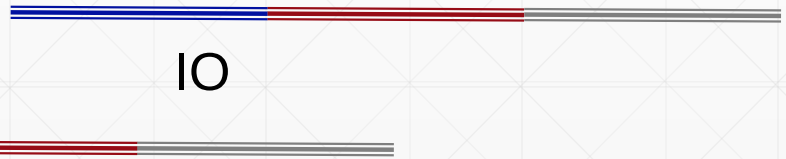
Computer Language

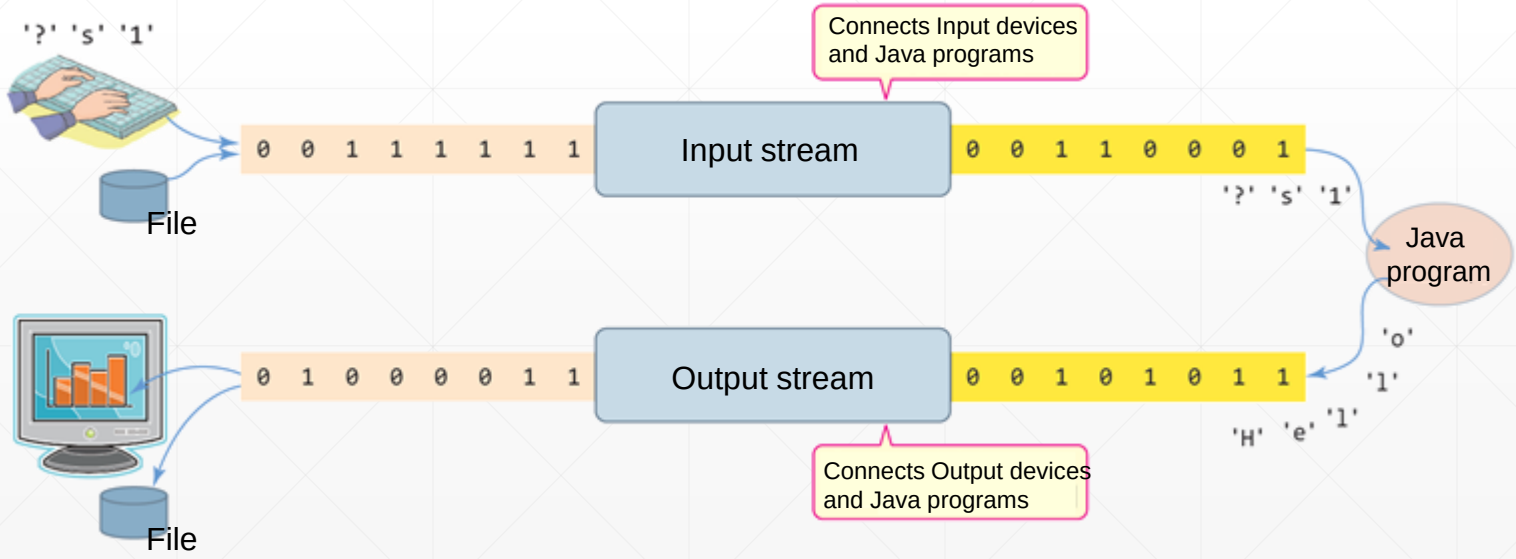


Agenda

- IO
- Exercises

Stream

- Stream IO
  - Input and Output (IO) processing based on the buffer
- Java's IO Stream
  - Input stream: takes data from input devices and bring it to the java program
  - Output stream: pass data to output devices

