Programming Languages and Techniques (CIS120)

Lecture 30

April 1, 2013

Histogram Design Exercise

Announcements

- HW09: Spellchecking is available now
  - Due Tuesday, April 9th at 11:59:59pm.

Design Example: Histogram.java

A design exercise using java.io and generic collections libraries.

Problem Statement

- Write a command-line program that, given a filename for a text file as input, calculates the frequencies (i.e. number of occurrences) of each distinct word of the file. The program should then print the frequency distribution to the console as a sequence of "word: freq" pairs (one per line).

Histogram result:

The : 1
Write : 1
a : 4
as : 2
calculates : 1
command : 1
console : 1
distinct : 1
distribution : 1
e : 1
each : 1
file : 2
filename : 1
for : 1
freq : 1
frequencies : 1
frequency : 1
given : 1
i : 1
input : 1
line : 2
number : 1
occurrences : 1
of : 4
one : 1
pairs : 1
per : 1
print : 1
program : 2
sequence : 1
should : 1
text : 1
that : 1
the : 4
then : 1
to : 1
word : 2

Interactive Demo

Histogram.java and WordScanner.java

Java Pragmatics Cheat Sheet

- Program entry point: `public static void main(String[] args)`
  - Command-line arguments are passed in the String array given to main.
  - Create a "Run Configuration…" to specify them with eclipse. For the Histogram demo: on the "Main tab" specify "Histogram" as the "Main class" and then under the "Arguments" tab give the filename under "Program arguments"
- Generic types cannot be instantiated by primitive datatypes (e.g. int, boolean); instead you must use “wrapper” classes (e.g. Integer, Boolean)
  - Java will automatically convert primitive values to wrapped objects.
  - See java.lang.Integer, java.lang.Character
  - This is a “kludge” due to Java’s history; generics weren’t added until long after the Java virtual machine was standardized…
- When creating an object of generic type, don’t forget to give type parameters: e.g. `new TreeMap<String, Integer>()`

Java Pragmatics Cheat Sheet

- Static fields and methods are "global" variables attached to a class name.
  - e.g. Character.isLetter(int c)

- Classes can be nested: e.g. `Map.Entry<K,V>`

- Abstract classes can't be instantiated, but they make good types.
  - Libraries use abstract classes to encapsulate shared algorithms.

- Calls to overloaded methods and constructors are determined by the number of arguments and their static types.

- Many I/O methods can fail by throwing an *exception*.
  - Exceptions are for unusual situations: File does not exist, Disk is full, etc.
  - Code that calls such methods can handle the error using:
    `try {…} catch (Exception e) {…}`