Jenny Hamer

CIS 36B

**Lab 7 – Questions : Chapter 11 *Using I/O***

**1. Why does Java define both byte and character streams?**

Byte streams allow for convenience in inputting and outputting bytes and are helpful with file I/O since they are used when reading and/or writing binary data. Character streams are designed for to input and output characters with ease, uses the Unicode international system, and in certain cases, is a more efficient method than byte streams.

**2. Even though console input and output is text-based, why does Java still use byte streams for this purpose?**

Java uses byte streams, because the predefined streams in the original specifications for Java (ie. System.in, System.out, System.err) used them and did not include character streams (though they were later added and can be used as wrappers).

**3. Show how to open a file for reading bytes. (Input by creating a FileInputStream object)**

FileInputStream(String *fileName*) throws FileNotFoundException // generic constructor

// filename 🡪 name of specific file to read from; if the file does not exist, a FileNotFoundException is thrown.

Ie. FileInputStream myFile = new FileInputStream(“test”);

**4. Show how to open a file for reading characters.**

FileReader charFile = new FileReader(“test”);

**5. Show how to open a file for random-access I/O.**

RandomAccessFile(String *fileName*, String *access*) throws FileNotFoundException // generic constructor

Ie. RandomAccessFile randoAccFile = new RandomAccessfile(“test”, “r”);

// using a RandomAccessFile object, the desired *fileName* and *access value*: “r” 🡪 **read**, “rw” 🡪 **read & write**, (and “rwd”, “rws”).

**6. How can you convert a numeric string, ie. “123.23” into its binary equivalent?**

It’s possible to convert a numeric string into its binary equivalent of Double or Float by using a type wrapper.

String numStr = “123.23”;

Double doub = Double.parseDouble(numStr);

**7 – 8. Programs: attached separately.**