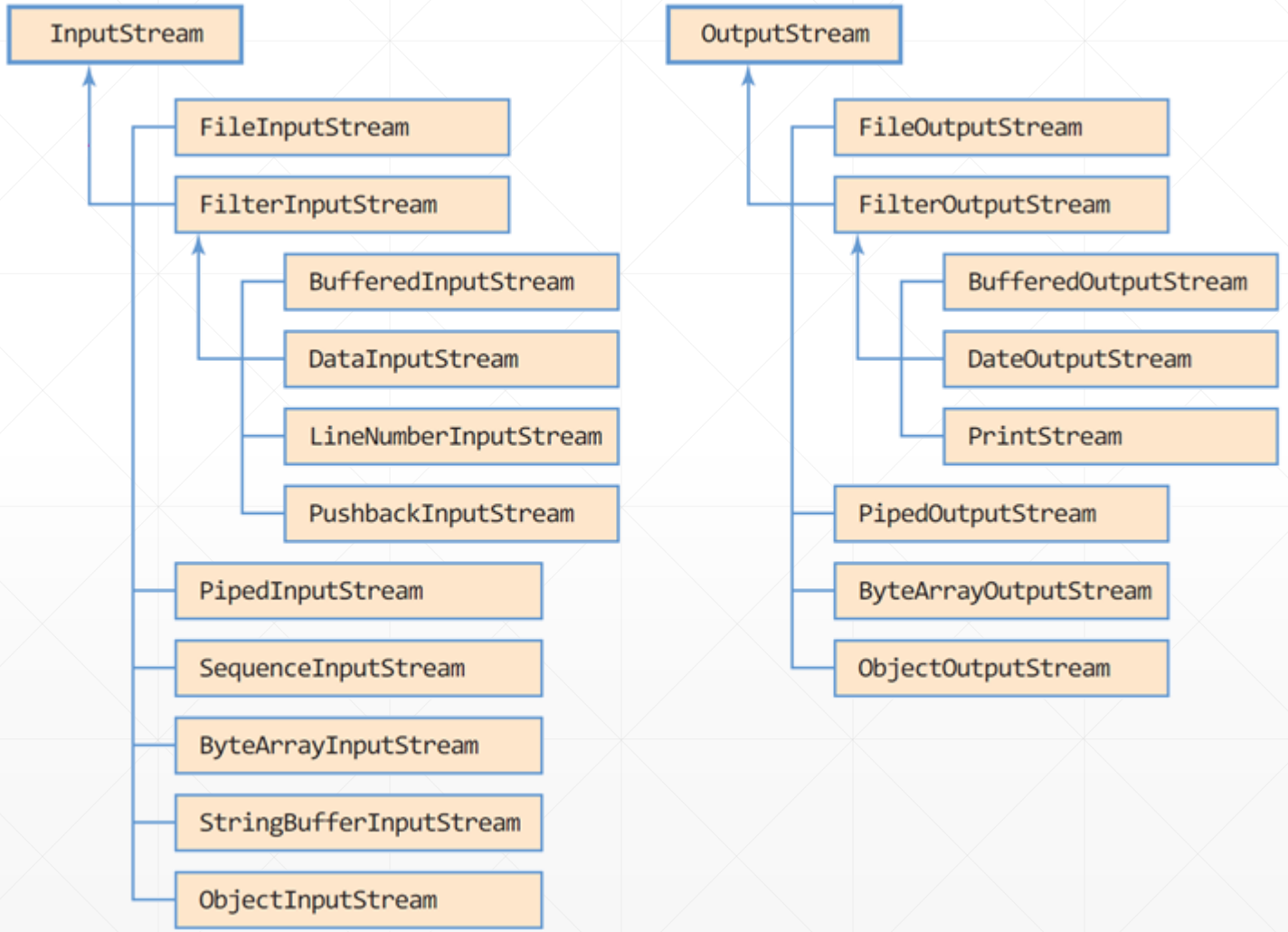


Stream (cont'd)

- Characteristics
  - Uni-directional
  - Basic unit
    - Byte for byte stream
      - Transmits any type of data (e.g., image, video, etc.)
    - Character for character stream
      - Transmits character data only (e.g., text file)
  - FIFO
    - Frist-in first-out

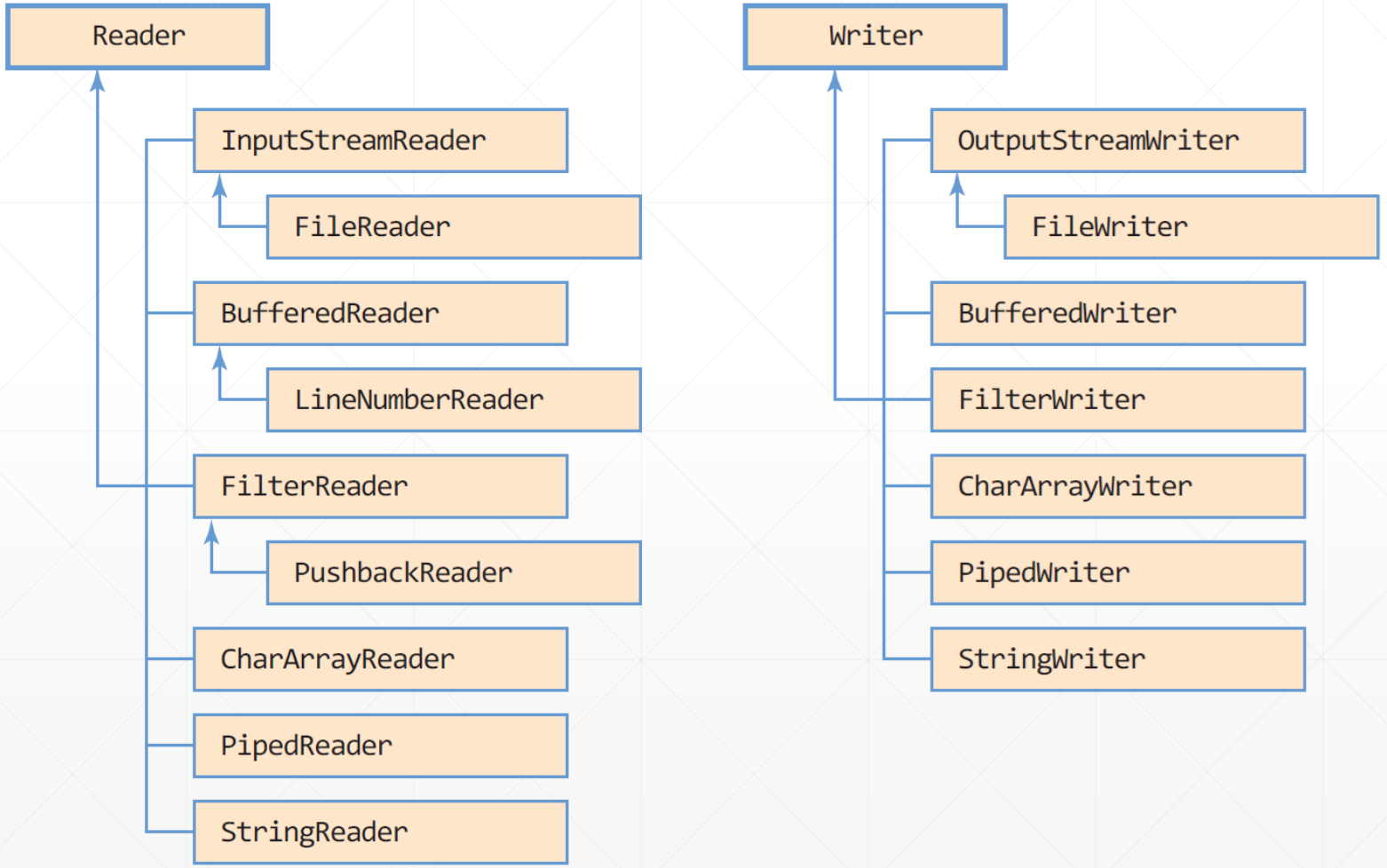
Stream (cont'd)

