Topic 3
Java Overview
with
Streams & Sockets

Part A

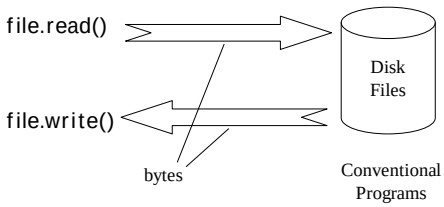
Basic Streams and Sockets

Streams

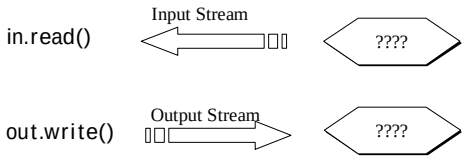
System.in	an Input Stream
System.out	an Output Stream

ordered sequences of data

Conventional File I/O

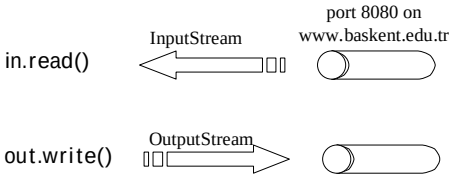


Stream-Based I/O



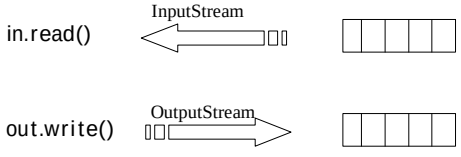
Streams hide underlying source of data

Stream-Based I/O



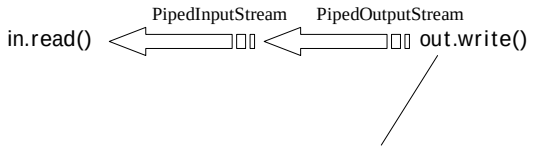
Option: create a stream on a socket

Stream-Based I/O



Option: create a stream on an array in memory

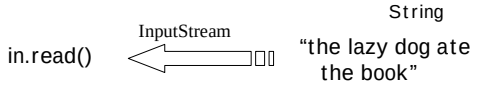
Stream-Based I/O



Option: create a stream on another stream.

Useful to allow threads to communicate!

Stream-Based I/O



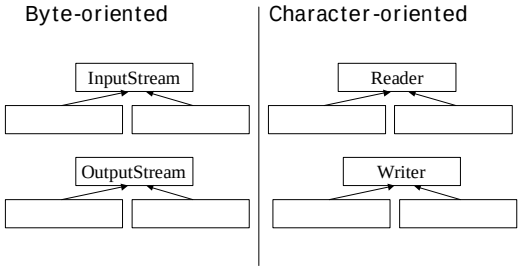
Option: create a stream on a StringBuffer

Useful for parsing a String with multiple items.

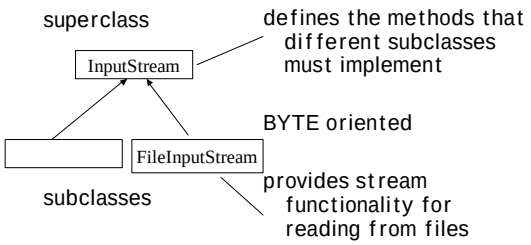
The java.io package

(defined in terms of streams)

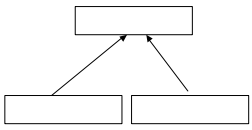
Two Stream Worlds



Abstract InputStream class



InputStream functionality



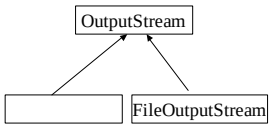
subclasses must implement read()

returns –1 if no more data

abstract int read()
int read(byte[])
int read(byte[], int off, int len)
long skip(long n)
void close()
int available()

returns number of bytes available w/o blocking

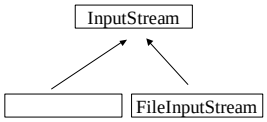
abstract OutputStream class



abstract int write(int b)
void write(byte[])
void write(byte[], int off, int len)
void close()

subclasses must implement write()

