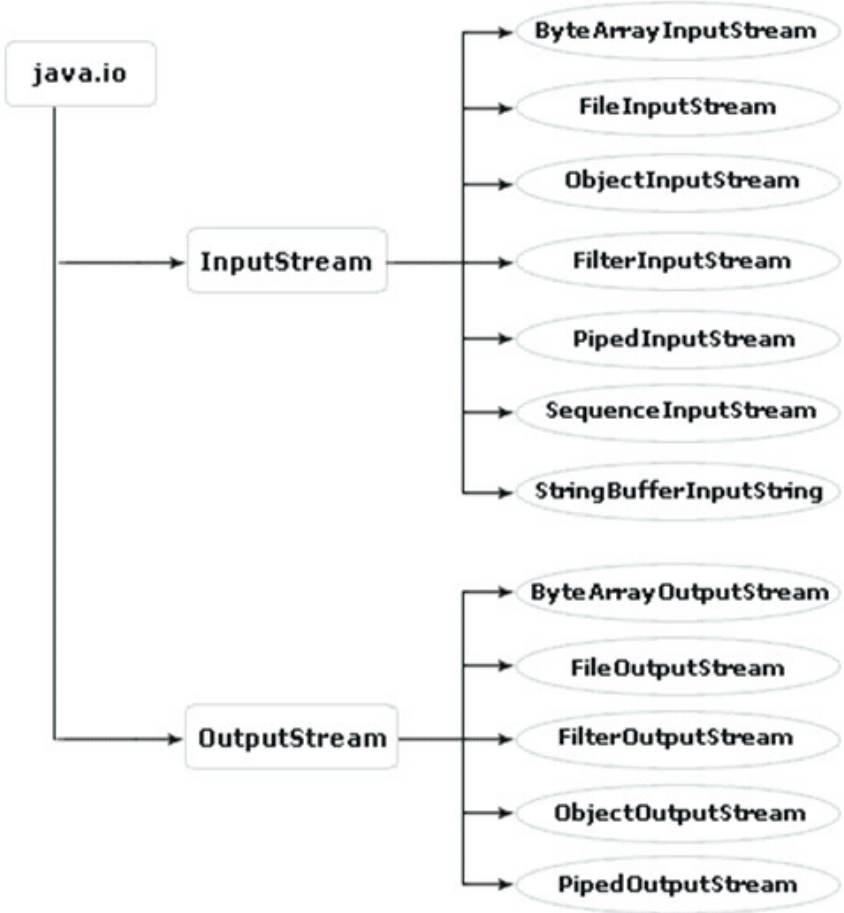


Figure 9.4: Stream Classes

The ***InputStream*** class is used for reading the data such as a byte and array of bytes from an input source. An input source can be a file, a string, or memory that may contain the data. It is an abstract class that defines the programming interface for all input streams that are inherited from it. An input stream is automatically opened when we create it. We can explicitly close a stream with the *close( )* method, or let it be closed implicitly when the object is found as a garbage.

The subclasses inherited from the ***InputStream*** class can be seen in a hierarchy manner as shown below in figure 9.5.

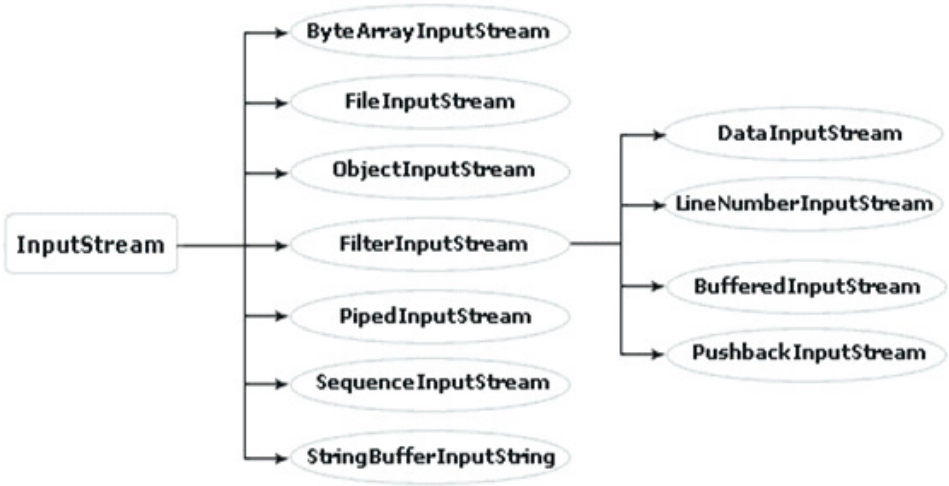


Figure 9.5: Input Stream Classes

InputStream is inherited from the ***Object*** class. Each class of the InputStreams provided by the java.io package.

The ***OutputStream*** class is a sibling to InputStream that is used for writing byte and array of bytes to an output source. Similar to input sources, an output source can be anything such as a file, a string, or memory containing the data. Like an input stream, an output stream is automatically opened when you create it. You can explicitly close an output stream with the close() method, or let it be closed implicitly when the object is garbage collected.

The classes inherited from the OutputStream class can be seen in a hierarchy structure as shown in Fig 9.6.

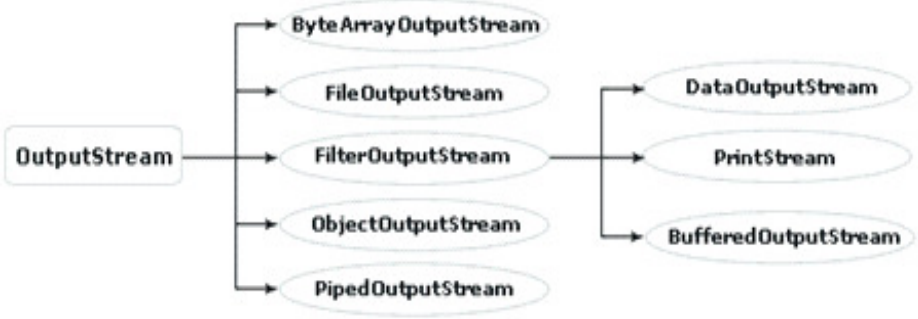


Figure 9.6: Output Stream Classes

OutputStream is also inherited from the Object class. Each class of the OutputStreams provided by the java.io package.

There are many byte stream classes. To demonstrate how byte streams work, we'll focus on the file I/O byte streams, FileInputStream and FileOutputStream. Other kinds of byte streams are used in much the same way, they differ mainly in the way they are constructed.

We'll explore FileInputStream and FileOutputStream by examining an example program named CopyBytes.java, which uses byte streams to copy kkhsou.txt, one byte at a time and write the content of the file on outagain.txt.

//Program 9.1 : **Program using File Input Stream and File Output Stream**

```
import  java.io.FileInputStream;
import  java.io.FileOutputStream;
import  java.io.IOException;
public  class  CopyBytes
{
       public  static  void  main(String[]  args)  throws  IOException
{
                FileInputStream  in  =  null;
                FileOutputStream  out  =  null;
                try
{
                     in  =  new  FileInputStream("kkhsou.txt");
                     out  =  new  FileOutputStream("outagain.txt");
                     int  c;
```