Building Java Programs

Lab 6: Ch. 6: File Processing

lab document created by Marty Stepp and Stuart Reges

Today's lab

Goals for today:

- use `Scanner` and `File` objects to read input data from files
- understand the way a `Scanner` breaks input into tokens
- get more practice using `Strings`
- explore token-based and line-based input
- Where you see this icon, you can click it to check the problem in Practice-It! 

Scanner methods

Method name	Description
`next()`	reads and returns the next token as a `String`
`nextLine()`	reads and returns as a `String` all the characters up to the next new line (`\n`)
`nextInt()`	reads and returns the next token as an `int`, if possible
`nextDouble()`	reads and returns the next token as `double`, if possible
`hasNext()`	returns `true` if there is still a token in the `Scanner`
`hasNextLine()`	returns `true` if there is still at least one line left to be read in the `Scanner`
`hasNextInt()`	returns `true` if the next token can be read as an `int`
`hasNextDouble()`	returns `true` if the next token can be read as an `double`

Exercise 1: File Scanner declaration syntax

Which of the following choices is the correct syntax for declaring a `Scanner` to read the file `example.txt` in the current directory ?

a. ○ `Scanner input = new Scanner("C:\example.txt");`

b. ○ `Scanner input = new Scanner(new File("example.txt"));`

c. ○ `Scanner input = new File("example.txt");`

d. ○ `File input = new Scanner("/example.txt");`

e. ○ `Scanner input = new Scanner("C:/example.txt");`

Exercise 2-A: Tokenizing

How many tokens are in the following `String`? ____

```
welcome...to the matrix.
```

What are the tokens that the `String` breaks up into?

a. ○ `"welcome", "to", "the", "matrix"`
b. ○ `"welcome...to the matrix."`
c. ○ `"welcome...to", "the", "matrix."`
d. ○ `"welcome...", "to", "the matrix."`

Exercise 2-B: More tokenizing

How many tokens are in the following `String`? ____

```
in fourteen-hundred 92
columbus sailed the ocean blue :)
```

What are the tokens that the `String` breaks up into?

1. ○ `"in", "fourteen-hundred", "92"`
2. ○ `"in", "fourteen-hundred", "92", "columbus", "sailed", "the", "ocean", "blue", ":)"`
3. ○ `"in", "fourteen", "hundred", "92", "columbus", "sailed", "the", "ocean", "blue"`
4. ○ `"in", "fourteen-hundred", "92\ncolumbus", "sailed", "the", "ocean", "blue :)"`

Exercise 3-A: Scanner practice

The next couple problems are about a file called `readme.txt` that has the following contents:

```
6.7  This file has
several input
LINES!

   10 20
```

What would be the output from the following code, as it would appear on the console?

```
Scanner input = new Scanner(new File("readme.txt"));
System.out.println(input.nextLine());  //
System.out.println(input.nextLine());  //
System.out.println(input.nextLine());  //
```

Exercise 3-B: Scanner practice

Input file: `readme.txt`

```
6.7  This file has
several input
LINES!

   10 20
```

What would be the output if the code was changed to the following?

```
Scanner input = new Scanner(new File("readme.txt"));
System.out.println(input.next());  //
System.out.println(input.next());  //
System.out.println(input.next());  //
```

Exercise 3-C: Scanner practice

Input file: `readme.txt`

```
6.7  This file has
several input
LINES!

   10 20
```

What would be the output for the following code? If there would be an error, write **error** .

```
Scanner input = new Scanner(new File("readme.txt"));
System.out.println(input.nextDouble());  //
System.out.println(input.nextDouble());  //
```

Exercise 3-D: Scanner practice

Input file: `readme.txt`

```
6.7  This file has
several input
LINES!

   10 20
```

What would be the output for the following code? If there would be an error, write **error** .

```
Scanner input = new Scanner(new File("readme.txt"));
while (!input.hasNextInt()) {
    input.next();
}
System.out.println(input.nextInt());  //
```

Exercise 4: Words

- **Download** the following files Words.java and wordinput.txt to your machine and open them with jGrasp.
- The program is supposed to read the input file and count how many words it contains. Finish the program so that it runs properly. Don't forget to add `import` statements and a `throws` clause to the code.

