Building Java Programs

Chapter 6
Lecture 6-3: Section Problems

reading: 6.3 - 6.5

Recall: Line-based methods

Method	Description
`nextLine()`	returns the next entire line of input
`hasNextLine()`	returns `true` if there are any more lines of input to read (always true for console input)

- `nextLine` consumes from the input cursor to the next `\n` .

```
Scanner input = new Scanner(new File("<filename>"));
while (input.hasNextLine()) {
    String line = input.nextLine();
    <process this line>;
}
```


What's wrong with this?

```
public class Hours {
    public static void main(String[] args)
                                throws FileNotFoundException {
      Scanner input = new Scanner(new File("hours.txt"));
      while (input.hasNextLine()) {
         String line = input.nextLine();

         int id = line.nextInt();            // e.g. 456
         String name = line.nextLine();      // e.g. "Greg"
         double sum = 0.0;
         int count = 0;
         while (line.hasNextDouble()) {
            sum = sum + line.nextDouble();
            count++;
         }

         double average = sum / count;
         System.out.println(name + " (ID#" + id + ") worked " +
             sum + " hours (" + average + " hours/day)");
      }
   }
}
```


Corrected code

```
public class Hours {
    public static void main(String[] args)
                                throws FileNotFoundException {
      Scanner input = new Scanner(new File("hours.txt"));
      while (input.hasNextLine()) {
         String line = input.nextLine();
         Scanner tokens = new Scanner(line);
         int id = tokens.nextInt();          // e.g. 456
         String name = tokens.next();        // e.g. "Greg"
         double sum = 0.0;
         int count = 0;
         while (tokens.hasNextDouble()) {
            sum = sum + tokens.nextDouble();
            count++;
         }

         double average = sum / count;
         System.out.println(name + " (ID#" + id + ") worked " +
             sum + " hours (" + average + " hours/day)");
      }
   }
}
```


Recall: Tokenizing lines

- A String `Scanner` can tokenize each line of a file.

```
Scanner input = new Scanner(new File("filename"));
while (input.hasNextLine()) {
    String line = input.nextLine();
    Scanner tokens = new Scanner(line);

    <process the tokens on this line>;
}
```


`collapseSpaces` exercise

- Write a method `collapeSpaces` that accepts a file `Scanner` and writes that file's text to the console, with multiple spaces/tabs between words reduced to a single space.

file input	`four score and`
	`seven years ago our`
	
	`fathers brought forth on`
	`this continent`
	`a new nation`

console output	`four score and`
	`seven years ago our`
	
	`fathers brought forth on this continent`
	`a new nation`

Exercise solution

```
public static void collapseSpaces(Scanner input) {
    while (input.hasNextLine()) {
        String line = input.nextLine();
        Scanner words = new Scanner(line);
        if (words.hasNext()) {
            System.out.print(words.next());
            while (words.hasNext()) {
                System.out.print(" " + words.next());
            }
        }
        System.out.println();
    }
}
```


leetSpeak exercise

- Write a method `leetSpeak` that accepts two parameters:
  - a `Scanner` representing an input file, and
  - a `PrintStream` representing an output file.
- Convert the input file's text to "leet speak", where various letters are replaced by other letters/numbers. Output the leet version of the text to the given output file.
  - Preserve the original line breaks from the input.
  - Wrap each word of input in parentheses.

```
// example call
Scanner input = new Scanner(new File("lincoln.txt"));
PrintStream output = new PrintStream(new File("leet.txt"));
leetSpeak(input, output);
```


1337-speak replacements

Original character	'Leet' character
o	0
l *(lowercase L)*	1
e	3

Original character	'Leet' character
a	4
t	7
s *(at end of word)*	Z

Input file	Output file
four score and seven years ago our fathers brought forth on this continent a new nation	(f0ur) (sc0r3) (4nd) (s3v3n) (y34rZ) (4g0) (0ur) (f47h3rZ) (br0ugh7) (f0r7h) (0n) (7hiZ) (c0n7in3n7) (4) (n3w) (n47i0n)

Exercise solution

```
public static void leetSpeak(Scanner input, PrintStream output) {
   while (input.hasNextLine()) {
      String line = input.nextLine();
      Scanner lineScanner = new Scanner(line);
      while (lineScanner.hasNext()) {
         String word = lineScanner.next();
         word = word.replace("o", "0");
         word = word.replace("l", "1");
         word = word.replace("e", "3");
         word = word.replace("a", "4");
         word = word.replace("t", "7");
         if (word.endsWith("s")) {
            word = word.substring(0, word.length() - 1) + "Z";
         }
         output.print("(" + word + ") ");
      }
      output.println();  // preserve line breaks
   }
}
```


Mixing tokens and lines

- Using `nextLine` in conjunction with the token-based methods on the same `Scanner` can cause bad results.

```
23   3.14
Joe   "Hello world"
         45.2   19
```

  - You'd think you could read `23` and `3.14` with `nextInt` and `nextDouble`, then read `Joe "Hello world"` with `nextLine`.

```
System.out.println(input.nextInt());      // 23
System.out.println(input.nextDouble());   // 3.14
System.out.println(input.nextLine());     //
```

  - But the `nextLine` call produces no output! Why?

Mixing lines and tokens

- Don't read both tokens and lines from the same `Scanner`:

```
23   3.14
Joe   "Hello world"
         45.2   19

input.nextInt()                                  // 23
23\t3.14\nJoe\t"Hello world"\n\t\t45.2  19\n
  ^

input.nextDouble()                               // 3.14
23\t3.14\nJoe\t"Hello world"\n\t\t45.2  19\n
      ^

input.nextLine()                                 // "" (empty!)
23\t3.14\nJoe\t"Hello world"\n\t\t45.2  19\n
      ^

input.nextLine()                    // "Joe\t\"Hello world\""
23\t3.14\nJoe\t"Hello world"\n\t\t45.2  19\n
                ^
```

Line-and-token example

```
Scanner console = new Scanner(System.in);
System.out.print("Enter your age: ");
int age = console.nextInt();

System.out.print("Now enter your name: ");
String name = console.nextLine();
System.out.println(name + " is " + age + " years old.");
```

Log of execution (user input underlined):

```
Enter your age: 12
Now enter your name: Sideshow Bob
 is 12 years old.
```

- Why?
  - Overall input: `12\nSideshow Bob`
  - After `nextInt()`: `12\nSideshow Bob`
    `  ^`
  - After `nextLine()`: `12\nSideshow Bob`
    `    ^`

13