Java IO

Exploring the java.io package and living to talk about it

Streams

- A stream is a sequence of data of undetermined length.
- Byte oriented and numeric data is written with output streams and read with input streams
- Text or character data is written with writers and read with … readers

Four Main Java classes

- Java.io.InputStream
- Java.io.OutputStream
- Java.io Reader
- Java.io.writer

- All are abstract classes with many subclasses.

Streams

- A stream is composed of discreet bytes, containing anything
- Processed best with a while loop looking for an end of stream marker
- Windows Ctrl-Z
- Unix Ctrl X

Subclasses

- BufferedInputStream
- ByteArrayInputStream
- DataInputStream
- FileInputStream
- FilterInputStream
- LineNumberInputStream
- ObjectInputStream
- PipedInputStream
- PrintStream
- PushbackInputStream
- SequenceInputStream
- StringBufferInputStream

- BufferedOutputStream
- ByteArrayOutputStream
- DataOutputStream
- FileOutputStream
- FilterOutputStream
- ObjectOutputStream
- PipedInputStream
- PipedOutputStream

Reading bytes of data

- The basic read() method of the InputStream class reads a single unsigned byte of data and returns the int value of the unsigned byte. This is a number between 0 and 255. If the end of stream is encountered it returns -1 instead, and you can use this as a flag to watch for the end of stream.
- public abstract int read() throws IOException

```
import java.io.*;
public class Echo {
public static void main(String[] args) {
 echo(System.in);
}
 public static void echo(InputStream in) {
     try {      while (true)
        {       // Notice that although a byte is read, an int
                // with value between 0 and 255 is returned.
                // Then this is converted to an ISO Latin-1 char
                // in the same range before being printed.
          int i = in.read();
                  // -1 is returned to indicate the end of stream
       if (i == -1) break;
                   // without the cast a numeric string like "65"
                   // would be printed instead of the character "A"
      char c = (char) i;
   System.out.print(c);
          }
   }    catch (IOException e) {
   System.err.println(e);     }
   System.out.println();
  }
```

Buffered Streams

- The java.io.BufferedInputStream and java.io.BufferedOutputStream classes buffer reads and writes by first storing the in a buffer (an internal array of bytes). Then the program reads bytes from the stream without calling the underlying native method until the buffer is empty. The data is read from or written into the buffer in blocks; subsequent accesses go straight to the buffer.

Readers and Writers

- The java.io.Reader and java.io.Writer classes are abstract superclasses for classes that read and write character based data. The subclasses are notable for handling the conversion between different character sets.

The Difference

- Input and output streams are fundamentally byte based. However readers and writers are based on characters, which can have varying widths depending on the character set being used. For example, ASCII and ISO Latin-1 use one byte characters. Unicode uses two byte characters. UTF-8 uses characters of varying width between one and three bytes. Readers and writers know how to handle all these character sets and many more seamlessly.

java.io.Reader class

- The methods of the java.io.Reader class are deliberately similar to the methods of the java.io.InputStream class.
  - However rather than working with bytes, they work with chars.
- The basic read() method reads a single character (which may may take between one and four bytes depending on the character set) and returns the character as an int between 0 and 65535.
- It returns -1 if the end of stream is seen.

The java.io.Writer class

- The methods of the java.io.Writer class are deliberately similar to the methods of the java.io.OutputStream class.
  - However rather than working with bytes, they work with chars.
- The basic write() method writes a single two-byte character with a value between 0 and 65535.

The InputStreamReader Class

- The java.io.InputStreamReader class serves as a bridge between byte streams and character streams: It reads bytes from the input stream and translates them into characters according to a specified character encoding.

More Java IO

Files

File Class

- File class is a general machine independent interface to the file system.
- File class is not for the contents of a file, but the file object
  - Directories are File Objects in java
- Several methods are available with this class.

File Class Methods

- getName()
- getPath()
- getAbsolutePath()
- getParent()
- isAbsolute()
- lastModified()
- IsFile()
- isDirectory()
- canRead()
- canWrite()

Directory entries

- String[] list()
- File[] listFiles()

- Both can use FileFilter interface
  - Boolean accept(File pathname)

File Streams

- FileInputStream and FileOutputStream
  - Byte
  - Reads and writes data as sequence of Bytes

File Writers and Readers

- Char
  - Reads and writes data as sequence Unicode characters

Random Access Files

- Random access files can be read from or written to or both from a particular byte position in the file. The position in the file is indicated by a file pointer.
- There are two constructors in this class:
- public RandomAccessFile(String name, String mode) throws IOException
- public RandomAccessFile(File file, String mode) throws IOException

- The first argument to the constructor is the file you want to access. The second argument is the mode for access. This should either be the String "r" for read-only access or the string "rw" for read and write access. Java does not support write only access. For example,
- RandomAccessFile raf = new RandomAccessFile("29.html", "r");

- The getFilePointer(), length(), and seek() methods allow you to determine and modify the point in the file at which reads and writes occur. Attempts to seek (position the file pointer) past the end of the file just move the file pointer to the end of the file. Attempts to write from the end of the file extend the file. Attempts to read from the end of the file throw EOFExceptions.

- public long getFilePointer() throws IOException
- public void seek(long pos) throws IOException
- public long length() throws IOException

- Reads and writes use methods that work identically to the methods of the DataInputStream and DataOutputStream classes, except that you can set the position at which the read or write occurs between calls to the read and write methods.