File Input/Output

Most data is stored in files, not input by the user every time. In this activity, you'll learn the basics of reading and writing plain text files.

Manager: Recorder:

Presenter: Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Parse user input and string objects using a `Scanner`.
- Read a text file line by line, and extract data from it.
- Create a new text file, and output several lines to it.

Process Skill Goals

During the activity, students should make progress toward:

- Reading Java API documentation to explore a class. (Information Processing)

Model 1 Review of Scanner

The `java.util.Scanner` class is useful for reading and parsing text from various sources:

```
// Example 1
Scanner in = new Scanner(System.in);
while (in.hasNextLine()) {
    String line = in.nextLine();
    System.out.println(line);
}

// Example 2
String text = "1 fish 2 fish red fish blue fish";
Scanner sc = new Scanner(text);
System.out.println(sc.nextInt());
System.out.println(sc.next());
System.out.println(sc.nextInt());
System.out.println(sc.next());
```

Questions (10 min) Start time:

1. For each example above, describe what the `Scanner` is scanning.

a) Example 1: `new Scanner(System.in)`

b) Example 2: `new Scanner(text)`

2. Based on the documentation for `Scanner`, explain the following:

a) `in.hasNextLine()`

b) `in.nextLine()`

c) `s.nextInt()`

d) `s.next()`

3. Open *ScannerDemo.java* in Eclipse, and run the program. Enter three lines of input, and notice the output. Then press Ctrl+D, which is the keyboard shortcut for "end of file" (EOF).

a) In the Console, what color was the user's input?

b) In the Console, what color was the program's output?

c) What was the complete output of the program? (Note: Do not include the input lines.)

4. What effect did pressing Ctrl+D have on the program? Explain how you think EOF works.

5. Rewrite the code for Example 2 to output each *word* of the string using a `while` loop. Run your code to make sure it works.

Model 2 Reading from a File

The Internet Movie Database (IMDb) maintains information about movies, television shows, video games, and more—including their cast, production crew, trivia, ratings, etc.

Download the *title2020.tsv* file into your Eclipse project. It's a subset of all titles from the year 2020 and is based on the data available at https://www.imdb.com/interfaces/.

Create a new program named *IMDb.java*, and add the following lines to the `main` method:

```
File file = new File("title2020.tsv");
Scanner in = new Scanner(file);
for (int i = 0; i < 3; i++) {
    System.out.println(in.nextLine());
}
in.close();
```

Note: You will need to import `java.io.File` and `java.util.Scanner`.

Questions (20 min) Start time:

6. Explain the compiler error when attempting to construct the `Scanner`.

7. Explain two ways you can modify the code to handle this error. (*Hint:* Eclipse offers them as "quick fixes".) Which way is better?

8. Modify the program so that it compiles: 1) surround the "`new Scanner`" line with try/catch; 2) remove the auto-generated `TODO` comment; and 3) initialize the `in` variable to `null` (before the `try` block). Copy and paste the beginning on your `main` method (from the "`File file`" line to the end of the `catch` block) in the box below.

9. Run the program. What is the output of the `for` loop?

10. TSV stands for "tab-separated values". Explain the format of the *title2020.tsv* file:

a) What does the first line represent?

b) What do the remaining lines represent?

c) How are "column breaks" represented?

11. By default, `Scanner.next()` separates input by whitespace. Replace the `for` loop in your `main` method with the following code. What is the resulting output?

```
in.useDelimiter("\t");
in.nextLine();

for (int i = 0; i < 3; i++) {
    System.out.println(in.next());
}
```

12. Remove the `for` loop from your program. Write code that outputs the `tid`, `type`, and `title` of the first 5 titles that start with `"A"`. (*Hint:* Use a `while` loop, call `in.next()` to get each of the three values, and call `in.nextLine()` to advance to the next line.) Paste your code below, starting from the `while` loop.

Model 3 Writing to a File

The `java.io.PrintWriter` class is useful for writing text files:

```
File file = new File("results.tsv");
PrintWriter out = new PrintWriter(file);
// output text to the file...
out.close();
```

Questions (15 min) Start time:

13. Examine the documentation for `PrintWriter`. What methods can be used to output a string to the file?

14. Modify your code from Question #12 to output to the *results.tsv* file instead of to the screen. Summarize your changes below:

15. In general, is it easier to write code that reads a file or writes a file? Explain your reasoning.

16. Make sure the end of your `main` method closes both files. Why is it important to close files when you are finished with them?

17. (Optional) What is the difference between the `print` methods and the `write` methods in the `PrintWriter` class?