CS239

Nathan Sprague

Reading Quiz 1/3

Which of the following would *not* be considered binary files?

1. Java source code files (.java)
2. Java byte code files (.class)
3. Image files (for example, .jpg)
4. Audio files (for example, .mpg)

Reading Quiz 2/3

Which is a valid reason you should call `close` on a `PrintWriter` object when you are finished with it?

1. Leaving the file open too long may damage the hard drive.
2. Calling close triggers an integrity check that ensures the contents of the file are correct.
3. During writing, data may be buffered. Closing the file ensures that buffered data is transferred to disk.
4. Trick question! There is no good reason to close the file.

Reading Quiz 3/3

Most of the code samples in todays reading include a `Throws` clause like the following:

```
1 public static void main(String[] args) throws IOException
```

Why is this?

1. Otherwise the code would not compile. The main includes method calls that could raise checked exceptions.
2. The `throws` clause is a comment designed to inform the reader that the method may throw an exception.
3. Any method that includes a `try ... catch` block must also have a `throws` clause.

File I/O in Java - Key Classes for Output

- java.io.File↗ (Actually represents file (and path) name.)
- java.io.PrintStream↗ (System.out is an instance of this class.)
- java.io.PrintWriter↗
- java.io.FileWriter↗ (May be used, along with PrintWrite, to append to files.)

File I/O in Java - Key Classes for Input

- java.io.File↗ (Actually represents file (and path) name.)
- java.io.InputStream↗ (System.in is an instance of this class.)
- java.util.Scanner↗
- java.util.StringTokenizer↗

Clicker Question

What if this program is forced to exit before reading a "!"?

```
1   public static void main(String[] args) throws IOException
2   {
3       String currentLine;
4       Scanner scanner = new Scanner(System.in);
5       PrintWriter writer = new PrintWriter("out.txt");
6
7       while (true)
8       {
9           currentLine = scanner.nextLine();
10          if (currentLine.equals("!"))
11              break;
12          writer.println(currentLine);
13      }
14      writer.close();
15  }
```

1. There will be *no* file named "out.txt" (unless it already existed).
2. The file out.txt will exist, but will be completely empty.
3. The file out.txt will contain all of the lines that were entered before exiting the program.
4. The file out.txt will contain a set of random characters.