- Byte stream hierarchy

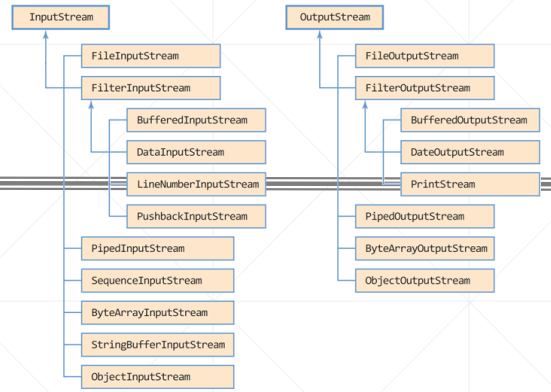


Stream (cont'd)

- Character stream hierarchy



Stream (cont'd)



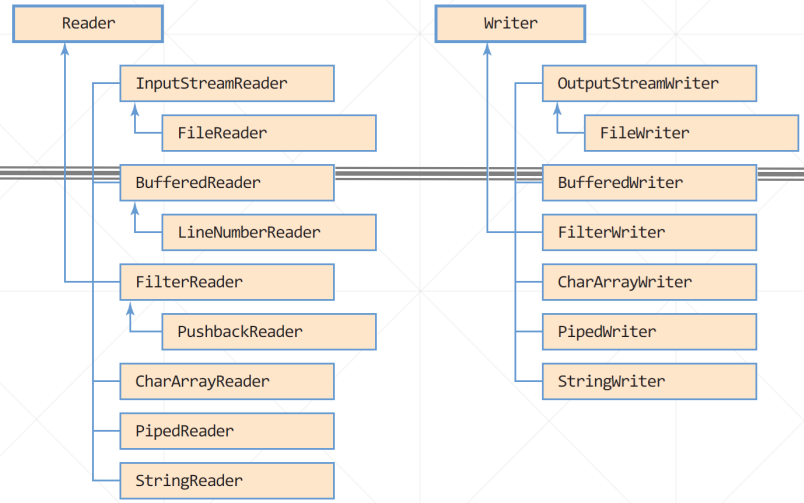
- Root abstract classes of byte streams
  - InputStream

void	**close**()	Closes this input stream and releases any system resources associated with the stream.
abstract 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.

  - OutputStream

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

Stream (cont'd)



