# Streams

Stream is a flow of data from the source to the destination.

A source can be a keyboard, file, client, server, etc.… and a destination can be a monitor, file, server, client, etc…

In java, Streams are broadly categorized as,

* Console input/output streams
* File input/output streams
* Network input/output streams

Diagram

Description automatically generated

Predefined streams:

* In
* Out
* Err

in 🡺 it is an object reference of java.io.InputStream class.

out and err 🡺 These are object references of the java.io.PrintStream class.

in,out,err 🡺 These are in java.lang.System class

**class: java.io.InputStream**

Method: public int read(byte[]) throws IOException

* This is used to accept the data from the keyboard.

**Class: java.io.PrintStram**

Method: public void print(…)

* This is used to print the data in the console.

**Class: java.lang.System**

public static final java.io.InputStream in;

public static final java.io.PrintStream out;

public static final java.io.PrintStream err;

* These 3 reference variables are defined in System class.

## How System.out.println() works

Refer code snippet: HowSystemOutPrintlnWorksExample1.java

Similar to the above example, out is a reference variable of PrintStream class and it is a static member of the System class. So System Has-A a relationship with PrintStream class.

AS-A relation only goes with extends keyword.

## Difference between “out” and “err” keywords

Refer code snippet: HowSystemOutPrintlnWorksExample1\_Class

System.out:

* It is used to display output messages.
* Stream data can be redirected to a file.

System.err:

* It is used to display error messages.
* Streams cannot be redirected to a file.

## Read From Console and Write To Console

Refer code snippet: ReadInputFromConsoleAndWriteInConsoleExample1\_Class

## Read From File and Write To Console

Diagram, schematic

Description automatically generated

Steps the develop:

* Open a file for reading
* Find out the file size
* Allocate the memory in the RAM according to file size
* Read data from the file.
* Display data on the monitor.
* Close the file.

Refer code snippet: ReadContentFromFiletoDisplayInConsole

## Read From File and Write To File

**Diagram, schematic

Description automatically generated**

* Open a file for reading
* Find out the file size
* Allocate the memory in the RAM according to file size
* Read data from the file.
* Open file from writing.
* Write data to file.
* Close the file.

Refer code snippet: ReadContentFromFiletoWriteInOtherFile

## Read From File and Append to File

* Open a file for reading
* Find out the file size
* Allocate the memory in the RAM according to file size
* Read data from the file.
* Open file from writing.
* Write data to file in append mode.
* Close the file.

Refer code snippet: ReadContentFromFiletoAppendInOtherFile

## Resource statement

This feature allows to write a resource with a try block.

A class that implements java.io.Closeable interface or java.lang.AutoCloseable interface only can be used as a resource.

This resource closed automatically even exception occurs in a program.

This feature is an alternative to finally block.

Refer code snippet: ResourceStatement

## Type of Streams

1. Byte Streams: it handles all types of data (text, images, audio, video, graphics, etc..)
2. Character Stream: it can handle only characters.

**Timeline

Description automatically generated**

DataInputStream and DataOutputStream support all types of primitive data types and strings. If we use these classes we don’t need to convert bytes to strings as we always do when we used inputstream and outputstream classes.

ObjectinputStream and ObjectOutputStream supports objects. At a time objects can be retrieved.

**A screenshot of a computer

Description automatically generated with medium confidence**

## Serialization:

**Java.io.ObjectOutputStream**

Constructor

Public ObjectOutputStream(OutputStream) throws IOException

Method:

Public final void writeObject(Object) throws IOException;

Public void close() throws IOException;

**Java.io.ObjectInputStream**

Constructor

Public ObjectInputStream(InputStream) throws IOException

Method:

Public final void readObject(Object) throws IOException,ClassNotFoundException;

Public void close() throws IOException;

Serialization:

It is a process of converting an object into a series of bits.

In java, the object must be serializable to do the following operations.

1. Writing object to a file
2. Reading objects from file
3. Writing object to a network
4. Reading objects from network

Class must implement java.io.serializable interface to make a serializable object.

Java.io.serializable interface is called a marker interface, tag interface, or empty interface because no members in this interface.

**Diagram

Description automatically generated**

Refer code snippet: SerializationExample