Exercise 4 - answer

```
import java.io.*;     // for File
import java.util.*;   // for Scanner

public class Words {
    public static void main(String[] args) throws FileNotFoundException {
        int wordCount = 0;
        Scanner input = new Scanner(new File("wordinput.txt"));

        // your code goes here ...
        while (input.hasNext()) {
            String word = input.next();
            wordCount++;
        }

        System.out.println("Total words = " + wordCount);
    }
}
```

Exercise 5: Syntax errors

- The following Java program has 11 errors. Can you find them all?

```
1  public class StringOops {
2      public static void main(String[] args) {
3          Scanner console = new Scanner(System.in);
4          System.out.print("Type your name: ");
5          String name = console.nextString();
6          process(name);
7      }
8
9      public static void process(string "name") {
10         if (name == Whitaker) {
11             System.out.println("You must be really awesome.");
12         }
13         replace("a", "e");
14         toUppercase(name);
15         name.substring(0, 3);
16         System.out.println(name + " has " + name.length + " letters");
17     }
18 }
```

- **Copy and paste** the code into jGrasp and see if you can fix the errors.

Exercise 5 - answer

1. line 5: `nextString` should be `next`
2. line 9: `string` should be `String`
3. line 9: `name` should not be in quotes
4. line 10: `Whitaker` should be in quotes
5. line 10: cannot compare strings with `==`; must use `.equals`
6. line 13: cannot call `replace` without specifying a string object (`name`)
7. line 14: `toUppercase` should be `toUpperCase`
8. line 14: `name.` should come before `toUpperCase`, not passed as a parameter to it
9. line 14: must say `name =` to store the result of `toUpperCase`
10. line 15: must say `name =` to store the result of `substring`
11. line 16: must use parentheses `()` when calling `length`

Exercise 5 - Corrected version

```
public class StringOops {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Type your name: ");
        String name = console.next();
        process(name);
    }

    public static void process(String "name") {
        if (name.equals("Whitaker")) {
            System.out.println("You must be really awesome.");
        }
        name = name.replace("a", "e");
        name = name.toUpperCase();
        name = name.substring(0, 3);
        System.out.println(name + " has " + name.length() + " letters");
    }
}
```

Exercise 6: Debug ZipCode Case Study

In this exercise we will practice the jGRASP debugger using the Case Study example from the end of Chapter 6. To download this example, follow these steps:

1. Go to the class web page and click the "Textbook" link.
2. Find the section labeled "Code Files" and click the "code files" link.
3. This will bring you to a directory listing that includes an entry for each chapter. Click the link for "ch06".
4. You want to download and save the files `ZipLookup.java` and `zipcode.txt`. Right-click the file names and choose the option to save the link in whatever folder you have been using for lab work. Make sure to save them in the same folder.
5. Compile and run `ZipLookup.java` in jGRASP. You might try using your own ZIP code and a relatively small radius like 0.5 miles. The program takes a while to run because it has to search a large data file.

continued on the next slide...

Exercise 6 - jGRASP Debugger

- The debugger can be particularly helpful to understand a program that processes a large data file. In this exercise are going to debug the `ZipLookup` program.
- This program makes two passes through the data file. First it looks for the target zip code in the method named `find`. Then it shows all matches in the method named `showMatches`. We want to debug `showMatches`.
- The data file has a total of 43,191 zip codes in it, so it is not practical to have it display all of the data. That's why it's important to have good debugging skills to be able to selectively stop the program at certain points to see what is going on.
- Ask a TA for help if you have trouble setting or clearing break points or otherwise completing this exercise.

continued on the next slide...

Exercise 6 - jGRASP Debugger

```
What zip code are you interested in? 20500
And what proximity (in miles)? 0.3

20500: Washington, DC
zip codes within 0.3 miles:
    20045 Washington, DC, 0.26 miles
    20500 Washington, DC, 0.00 miles
    20501 Washington, DC, 0.27 miles
    20502 Washington, DC, 0.27 miles
```

- Now let's see what is true just before the `while` loop in the `showMatches` method executes. Set a break point on the `while` loop itself. Then debug to find `lat1` and `long1` (latitude and longitude of the White House ZIP code).

lat1	
long1	

continued on the next slide...

Exercise 6 - jGRASP Debugger

Clear your previous break point and set a new break point inside on the `printf` inside the `if`. Then hit the resume button that looks like a play button and fill in the table below with the values for `zip`, `lat2`, and `long2`.

zip	lat2	long2

Exercise 7: runningSum

Write a static method called `runningSum` that accepts as a parameter a `Scanner` holding a sequence of real numbers and that outputs the running sum of the numbers followed by the maximum running sum. For example if the `Scanner` contains the following data:

```
3.25 4.5 -8.25 7.25 3.5 4.25 -6.5 5.25
```

Your method should produce the following output:

```
running sum = 3.25 7.75 -0.5 6.75 10.25 14.5 8.0 13.25
max sum = 14.5
```

Click on the check-mark above to try out your solution in Practice-it!