- Root abstract classes of character streams
  - Reader

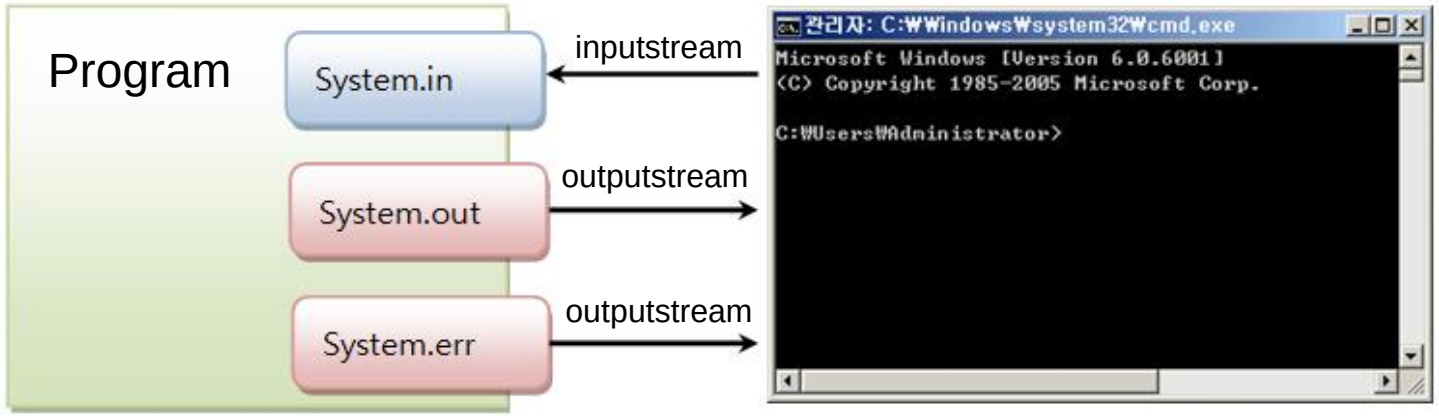
int	**read**()	Reads a single character
int	**read**(char[] cbuf)	Reads characters into an array
abstract int	**read**(char[] cbuf, int off, int len)	Reads characters into a portion of an array

  - Writer

abstract void	**flush**()	Flushes the stream
void	**write**(char[] cbuf)	Writes an array of characters
abstract void	**write**(char[] cbuf, int off, int len)	Writes a portion of an array of characters
void	**write**(int c)	Writes a single character
void	**write**(**String** str)	Writes a string
void	**write**(**String** str, int off, int len)	Writes a portion of a string

Stream: Console

- System software, application interface
  - Linux terminal, windows prompt, IntelliJ/Eclipse console, etc.
  - Take input from the keyboard
  - Output the data to display



- System class
  - “in” field: standard input stream (InputStream)
  - “out” field: standard output stream (PrintStream)
  - “err” field: standard error error stream (PrintStream)

FileReader

Reading a text file

Constructor	Description
FileReader(File file)	Creates a new FileReader, given the File to read
FileReader(File file, Charset charset)	Creates a new FileReader, given the File to read and the charset.
FileReader(String fileName)	Creates a new FileReader, given the name of the file to read
FileReader(String fileName, Charset charset)	Creates a new FileReader, given the name of the file to read and the charset.

```
public class FileReaderEx {
   public static void main(String[] args) {
      FileReader fin = null;
      try {
         fin = new FileReader("c:₩₩windows₩₩system.ini");
         int c;
         while ((c = fin.read()) != -1) { // read a character
            System.out.print((char)c);
         }
         fin.close();
      }
      catch (IOException e) {
         System.out.println("IO error!");
      }
   }
}
```

FileWriter

- Writing to a text file

Constructor	Description
FileWriter(File file)	Constructs a FileWriter given the File to write
FileWriter(File file, boolean append)	Constructs a FileWriter given the File to write and a boolean indicating whether to append the data written
FileWriter(File file, Charset charset)	Constructs a FileWriter given the File to write and charset.
FileWriter(File file, Charset charset, boolean append)	Constructs a FileWriter given the File to write, charset and a boolean indicating whether to append the data written.
FileWriter(String fileName)	Constructs a FileWriter given a file name
FileWriter(String fileName, boolean append)	Constructs a FileWriter given a file name and a boolean indicating whether to append the data written
FileWriter(String fileName, Charset charset)	Constructs a FileWriter given a file name and charset.
FileWriter(String fileName, Charset charset, boolean append)	Constructs a FileWriter given a file name, charset and a boolean indicating whether to append the data written.

- Character or Block writing is possible

```
FileWriter fout = new FileWriter("c:₩₩Temp₩₩test.txt");
fout.write('A'); // writing character 'A' to the file
fout.close();
```

```
char [] buf = new char [1024];
// writing the contents of buf[] (1024 characters) to the file
fout.write(buf, 0, buf.length);
```

FileWriter (cont'd)

- Writing to a text file

```
import java.io.*;
import java.util.*;

public class FileWriterEx {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        FileWriter fout = null;
        int c;
        try {
            fout = new FileWriter("c:₩₩Temp₩₩test.txt");
            while(true) {
                String line = scanner.nextLine();
                if(line.length() == 0)
                    break;
                fout.write(line);
                fout.write("₩r₩n");
            }
            fout.close();
        } catch (IOException e) {
            System.out.println("IO error!");
        }
        scanner.close();
    }
}
```