FileInputStream class



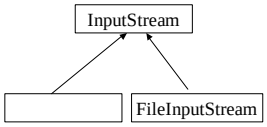
Constructor:
FileInputStream(filename String)

```
class SpaceCount {
   public static void main(String[] args) throws IOException {
        InputStream in;
        if (args.length == 0)
                in = System.in;
        else
                in = new FileInputStream(args[0]);
        int ch, total, spaces = 0;
        for (total=0; (ch = in.read()) != -1; total++)
                if (Character.isSpaceChar((char)ch))
                        spaces++;

        System.out.println(total+" characters and "
                + spaces + " spaces");
   }
}
```

```
%java SpaceCount this is it
12 characters and 2 spaces
```

FileInputStream

superclass



provides stream functionality for reading from files

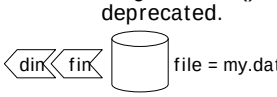
subclasses

Limited to BYTES … need something more…

DataInputStream

```
DataInputStream din = new DataInputStream(… )

k = din.readInt()
f = din.readFloat()
b = din.readByte()
c = din.readChar()
```

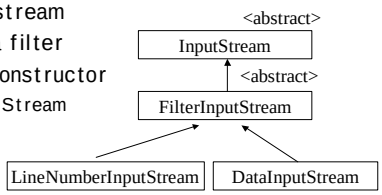
note:
String readLine() is deprecated.



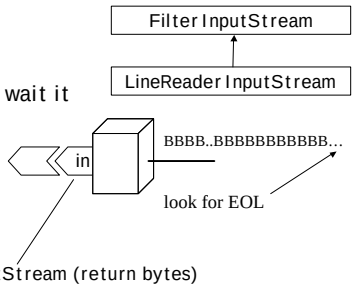

```
FileInputStream fin = new FileInputStream("my.dat");
DataInputStream din = new DataInputStream(fin);
```

DataInputStream is-a Filter Stream

- Gets input from another stream
- Acts as a filter
- Special constructor
  - arg is a Stream

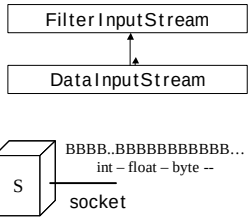


FilterInputStream



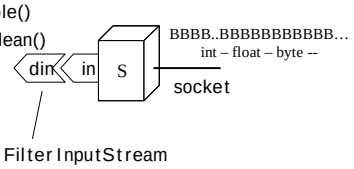
- can return whatever we wait it to!!
- as long as we provide the methods

DataInputStream



METHODS

int readInt()
float readFloat()
double readDouble()
boolean readBoolean()
char readChar()
byte readByte()



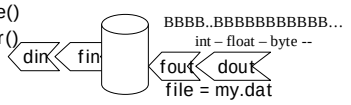
DataInputStream din = new DataINputStream(in)

DataInputStream

Must know order in which data elements were written TO the corresponding Output Stream.

OutputStream

k = din.readInt()
f = din.readFloat()
b = din.readByte()
c = din.readChar()



FileInputStream fin = new FileInputStream("my.dat");
DataInputStream din = new DataInputStream(fin);

Sample Program DataStream Test

shows how to read and write Java primitive types using DataStreams

```
public static void writeData(double[] data, String file)
    throws IOException
{
    OutputStream fout = new FileOutputStream(file);
    DataOutputStream out = new DataOutputStream(fout);

    out.writeInt(data.length);
    for (int i=0; i < data.lentgh; i++)
        out.writeDouble(data[i]);

    out.close();
}
```

```
public static double[] readData(String file)
                                    throws IOException
{
    InputStream fin = new FileInputStream(file);
    DataInputStream in = new DataInputStream(fin);
    double[] data = new double[in.readInt()];

    for (int i=0; i < data.lentgh; i++)
        data[i] = in.readDouble();

    in.close();
    return data;
}
```

```
class  DataStreamTest {
public static void main(String[] args) throws IOException {
   double[] doubleDat = {12.2, 33.5, 99.45};
   double newDat[];
   char option = args[0].charAt(0);
   String fname = args[1];

   if (option == 'w')
        writeData(doubleDat, fname);
   else if (option == 'r') {
        newDat = readData(fname);
        for (int i=0; i<newDat.length; i++)
          System.out.println(newDat[i]);
   }
}
```

