Week 8: Topics

- Files
- Streams

Files

- Data stored in variables and arrays is *temporary*

  Lost when the local variable goes out of scope or when the program terminates

- For long-term retention of data, even after the program that creates the data terminates

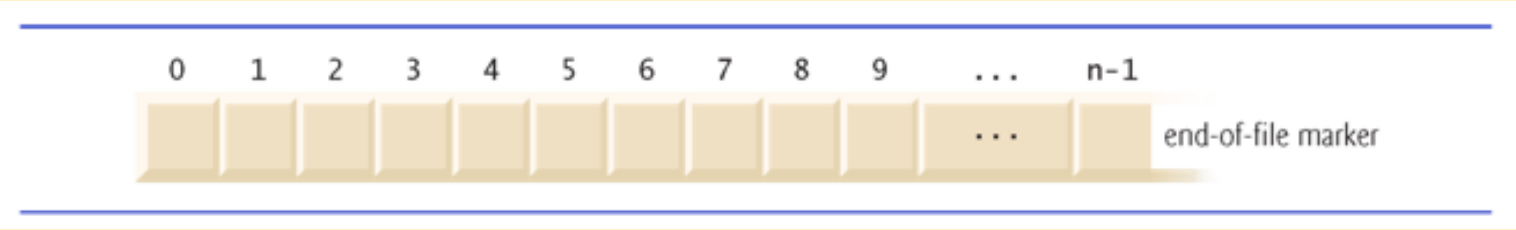
  Computer make use of files

- Computers store files on *secondary storage* devices
  - Hard disks, flash drives, DVDs etc.

Files and Streams

- Java views each file as a

  Sequential stream of bytes



- Every OS provides a mechanism to
  - Determine the end of a file

    i.e. an end-of-file marker

Byte-Based and Character-Based Streams

- **Byte-based streams** output and input data in its *binary* format

  A char is two bytes, an int is four bytes, a double is eight bytes etc.

- **Character-based streams** output and input data as a *sequence of characters*

  In which every character is two bytes

Byte-Based and Character-Based Streams

- Files created using byte-based streams are referred to as **Binary Files**

  Binary files can be read by a program that understands the file's specific content and ordering

- Files created using character-based streams are referred to as **Text Files**

  Text files can be read by a text editor

A quick tutorial:

What are the min and max values that can be stored in a `byte`?

What are the min and max values that can be stored in an `int`?

How many bytes would the following sequence of characters 2000000000 require?

Java Primitive Data Types

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Java Input and Output (I/O)

- The java.io package is used to process input and produce output
  - Including the handling of files

- Java creates *three* streams which are attached to the console automatically

System.out: standard output stream
System.in: standard input stream
System.err: standard error stream

e.g. System.err.println("error message");

Java `FileOutputStream`

- Used for writing primitive values (i.e. raw bytes) to a file

  Usually byte-oriented data

- `void write(byte[] ary)`

  Write ary.length bytes from the byte array to the file output stream

- `void write(byte[] ary, int off, int len)`
  - Write `len` bytes from the byte array starting at offset `off` to the file output stream

- `void write(int b)`

  Write the specified byte to the file output stream

FileOutputStream: Write a single byte

Blackboard: Week8/FileOutputStreamExample1

Problem:

Modify the code in the previous example in order to write a `String` to a file as a byte array.

FileOutputStream: Write a string

Blackboard: Week8/FileOutputStreamExample2

Java `FileInputStream`

- Used for reading primitive values from a file

  Usually byte-oriented data

- `int read()`
  - Used to read the byte of data from the input stream

- `int read(byte[] b)`

  Used to read up to b.length bytes of data from input stream

- `int read(byte[] b, int off, int len)`

  Used to read up to len bytes of data from the input stream

`FileInputStream`: Read a single byte

Blackboard: Week8/DataStreamExample1

Problem:

Modify the class `DataStreamExample` to read all bytes in the file. Assume the file `out.txt` contains the string “Welcome to Java”.

FileInputStream: Read all bytes

Blackboard: Week8/DataStreamExample2

Java BufferedOutputStream

- A buffer is used to store data, which is then written to a file

  Making it more efficient

- `void write(int b)`
  - Writes the specified byte to the buffered output stream

- `void write(byte[] b, int off, int len)`

  Write the bytes from the specified byte-input stream into a specified byte array, starting with the given offset

- `void flush()`

  Flushes the buffered output stream

BufferedOutputputStream: Write bytes

Blackboard: Week8/BufferedOutputStreamExample

Java `FileWriter`

- Used for writing character-oriented data to a file

  Method provided to convert string into byte array

- `void write(String text)`
  - Used to write a String into FileWriter
- `void write(char c)`
  - Used to write a `char` into FileWriter
- `void write(char[] c)`
  - Used to write a `char` array into FileWriter
- There is an equivalent `FileReader` class

  For reading streams of characters

Problem:

Create a `FileWriterExample` class using the `Java FileWriter` library class that writes a string to a file without the need to convert it into a byte array.

You can make use of the `FileOutputStream` class that we created earlier.

Java FileWriter

Blackboard: Week8/FileWriterExample

Java BufferedReader

- Read text from a character-based input stream
- `int read()`
  - Used for reading a single character
- `int read(char[] cbuf, int off, int len)`

  Used for reading characters into a portion of an array
- `String readLine()`

  Used for reading a line of text
- There is an equivalent `BufferedWriter` class

Problem:

Use the `BufferedReader` class to read the contents of a file (`out.txt`) and print the output to the standard output. You are required to make use of the `FileReader` class.

Java BufferedReader

Blackboard: Week8/BufferReaderExample