Inserts “\r\n” escape characters to insert a new line

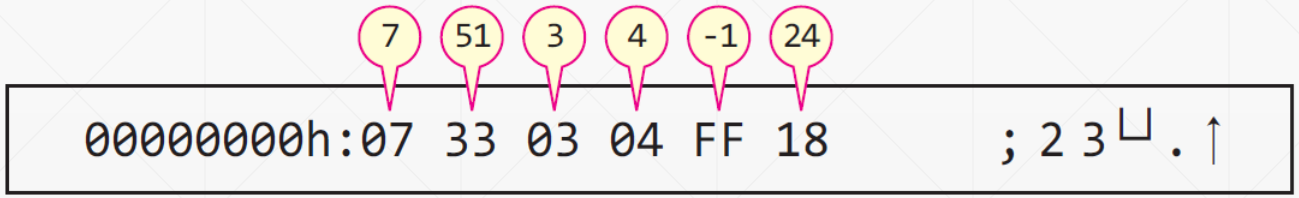
FileOutputStream

- Writing to a binary file

Constructor	Description
FileOutputStream(File file)	Creates a file output stream to write to the file represented by the specified File object.
FileOutputStream(File file, boolean append)	Creates a file output stream to write to the file represented by the specified File object.
FileOutputStream(String name)	Creates a file output stream to write to the file with the specified name.
FileOutputStream(String name, boolean append)	Creates a file output stream to write to the file with the specified name.

- Binary data is not human-readable (not text!)

```
byte b[] = {7, 51, 3, 4, -1, 24};
try {
    FileOutputStream fout =
            new FileOutputStream("c:\\Temp\\test.out");
    for (int i = 0; i < b.length; i++)
        fout.write(b[i]);
    fout.close();
} catch (IOException e) {
    System.out.println("could not save the file!");
    return;
}
System.out.println("saved to c:\\Temp\\test.out");
```



FileInputStream

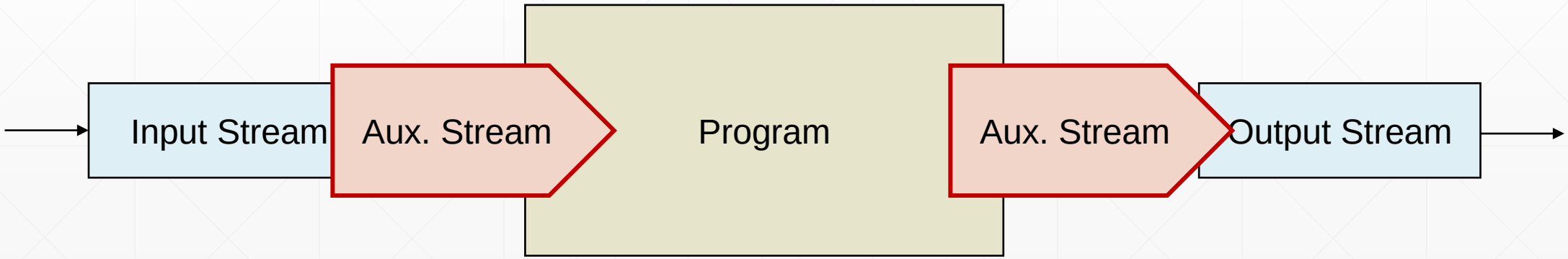
- Reading a binary file

Constructor	Description
FileInputStream(File file)	Creates a FileInputStream by opening a connection to an actual file, the file named by the File object file in the file system.
FileInputStream(String name)	Creates a FileInputStream by opening a connection to an actual file, the file named by the path name name in the file system.

```
byte b[] = new byte [6];
try {
    FileInputStream fin = new FileInputStream("c:\\Temp\\test.out");
    int n=0, c;
    while((c = fin.read())!= -1) {
        b[n] = (byte)c;
        n++;
    }
    System.out.println("Printing the contents from c:\\Temp\\test.out");
    for(int i=0; i<b.length; i++) System.out.print(b[i] + " ");
    System.out.println();
    fin.close();
} catch(IOException e) {
    System.out.println( "could not read c:\\Temp\\test.out!!");
}
```

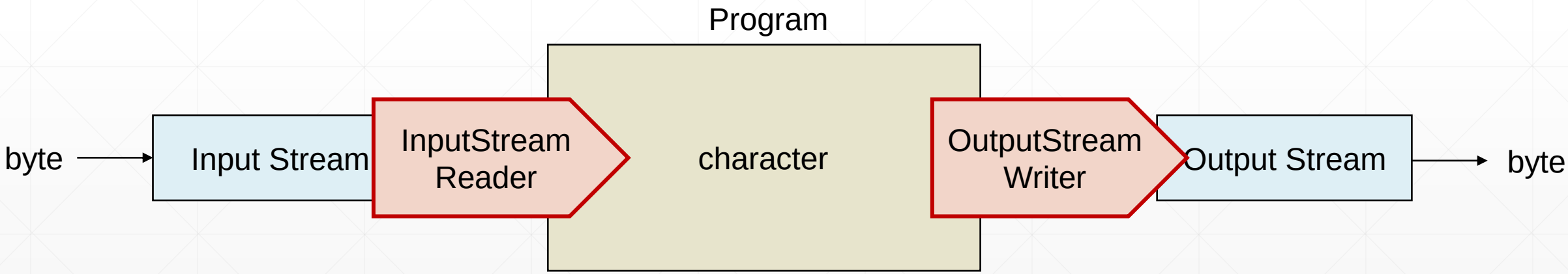
Auxiliary Stream

- Bridge between the streams
  - Provides useful stream features
    - Character conversion
    - Buffered I/O
    - Object I/O
    - …
  - Can be chained