Program: FilterTest

Shows use of
LineNumberInputStream
(a subclass of
FilterInputStream)

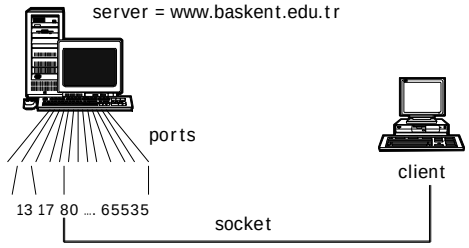
```
class FilterTest {
   public static void main(String[] args) throws IOException {

   if (args.length != 2)
        error("Need a character and a file name");

   int matchChar = args[0].charAt(0);

   FileInputStream fileIn = new FileInputStream(args[1]);

   LineNumberInputStream in = new
                              LineNumberInputStream(fileIn);
```

```
int ch;
while ((ch = in.read()) != -1) {
   if (ch == matchChar) {
      System.out.println("→" + (char)ch +
            "at line " + in.getLineNumber() );
      System.exit(0);
   }
   System.out.println(matchChar + " not found");
}
```

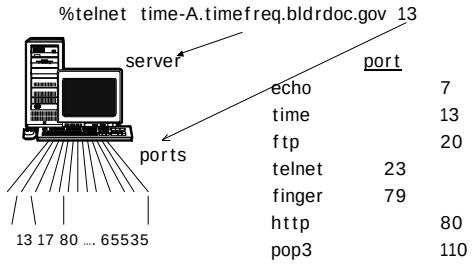
- java FilterTest j FilterTest.java
- → j at line 0
- java FilterTest C FilterTest.java
- → C at line 8

Examine source code for
LineNumberInputStream

Sockets and Strems



TCP/IP in action



port	
echo	7
time	13
ftp	20
telnet	23
finger	79
http	80
pop3	110

lower port numbers
0…1023 are reserved

class Socket

- Socket is the Java representation of a TCP network connection
- Using Socket, a client can create a stream-based communication channel with a remote host
- To communicate, a stream connection to the remote host must be established – specifying the address and the port
- There must be a server program actively listening on the port or the connection will fail.

Socket constructor

Socket(String host, int port)
throws IOException

- port must be by name or text IP address
- port must be 1 - 65535

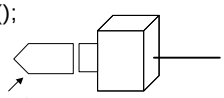
java.net.Socket

Constructor:

- . . . **

```
Socket s = new Socket(www.sun.com, 80)
in = s.getInputStream();
int k = in.read();
```

returns 1st byte of web page



an InputStream (returns bytes)

Socket methods

- Socket(String host, int port)
  - constructor
- synchronized void close()
  - closes the socket
- InputStream getInputStream()
  - gets InputStream for reading
- OutputStream getOutputStream()
  - gets OutputStream for writing
- void setSoTimeOut(int timeout)
  - Socket will block for only this amount of time –
    - then InterruptedException is raised

Program: SocketTest

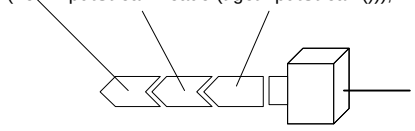
Reads a line at a time from a server …

Uses a BufferedReader stream which "knows" how to readline()

SocketTest.java

```
import java.io.*;
import java.net.*;

class SocketTest {
    public static void main(String[] args) {
      try {
        Socket  t = new Socket("time-A.timefreq.bldrdoc.gov", 13);

        BufferedReader is = new BufferedReader
                (new InputStreamReader(t.getInputStream()));
        boolean more = true;
        while (more) {
           String str = is.readLine();
           if (str == null) more = false;
           else             System.out.println(str);
         }
      } catch (IOException e) {
         System.out.println("Error"+e);
      {
}
```

