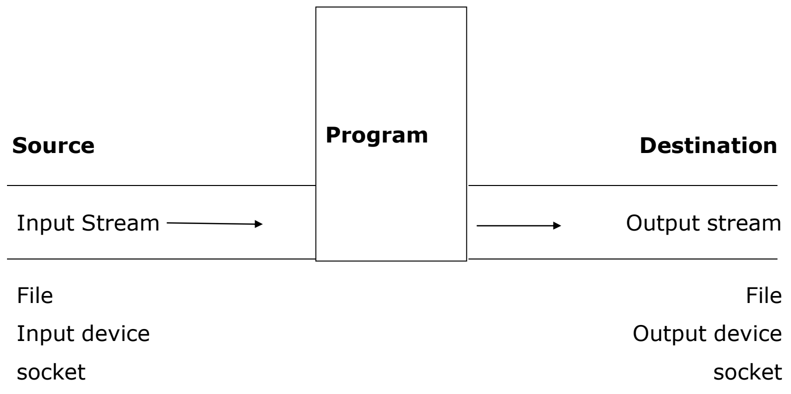
Core Java

IO and File Handling

Objectives

- Identify Streams
- Describe the I/O hierarchy
- Explain the File, RandomAccessFile classes
- Identify Stream oriented classes
- Identify the Reader & Writer classes

Basics



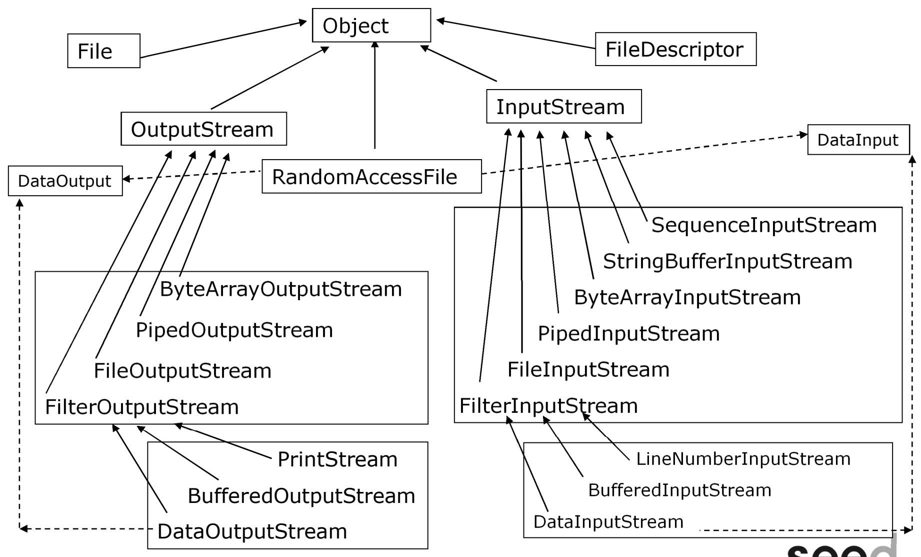
Streams

- InputStream – This depicts the flow of bytes from data source to the programs memory.
- OutputStream – This depicts the flow of bytes from the programs memory to the destination data store.
- Java views these streams in terms of objects that will perform different operations on the streams through their method calls.
- Two basic operations involved are
  - Read from input stream
  - Write to output stream

java.io package

- Provides an extensive set of classes for handling I/O to & from various devices.
- Contains many classes each with a variety of member variables & methods.
- It is layered i.e.. it does not attempt to put too much capability into a single class.
- Instead a programmer can get the features he/she wants by layering one class over another.

Class Hierarchy



IO Hierarchies

- Two Major hierarchies in Java
  - InputStream and OutputStream class hierarchy
    - This hierarchy enables data input output in binary mode.
  - Reader and Writer hierarchy
    - This hierarchy enables data input output in text mode.

InputStream Class

- An abstract class that defines methods for performing i/p.(Read from source)
- Serves as base class for all other InputStream classes.
- Defines a basic interface for reading streamed bytes of information.
- Data in InputStream is transmitted one byte at a time.

Some Methods of the InputStream Class

- int read()
- int read(byte b[])
- int read(byte [] b, int offset, int len)
- int available()
- long skip(long n)
- void close()

FileInputStream Class

- Obtains input bytes from a file in a file system. Useful for performing simple file I/O.
- Can be instantiated using one of the following three constructors:
  - FileInputStream(String name)
  - FileInputStream(File file)
  - FileInputStream(FileDescriptor fdObj)
- This class has been reimplemented using **java.nio** package in order to take the advantage of speed increase, so you will benefit even if you don’t explicitly write code with **nio**.

