Reading Text Files and Handling Exceptions

CS 121

Department of Computer Science
College of Engineering
Boise State University

October 17, 2016

Topics

- Files
- Exceptions
- Reading/Writing text files
- String Tokenizer

Files

- The `File` from the `java.io` allows us to interact with files on the system.

```
File image = new File("photo.jpg");
if (image.exists()) {
    System.out.println("Image size: " + image.length()
        + " bytes");
} else {
    System.out.println("File does not exist: " + file);
}
```

- Example: FileTest.java

Reading Text Files (1)

- A `File` object can be passed to the `Scanner` constructor, which allows to read from a file by iterating through it.
- Scanner can be used to read a file line by line or token by token. Here is a code snippet for reading a file line by line:

```
Scanner fileScan = new Scanner(new File("input.txt"));
while (fileScan.hasNextLine()) {
    String line = fileScan.nextLine();
    // do something with the line
}
fileScan.close();
```

Using the Scanner

- The `Scanner` class is an iterator, meaning it allows you to process a collection of items one at a time.
- Because it implements the `Iterator` interface, it has the following methods defined.
  - the `hasNext` method: returns true if there is more data to be scanned.
  - the `next` method: returns the next scanned token as a string.
- The `Scanner` class also has variations on the `hasNext` method (such as `hasNextInt`, `hasNextLine`).

Exceptions

- Java has a predefined set of exceptions and errors that may occur during program execution.
- When we use a `Scanner` to open a file, it is possible to get an exception thrown because the file wasn't found.
- We have two choices for an `Exception`:
  - try and catch: Use a try-catch statement to handle the exception in the method.

```
try {
    ...
} catch (FileNotFoundException e) {
    // print or handle appropriate error
}
```

  - throw: The other option is for the method to pass on the exception to the calling method using the `throws` clause

```
public static void readFile(File file)
    throws FileNotFoundException
{
    ...
}
```

Reading Text Files: Examples

- Example: FileReading.java
- Example: ListFileWords.java
- Suppose we wanted to read and process a list of URLs stored in a file.
- One scanner can be set up to read each line of the input file until the end of the file is encountered.
- Another scanner can be set up for each URL to process each part of the path.
- Example: URLDissector.java
- Example on writing text files: FileWriting.java

Scanner Delimiter

- The default `Scanner` works well for reading input separated by whitespace, but sometimes we want to break strings on different characters.
- For example, we may want to break the string

  ```
  scheme, and the "plan" (for us)
  ```

  into

  ```
  scheme
  and
  the
  plan
  for
  us
  ```

- The `Scanner` class allows an application to break a string based on additional characters.

Scanner Delimiters

- A set of delimiters (the characters that separate words) may be specified for a `Scanner` object after it is created using the `useDelimiter` method.
- Delimiter characters themselves will not be treated as tokens.
- Once the delimiters have been set, you can iterate over the words using the `hasNext` and `next` methods

```
Scanner scan = new Scanner(text);
scan.useDelimiter(delimiters)
while (scan.hasNext()) {
    String token = scan.next();
    // do something with the word
}
```

- Example: UseScannerDelimiter.java

In-class Exercises

- Write a program that counts the number of lines in a given text file named `list.txt`.
- PP 4.18: Write a program that compares two text files line by line and prints all the differences.
  - Example: file-io/FileDiff.java
  - Sample input files: `Bill-of-Rights.txt`, `Bill-of-Rights-new.txt`

Exercises

- Read Section 4.6.
- **Recommended Homework**:
  - Projects: PP 4.16, 4.19.