Auxiliary Stream: Character Conversion

- InputStreamReader / OutputStreamWriter
  - Converts byte data from Input stream to character data
  - Converts character data to byte data for Output stream
  - Can set a specific character set

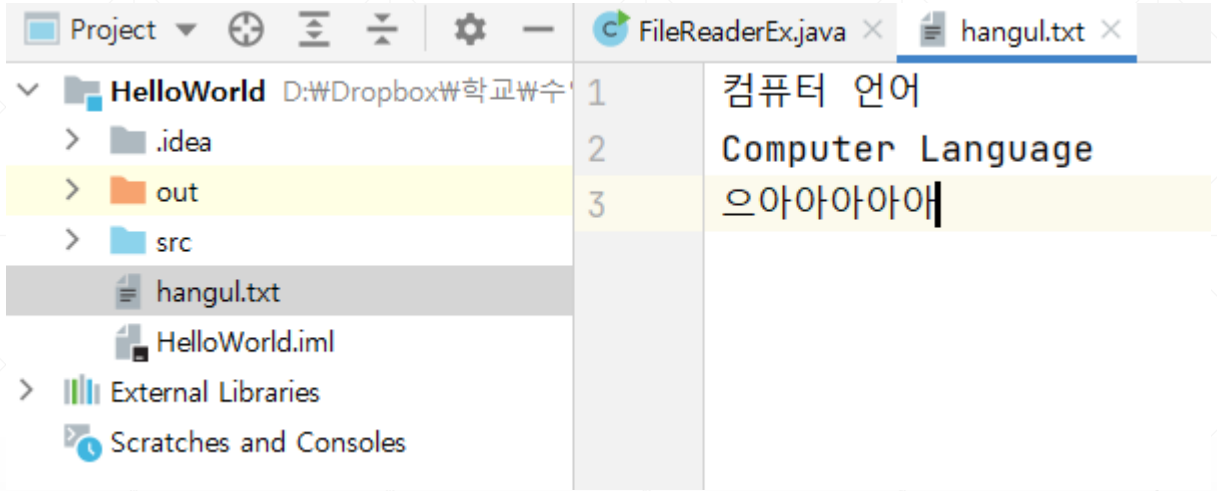


Auxiliary Stream: Character Conversion (cont'd)

- Example)

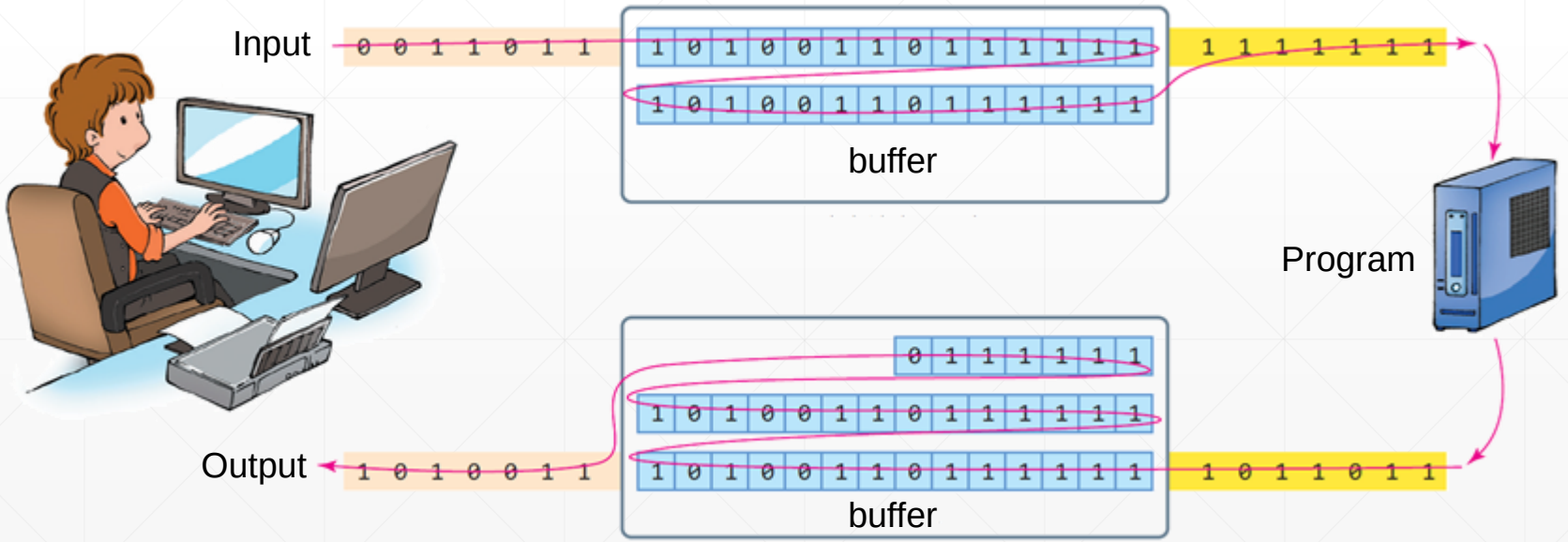
```
InputStreamReader in = null;
FileInputStream fin = null;
try {
    fin = new FileInputStream("hangul.txt");
    in = new InputStreamReader(fin, "utf-8");
    int c;

    System.out.println("encoding: " + in.getEncoding());
    while ((c = in.read()) != -1) {
        System.out.print((char)c);
    }
    in.close();
    fin.close();
} catch (IOException e) {
    System.out.println("IO error!");
}
```