FilterInputStream

- This is a base class for other classes that act like a filter to transform the raw data bytes to a desired form.
- The important subclasses are
  - DataInputStream
  - BufferedInputStream

seed
beyond the obvious

Chaining or Layering of Streams

- To use these file filters chaining of streams is required.

```
FileInputStream fis = new
                    FileInputStream(“c:\a.txt”);
BufferedInputStream bis = new
                    BufferedInputStream(fis);
DataInputStream dis = new
                    DataInputStream(bis);
```

seed
beyond the obvious

BufferedInputStream

- By default streams are not buffered.
  - i.e. every call to read() contacts the OS to ask it to provide next byte.
- BufferedInputStreams reads characters from a stream without causing a device access everytime.
- Maintains a buffer of bytes read from the original input stream

BufferedInputStream

- Requests to read from the BufferedInputStream, retrieves bytes from this buffer, rather than performing read() operations on original InputStream.
- When all bytes from the buffer have been read, the buffer is refilled with input from the original inputstream.
- This can improve performance significantly because it reduces the number of read() operations on the original input stream.

DataInputStream

- Useful for reading primitive java data types from an I/P stream in a portable manner.
- It aggregates groups of bytes into primitive data types.
- Methods implemented by DataInputStream are variations of the read() method for different fundamental data types.

StringBufferInputStream class

- Enables user to use a string as a buffered source of input.
- This class allows an application to create an input stream in which the bytes read are supplied by the contents of a string
- public StringBufferInputStream(String s)
  - Creates a string input stream to read data from the specified string

OutputStream

- Similar to InputStream there is an OutputStream class
- The methods in OutputStream class are
  - write()
  - flush()
- The important OutputStream subclasses are

  FileOutputStream

  PipedOutputStream

  FilterOutputStream

  DataOutputStream

File Class

- Models an OS dir entry, enabling you to access info about a file.
- File objects are used to do all operations related to files & directories.
- Objects of File do not actually open a file or provide any file processing capabilities.
- It provides methods for performing file related operations that actually interact with the underlying file system.

File Class Methods

- boolean canRead() / canWrite()
- boolean exists()
- boolean isAbsolute()
- boolean isDirectory() / isFile()
- String getParent()
- long length()
- String[] list()
- boolean mkdir()
- boolean delete()
- boolean createNewFile()

seed
beyond the obvious

RandomAccessFile

- RandomAccessFile(String name, String mode)
- RandomAccessFile(File file, String mode)
- This class lets you find or write data anywhere in a file.
- A RAF has a file-pointer setting that comes with it.
- The FP indicates the position of the next record that will be read or written.
- This class has been reimplemented using **java.nio** package in order to take the advantage of speed increase, so you will benefit even if you don’t explicitly write code with **nio**.

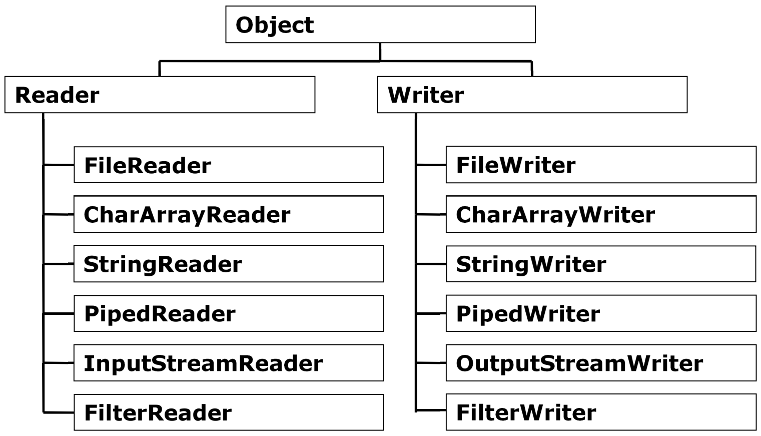
seed
beyond the obvious

RandomAccessFile

- seek(long pos) : sets the file pointer to an arbitrary byte position within the file.
- getFilePointer() : returns current location of file pointer from the beginning of file.
- long length() : length of file in bytes.
- skipBytes() : moves current input position the specified number of byte forward or backward.

- Example:- IOStreamDemo.java

Reader - Writer Hierarchy



The System Class

- Three static I/O objects have already been created by the time main() method gains control.
- All 3 are public static members of System class.
  - System.in
  - System.out
  - System.err
  - Streams associated with these objects provide communication channels between a program & a particular file or device.

The System Class - Demo

```
import java.io.*;
class ReadKeys {
 public static void main (String args[]) {
      StringBuffer sb = new StringBuffer();
      char c;
     try {
     while((ch =(char)System.in.read()) != '\n'))
             {
                     sb.append(c); }
             }
        catch (Exception e) { ...  }
  String s = new String(sb);
  System.out.println(s);
     }
 }
```