

hi

Building Java Programs

Chapter 6
Lecture 6-2: Line-Based File Input

reading: 6.3 - 6.5

Copyright 2010 by Pearson Education

Hours question

- Given a file `hours.txt` with the following contents:


```
123 Kim 12.5 8.1 7.6 3.2
456 Ben 4.0 11.6 6.5 2.7 12
789 Jesse 8.0 8.0 8.0 8.0 7.5
```
- Consider the task of computing hours worked by each person:


```
Kim (ID#123) worked 31.4 hours (7.85 hours/day)
Ben (ID#456) worked 36.8 hours (7.36 hours/day)
Jesse (ID#789) worked 39.5 hours (7.90 hours/day)
```
- Let's try to solve this problem token-by-token ...

Copyright 2010 by Pearson Education

Hours answer (flawed)

```
// This solution does not work!
import java.io.*; // for File
import java.util.*; // for Scanner

public class HoursWorked {
    public static void main(String[] args)
        throws FileNotFoundException {
        Scanner input = new Scanner(new File("hours.txt"));
        while (input.hasNext()) {
            // process one person
            int id = input.nextInt();
            String name = input.next();
            double totalHours = 0.0;
            int days = 0;
            while (input.hasNextDouble()) {
                totalHours += input.nextDouble();
                days++;
            }
            System.out.println(name + " (ID#" + id +
                ") worked " + totalHours + " hours (" +
                (totalHours / days) + " hours/day)");
        }
    }
}
```

Copyright 2010 by Pearson Education

Flawed output

```
Susan (ID#123) worked 487.4 hours (97.48 hours/day)
Exception in thread "main"
java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:840)
    at java.util.Scanner.next(Scanner.java:1461)
    at java.util.Scanner.nextInt(Scanner.java:2091)
    at HoursWorked.main(HoursBad.java:9)
```

- The inner while loop is grabbing the next person's ID.
- We want to process the tokens, but we also care about the line breaks (they mark the end of a person's data).
- A better solution is a hybrid approach:
 - First, break the overall input into lines.
 - Then break each line into tokens.

Copyright 2010 by Pearson Education

Line-based Scanner methods

Method	Description
<code>nextLine()</code>	returns next entire line of input (from cursor to <code>\n</code>)
<code>hasNextLine()</code>	returns true if there are any more lines of input to read (always true for console input)

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

Copyright 2010 by Pearson Education

Consuming lines of input

```
23 3.14 John Smith "Hello" world
45.2 19
```

- The Scanner reads the lines as follows:


```
23\t3.14 John Smith\t"Hello" world\n\t\t45.2 19\n
^
23\t3.14 John Smith\t"Hello" world\n\t\t45.2 19\n
^
23\t3.14 John Smith\t"Hello" world\n\t\t45.2 19\n
^
```
- Each `\n` character is consumed but not returned.

Copyright 2010 by Pearson Education

bye

1

hi

Scanners on Strings

- A Scanner can tokenize the contents of a String:
Scanner **name** = new Scanner(**String**);
- Example:

```
String text = "15 3.2 hello 9 27.5";
Scanner scan = new Scanner(text);
int num = scan.nextInt();
System.out.println(num); // 15
double num2 = scan.nextDouble();
System.out.println(num2); // 3.2
String word = scan.next();
System.out.println(word); // hello
```

7

Mixing lines and tokens

Input file input.txt:	Output to console:
The quick brown fox jumps over the lazy dog.	Line has 6 words Line has 3 words

```
// Counts the words on each line of a file
Scanner input = new Scanner(new File("input.txt"));
while (input.hasNextLine()) {
    String line = input.nextLine();
    Scanner lineScan = new Scanner(line);
    // process the contents of this line
    int count = 0;
    while (lineScan.hasNext()) {
        String word = lineScan.next();
        count++;
    }
    System.out.println("Line has " + count + " words");
}
```

8

Hours question

- Fix the Hours program to read the input file properly:

```
123 Kim 12.5 8.1 7.6 3.2
456 Ben 4.0 11.6 6.5 2.7 12
789 Jesse 8.0 8.0 8.0 8.0 7.5
```
- Recall, it should produce the following output:

```
Kim (ID#123) worked 31.4 hours (7.85 hours/day)
Ben (ID#456) worked 36.8 hours (7.36 hours/day)
Jesse (ID#789) worked 39.5 hours (7.90 hours/day)
```