Exercise 8: flipLines

Write a method named `flipLines` that accepts a `Scanner` for an input file and writes to the console the same file's contents with each pair of lines reversed in order. For example, if the file contains:

```
Twas brillig and the slithy toves
did gyre and gimble in the wabe.
All mimsey were the borogroves,
and the mome raths outgrabe.

The End
```

your method should produce the following output:

```
did gyre and gimble in the wabe.
Twas brillig and the slithy toves
and the mome raths outgrabe.
All mimsey were the borogroves,
The End
```

Exercise 9: countWords errors

```
1  // Counts the total lines and words in the given input scanner.
2  public static void countWords(Scanner input) {
3      Scanner input = new Scanner(new File("example.txt"));
4      int lineCount = 0;
5      int wordCount = 0;
6
7      while (input.nextLine()) {
8          String line = input.line();       // read one line
9          lineCount++;
10         while (line.next()) {             // count tokens in line
11             String word = line.hasNext;
12             wordCount++;
13         }
14     }
15 }
```

The above attempted solution to Practice-It problem "countWords" has 5 errors. Open Practice-It from the link above, copy/paste this code into it, and fix the errors. Complete the code so that it passes the test cases.

Exercise 9 - answer

1. line 3: should not declare another `Scanner` for the file
2. line 7: `nextLine` should be `hasNextLine`
3. line 8: `line` should be `nextLine`
4. line 10: need a second line `Scanner` to read the tokens of each line
5. line 11: `hasNext` should be `next()`
6. line 14: need to add 3 `println` statements to print line/word stats

Exercise 9 - solution

```
1  // Counts the total lines and words in the given input scanner.
2  public static void countWords(Scanner input) {
3      Scanner input = new Scanner(new File("example.txt"));
4      int lineCount = 0;
5      int wordCount = 0;
6
7      while (input.hasNextLine()) {
8          String line = input.nextLine();   // read one line
9          lineCount++;
10         Scanner lineScan = new Scanner(line);
11         while (lineScan.hasNext()) {      // count tokens in line
12             String word = line.next();
13             wordCount++;
14         }
15     }
16
17     System.out.println("Total lines = " + lineCount);
18     System.out.println("Total words = " + wordCount);
19     System.out.printf("Average words per line = %.3f\n", (double) wordCount / lineCount);
20 }
```

Exercise 10: coinFlip

Write a method named `coinFlip` that accepts a `Scanner` for an input file of coin flips that are heads (H) or tails (T). Consider each line to be a separate set of coin flips and output the number and percentage of heads in that line. If it is more than 50%, print "You win!". Consider the following file:

```
H T H H T
T t   t  T h  H
      h
```

For the input above, your method should produce the following output:

```
3 heads (60.0%)
You win!

2 heads (33.3%)

1 heads (100.0%)
You win!
```

Exercise 11: printDuplicates

Write a method `printDuplicates` that accepts a `Scanner` for an input file. Examine each line for consecutive occurrences of the same token on the same line and print each duplicated token along how many times it appears consecutively. For example the file:

```
hello how how are you you you you
I I I am Jack's Jack's smirking smirking smirking smirking smirking revenge
one fish two fish red fish blue fish
   bow  wow wow yippee yippee   yo yippee   yippee yay  yay yay
```

leads to the following console output:

```
how*2 you*4
I*3 Jack's*2 smirking*5

wow*2 yippee*2 yippee*2 yay*3
```

Exercise 12: mostCommonNames

Write a method `mostCommonNames` that accepts a `Scanner` for an input file with names on each line separated by spaces. Some names appear multiple times in a row. For example:

```
Benson Eric   Eric  Marty Kim  Kim Kim   Jenny  Nancy Nancy  Nancy  Paul  Paul
Stuart Stuart Stuart Ethan Alyssa Alyssa Helene Jessica Jessica Jessica Jessica
Jared  Alisa Yuki   Catriona  Cody   Coral   Trent Kevin  Ben Stefanie Kenneth
```

For each line, print the most commonly occurring name. If there's a tie, use the first name that had that many occurrences.

```
Most common: Kim
Most common: Jessica
Most common: Jared
```

Also return the total number of unique names in the whole file (e.g. 23 for the above input).

Exercise 13: frequentFlier

Write a method `frequentFlier` that accepts a `Scanner` for an input file of ticket type / mileage pairs and reports how many frequent-flier miles the person earned.

- 1 frequent flyer mile is earned for each mile traveled in coach.
- 2 frequent flyer miles are earned for each mile traveled in first class.
- 0 frequent flyer miles are earned on a discounted flight.

For example, given the input below, your method should return 15600 (2*5000 + 1500 + 100 + 2*2000).

```
firstclass  5000 coach   1500  coach
100 firstclass 2000  discount 300
```

If you finish them all...

If you finish all the exercises, try out our **Practice-It** web tool. It lets you solve Java problems from our *Building Java Programs* textbook.

You can view an exercise, type a solution, and submit it to see if you have solved it correctly.

Choose some problems from the book and try to solve them!