Auxiliary Stream: Buffering I/O

- Buffered Streams
  - BufferedInputStream / BufferedOutputStream (for binary data)
  - BufferedReader / BufferedWriter (for character data)
  - Improves I/O efficiency by reducing native I/O operations
    - Keep the data in the buffer!



Auxiliary Stream: Buffering I/O (cont'd)

- Example)

```
FileReader fin = null;
int c;
try {
    fin = new FileReader("c:\\windows\\system.ini");
    BufferedOutputStream out = new
            BufferedOutputStream(System.out, 128);
    while ((c = fin.read()) != -1) {
        out.write(c);
    }

    new Scanner(System.in).nextLine(); // waiting for Enter
    out.flush(); // flushing buffer!
    fin.close();
    out.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

File

- java.io.File
  - Class handling a file's information (metadata)
    - Path of file/directory
  - Class handling file management
    - Renaming, removing, creating, etc
  - Does not support read/write functionalities

- File instance

```
File f = new File("c:₩₩windows₩₩system.ini");
```

File (cont'd)

- Methods
  - Creation and deletion

Modifier and Type	Method	Description
boolean	createNewFile()	Atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.
boolean	mkdir()	Creates the directory named by this abstract pathname.
boolean	mkdirs()	Creates the directory named by this abstract pathname, including any necessary but nonexist ent parent directories.
boolean	delete()	Deletes the file or directory denoted by this abstract pathname.

File (cont'd)

- Methods
  - Get information of files and directories

Modifier and Type	Method	Description
boolean	canExecute()	Tests whether the application can execute the file denoted by this abstract pathname.
boolean	canRead()	Tests whether the application can read the file denoted by this abstract pathname.
boolean	canWrite()	Tests whether the application can modify the file denoted by this abstract pathname.
String	getName()	Returns the name of the file or directory denoted by this abstract pathname.
String	getParent()	Returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.
File	getParentFile()	Returns the abstract pathname of this abstract pathname's parent, or null if this pathname does not name a parent directory.
String	getPath()	Converts this abstract pathname into a pathname string.

File (cont'd)

- Methods
  - Get information of files and directories

Modifier and Type	Method	Description
boolean	isDirectory()	Tests whether the file denoted by this abstract pathname is a directory.
boolean	isFile()	Tests whether the file denoted by this abstract pathname is a normal file.
long	length()	Returns the length of the file denoted by this abstract pathname.
String[]	list()	Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname.
File[]	listFiles()	Returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname.

File (cont'd)

Example)

- Create a File instance

```
File f = new File("c:₩₩windows₩₩system.ini");
```

- Get File path

```
String filename = f.getName();    // "system.ini"
String path = f.getPath();        // "c:₩₩windows₩₩system.ini"
String parent = f.getParent();    // "c:₩₩windows"
```

- Check

```
if(f.isFile()) // in case of file
   System.out.println(f.getPath() + "is a file.");
else if(f.isDirectory()) // in case of directory
   System.out.println(f.getPath() + "is a directory.");
```

- Get files and subdirectories

```
File f = new File("c:₩₩Temp");
File[] subfiles = f.listFiles(); // get files and subdirectories of c:₩Temp

for(int i=0; i< subfiles.length; i++) {
   System.out.print(subfiles[i].getName()); // print names
   System.out.println("₩tFile size: " + subfiles[i].length()); // print length
}
```

File (cont'd)

- Example)

```
import java.io.File;

public class FileEx {
    public static void listDirectory(File dir) {
        System.out.println("-----" + dir.getPath() + "'s sub list -----");
        File[] subFiles = dir.listFiles();
        for(int i=0; i<subFiles.length; i++) {
            File f = subFiles[i];
            long t = f.lastModified();
            System.out.print(f.getName());
            System.out.print("₩tFile Size: " + f.length());
            System.out.printf("₩tModified time: %tb %td %ta %tT₩n",t, t, t, t);
        }
    }

    public static void main(String[] args) {
        File f1 = new File("c:₩₩windows₩₩system.ini");
        System.out.println(f1.getPath() + ", " + f1.getParent() + ", " +  f1.getName());
        String res="";
        if(f1.isFile()) res = "File";
        else if(f1.isDirectory()) res = "Directory";
        System.out.println(f1.getPath() + " is " + res);
        File f2 = new File("c:₩₩Temp₩₩java_sample");
        if(!f2.exists()) {
            f2.mkdir(); // if not exist, make a new directory
        }
        listDirectory(new File("c:₩₩Temp"));
        f2.renameTo(new File("c:₩₩Temp₩₩javasample"));
        listDirectory(new File("c:₩₩Temp"));
    }
}
```

Example: Copying Text Files

