File Input/Output

Most data is stored in files, not input by the user. In this activity, you'll learn the basics of reading and writing 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 The Scanner Class

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 code above and the documentation for `Scanner`, explain the following:

a) `in.hasNextLine()`

b) `in.nextLine()`

c) `sc.nextInt()`

d) `sc.next()`

3. Open *ScannerDemo.java* in your IDE, and run the program. Enter three lines of input, and notice the output. Then press the keyboard shortcut for "end of file" (Ctrl+D on Linux/macOS, or Ctrl+Z on Windows).

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 "end of file" (EOF) have on the program? Explain how EOF relates to the `while` loop.

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.

Open *IMDb.java* in your IDE. This program attempts to read the file *title2020.tsv* (which should be in the same folder as *IMDb.java*). The *title2020.tsv* file is a subset of movies and TV shows from the year 2020 based on the data available at https://www.imdb.com/interfaces/.

Questions (20 min) Start time:

6. What is the compiler error on Line 8 of *IMDb.java*?

7. Explain two ways you can modify the code to handle this error. (*Note:* The Eclipse IDE 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) initialize the `in` variable to `null` before the `try` block. Summarize the beginning of your `main` method (from the "`File file`" line to the end of the `catch` block):

9. Run the program, and describe 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?

d) How many rows/lines are in the file?

11. Replace the `for` loop in your `main` method with the following code:

```
int count = 0;
while (count < 1) {
    String tid = in.next();
    String type = in.next();
    String title = in.next();
    if (tid.equals("6723592")) {
        System.out.println(tid + " is a " + type + " named " + title);
        count++;
    }
    in.nextLine();
}
```

What is the resulting output?

12. What is the purpose of `in.nextLine()` at the end of the `while` loop?

13. Modify the code to find the first 5 titles that start with `"A"`. Describe your changes below:

14. (Optional) How could you modify the program to count the total number of lines read?

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:

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

16. Modify your code from Question #13 to output to the *results.tsv* file instead of to the screen. Summarize your changes below:

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

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

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