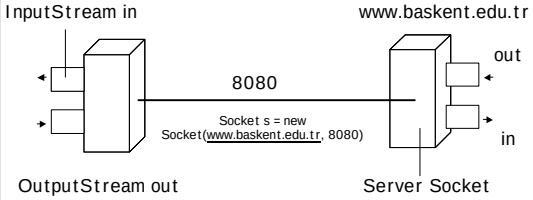
```
Socket  t = new Socket("time-A.timefreq.bldrdoc.gov", 13);

BufferedReader is = new BufferedReader
        (new InputStreamReader(t.getInputStream()));
```



BufferedReader has non-deprecated readLine()

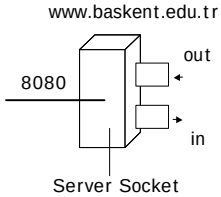
Java Socket Server

Server World



java.net.ServerSocket

- Listens on port for connection requests from clietns
- Specify port when you create a ServerSocket



- ServerSocket s = new ServerSocket(8080);

java.net.ServerSocket

- ServerSocket(int port)
  - simple constructor
- accept()
  - listens for incoming client requests
  - when a client connects, returns a Socket
- close()
  - closes ServerSocket so it no longer listens
- getInetAddress
  - returns local host address
  - after client connection, returns client address

Life of a Server Socket

- Instantiated
  - when constructor completes; call accept() and begin listening
- Accepting
  - when accept() is called; it will block and wait forever until client shows up
- Connected
  - when client shows up a Socket is returned
  - use getInetAddress and getLocalPort to get Information about client

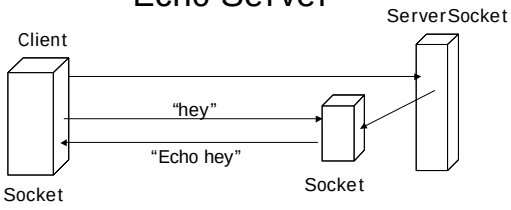
Server Testing

- Use one machine
  - server = "localhost" or 127.0.0.1
- Allow a socket to connect to a server running on the same machine

TCP vs UDP

- UDP – Unreliable Datagram Protocol
- Packets not guaranteed
- Order not guaranteed
- Separate classes
  - Datagram Socket
  - Datagram Packet

Echo Server



- one connection – no more connections allowed
- when done ServerSocket terminates

Simple Echo Server

```
ServerSocket s = new ServerSocket(8080);
Socket incoming = s.accept();
```

Socket – same class used when creating a client connection

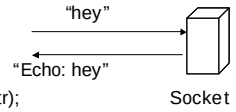
blocks until client arrives and requests a connection

Simple Echo Server

```
ServerSocket s = new ServerSocket(8080);
Socket incoming = s.accept();

BufferedReader in = new BufferedReader
        (new InputStreamReader(incoming.getInputStream()));

PrintWriter out = new PrintWriter
        (incoming.getOutputStream(), true /*autoFlush*/);
                ↗
//ask Socket for a stream
out.println("Hi! Type BYE to quit");
```

Main Echo Server Loop

```
boolean done = false;
while (! done) {
   String str = in.readLine();
   if (str == null) done = true;
   else {
        out.println("Echo:" + str);
        if (str.trim().equals("BYE"))
                done = true;
   }
}
```



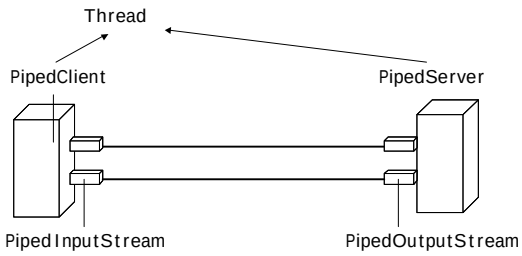
BUT, our server can only serve ONE client .. the FIRST to arrive

Ex2.2 - XDRInputStream

- a kind-of FilterInputStream
- defines:
  - readByte()
  - readFloat()
  - readXXX()

… based on XDR formatted data stream

Threads & PipedInputStreams



See Examples

- PipedClient 2.4; PipedServer 2.5; PipedApp 2.6
- shows extend Thread; run(); start(); join(); client server w/o sockets!

Summary

- Sockets and streams provide low-level communication capability for clients and servers
- Requires that the clients and servers understand what will be sent and received
- The “protocol is the program”