```
import java.io.*;

public class TextCopyEx {
   public static void main(String[] args){
      File src = new File("c:₩₩windows₩₩system.ini"); // source file
      File dest = new File("c:₩₩Temp₩₩system.txt"); // destination file
      int c;
      try {
         FileReader fr = new FileReader(src);
         FileWriter fw = new FileWriter(dest);
         while((c = fr.read()) != -1) { // read a single character
            fw.write((char)c); // write a single character
         }
         fr.close(); fw.close();
         System.out.println(src.getPath()+  "  copied to  " + dest.getPath());
      } catch (IOException e) {
         System.out.println("IO error!");
      }
   }
}
```

Example: Copying Binary Files

```
import java.io.*;

public class BinaryCopyEx {
   public static void main(String[] args) {
      File src = new File("img1.jpg");
      File dest = new File("copyimg.jpg");
      int c;
      try {
         FileInputStream fi = new FileInputStream(src);
         FileOutputStream fo = new FileOutputStream(dest);
         while((c = fi.read()) != -1) {
            fo.write((byte)c);
         }
         fi.close();
         fo.close();
         System.out.println(src.getPath()+ " copied to " + dest.getPath());
      } catch (IOException e) {
         System.out.println("IO Error!");
      }
   }
}
```

Example: Copying Binary Files with Buffer

```
import java.io.*;

public class BinaryCopyEx {
    public static void main(String[] args) {
        File src = new File("img1.jpg");
        File dest = new File("copyimg.jpg");
        int c;
        try {
            FileInputStream fi = new FileInputStream(src);
            FileOutputStream fo = new FileOutputStream(dest);

            BufferedInputStream bi = new BufferedInputStream (fi);
            BufferedOutputStream bo = new BufferedOutputStream (fo);

            while((c = bi.read()) != -1) {
                bo.write((byte)c);
            }
            bi.close();
            bo.close();
            fi.close();
            fo.close();
            System.out.println(src.getPath()+ " copied to " + dest.getPath());
        } catch (IOException e) {
            System.out.println("IO Error!");
        }
    }
}
```

Scanner with File

- A simple text scanner
  - Java.util.scanner

Constructors

Constructor	Description
`Scanner(File source)`	Constructs a new Scanner that produces values scanned from the specified file.
`Scanner(File source, String charsetName)`	Constructs a new Scanner that produces values scanned from the specified file.
`Scanner(File source, Charset charset)`	Constructs a new Scanner that produces values scanned from the specified file.
`Scanner(InputStream source)`	Constructs a new Scanner that produces values scanned from the specified input stream.
`Scanner(InputStream source, String charsetName)`	Constructs a new Scanner that produces values scanned from the specified input stream.
`Scanner(InputStream source, Charset charset)`	Constructs a new Scanner that produces values scanned from the specified input stream.
`Scanner(Readable source)`	Constructs a new Scanner that produces values scanned from the specified source.
`Scanner(String source)`	Constructs a new Scanner that produces values scanned from the specified string.
`Scanner(ReadableByteChannel source)`	Constructs a new Scanner that produces values scanned from the specified channel.
`Scanner(ReadableByteChannel source, String charsetName)`	Constructs a new Scanner that produces values scanned from the specified channel.
`Scanner(ReadableByteChannel source, Charset charset)`	Constructs a new Scanner that produces values scanned from the specified channel.
`Scanner(Path source)`	Constructs a new Scanner that produces values scanned from the specified file.
`Scanner(Path source, String charsetName)`	Constructs a new Scanner that produces values scanned from the specified file.
`Scanner(Path source, Charset charset)`	Constructs a new Scanner that produces values scanned from the specified file.

Scanner with File (cont'd)

- Reading Text files using Scanner
  - Scanner(File)
  - Scanner(FileReader)

```
FileReader fin = null;
try {
   fin = new FileReader("c:₩₩windows₩₩system.ini");
   int c;
   while ((c = fin.read()) != -1) { // read a character
            System.out.print((char)c);
   }
   fin.close();
}
catch (IOException e) {
   System.out.println("IO error!");
}
```

```
try {
   Scanner scn = new Scanner(new File("c:\\windows\\system.ini"));
   while (scn.hasNext()) {
      String tmp = scn.nextLine();
      System.out.println(tmp);
   }
   scn.close();

} catch (IOException e) {
   e.printStackTrace();
}
```

```
FileReader fin = null;
try {
   fin = new FileReader("c:\\windows\\system.ini");
   Scanner scn = new Scanner(fin);
   while(scn.hasNext()) {
      String tmp = scn.nextLine();
      System.out.println(tmp);
   }
   fin.close();
   scn.close();

} catch (IOException e) {
   e.printStackTrace();
}
```

Q&A

- Next week (Offline Test)
  - Final exam (Openbook lab test)
  - 29/May, 10:00 ~ 13:00