Java Lab 19: File I/O

This lab tests the hypothesis that buffered readers and writers are faster than unbuffered, and whether wrapping them inside Scanner/PrintWriter make any difference.

1. Create project Lab19 with Lab19Main and six ***static long*** methods – three that write a data file and three that read it. The first line in each of these methods should be

```
long startTime = System.nanoTime();
```

At the end of each method, declare another ***long*** set to nanoTime( ), subtract them, and return the resulting value. The nanoTime( ) method gives the current time in nanoseconds and returns a ***long*** integer.

Each method will have two parameters, a file name and an int, n. It will contain a ***for loop*** that counts from ***1 to n*** and either writes or reads a single char “A” to a file that many times. For example, if the parameters are “myfile.txt” and 1000, you’ll write “A” (or read that character) 1000 times.

Use the code on Slide 13 from today's lecture, "Scanner, cont." that shows some code with a Scanner wrapping some other stuff. Instead of reading or writing int's, the code will read or write char's.

2. ***printWriterTest( )*** will use the three-part wrapping: `FileWriter` inside a `BufferedWriter` inside a `PrintWriter`. Use the print(char) method to write out “A” inside the for loop.

3. ***bufferWriterTest( )*** will use two-part wrapping: `FileWriter` inside a `BufferedWriter`. Use the BufferedWriter write(char) method to write out 'A' inside the loop. Make sure you new up a BufferedWriter object and use it inside the for loop.

4. ***fileWriterTest( )*** will just use a `FileWriter`. Use the FileWriter write(char) method to write out 'A' inside the loop. Make sure you new-up a FileWriter object and use it inside the for loop.

5. ***scannerTest( )*** will use the three-part wrapping: `FileReader` inside a `BufferedReader` inside a `Scanner`. Before the for loop, put the line:

```
scanner.useDelimiter("");
```

to ensure single characters are read – Scanner really wasn't meant for this, so it's a bit of a kludge. Use

```
scanner.next().charAt(0)
```

to read single characters.

6. ***bufferedReaderTest( )*** will use two-part wrapping: `FileReader` inside a `BufferedReader`. Use read( ) inside the for loop.

7. ***fileReaderTest*** will just use a `FileReader`. Use read( ) inside the for loop.

8. In main, call each of the six methods in the order given using "test.txt" and 10000 as the parameters. Display the resulting values in a formatted printf( ) statement, with a label for each run (method name) and using %15d for the return value of the method (so the numbers get lined up).

9. Repeat #8 using 1000000.