9

Hours answer, corrected

```
// Processes an employee input file and outputs each employee's hours.
import java.io.*; // for File
import java.util.*; // for Scanner

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 lineScan = new Scanner(line);
            int id = lineScan.nextInt(); // e.g. 456
            String name = lineScan.next(); // e.g. "Eric"
            double sum = 0.0;
            int count = 0;
            while (lineScan.hasNextDouble()) {
                sum = sum + lineScan.nextDouble();
                count++;
            }
            double average = sum / count;
            System.out.println(name + " (ID#" + id + ") worked " +
                sum + " hours (" + average + " hours/day)");
        }
    }
}
```

10

File output

reading: 6.4 - 6.5

11

Output to files

- PrintStream:** An object in the java.io package that lets you print output to a destination such as a file.
 - Any methods you have used on System.out (such as print, println) will work on a PrintStream.
- Syntax:**

```
PrintStream name = new PrintStream(new File("file name"));
```
- Example:**

```
PrintStream output = new PrintStream(new File("out.txt"));
output.println("Hello, file!");
output.println("This is a second line of output.");
```

12

bye

2

hi

Details about PrintStream

```
PrintStream name = new PrintStream(new File("file name"));
```

- If the given file does not exist, it is created.
- If the given file already exists, it is overwritten.
- The output you print appears in a file, not on the console. You will have to open the file with an editor to see it.
- Do not open the same file for both reading (Scanner) and writing (PrintStream) at the same time.
 - You will overwrite your input file with an empty file (0 bytes).

Copyright 2010 by Pearson Education 13

System.out and PrintStream

- The console output object, System.out, is a PrintStream.

```
PrintStream out1 = System.out;
PrintStream out2 = new PrintStream(new File("data.txt"));
out1.println("Hello, console!"); // goes to console
out2.println("Hello, file!"); // goes to file
```

- A reference to it can be stored in a PrintStream variable.
 - Printing to that variable causes console output to appear.
- You can pass System.out to a method as a PrintStream.
 - Allows a method to send output to the console or a file.

Copyright 2010 by Pearson Education 14

PrintStream question

- Modify our previous Hours program to use a PrintStream to send its output to the file hours_out.txt.
- The program will produce no console output.
- But the file hours_out.txt will be created with the text:

```
Kim (ID#123) worked 31.4 hours (7.85 hours/day)
Ben (ID#456) worked 36.8 hours (7.36 hours/day)
Jesse (ID#789) worked 39.5 hours (7.9 hours/day)
```

Copyright 2010 by Pearson Education 15

PrintStream answer

```
// Processes an employee input file and outputs each employee's hours.
import java.io.*; // for File
import java.util.*; // for Scanner

public class Hours2 {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("hours.txt"));
        PrintStream out = new PrintStream(new File("hours_out.txt"));
        while (input.hasNextLine()) {
            String line = input.nextLine();
            Scanner lineScan = new Scanner(line);
            int id = lineScan.nextInt(); // e.g. 456
            String name = lineScan.next(); // e.g. "Eric"
            double sum = 0.0;
            int count = 0;
            while (lineScan.hasNextDouble()) {
                sum = sum + lineScan.nextDouble();
                count++;
            }
            double average = sum / count;
            out.println(name + " (ID#" + id + ") worked " +
                sum + " hours (" + average + " hours/day)");
        }
    }
}
```

Copyright 2010 by Pearson Education 16

Prompting for a file name

- We can ask the user to tell us the file to read.
 - The filename might have spaces; use nextLine(), not next()

```
// prompt for input file name
Scanner console = new Scanner(System.in);
System.out.print("Type a file name to use: ");
String filename = console.nextLine();
Scanner input = new Scanner(new File(filename));
```

- Files have an exists method to test for file-not-found:

```
File file = new File("hours.txt");
if (!file.exists()) {
    // try a second input file as a backup
    System.out.print("hours file not found!");
    file = new File("hours2.txt");
}
```

Copyright 2010 by Pearson Education 17

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?

Copyright 2010 by Pearson Education 18

bye

3

hi

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

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

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

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

Copyright 2010 by Pearson Education 19

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
^

Copyright 2010 by Pearson Education 20

bye