Acuitus Candidate Sample Projects

We want to see sample Java code that accurately represents your style and skills. This code should show some level of complexity, whether it be a particular algorithm or a multitude of parts working together in a complete application.

You have several options for giving us a project:

1) Tackle a small-scale problem where a properly designed software solution would improve the efficiency of a certain task.

You've probably run into the situation many times where you'd note that a piece of software would save you time and aggravation, only to complete the task manually due to other constraints. Well, now could be the time to take care of the problem. The result could be a freeware or shareware program that could save others time as well.

Past example:
- An image editor for a game mod. Allowed for 2D polygonal graphics editing (simple color-filled shapes defined by arbitrary vertices on user-selectable layers). The game required 36 rotations of the image in a 2D plane; the program generated these automatically.

2) Write a program to try out features of Java that you'd like to become more familiar with.

One of the benefits Java brings to coding is a substantial set of well-documented and -tested libraries so you don't have to reinvent the wheel. This is your chance to see how they work in a self-contained environment.

Past examples:
- A 3D engine for spinning a text string around the X, Y, and Z axes, written atop Java 2D, to examine drawing performance issues and learn the API.
- An image sequence to video converter. Given 30GB of JPEGs captured at various frame rates and resolutions, the application would fill in dropped frames and scale smaller images to a target resolution, finally wrapping the result in a Motion-JPEG video. Using features of NIO, processing time dropped from 6 hours to 5 minutes.

3) Contribute to an open-source project.

Often these projects come very close to satisfying your needs but are missing that special "something", be it a feature or layer of polish. Ideally, you would make a fairly localized change or addition. We'd need to see CVS/SVN logs and diffs to identify your work.

4) Implement the following sample spec.

This project provides good exposure to many of the Java language features and APIs. It will also result in a practical, usable application.

LinkChecker

How often have you come across a broken link on a web page? We're looking to eliminate this problem with the creation of the LinkChecker.

Given a starting URL, the program will download the HTML page, scan it for links, and then download the link targets. Information about working and broken links for the starting page will be presented to the user.

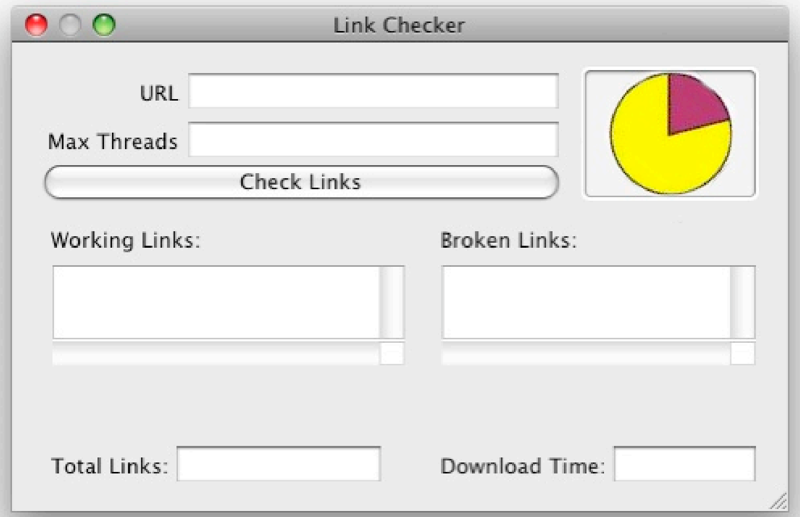
The process of checking links is, of course, complicated by the fact that connections can be flaky. You'll want to be on the lookout for typos and HTTP error codes and to have a maximum download timeout of approximately 10 seconds per link. A properly working link will allow you to fully read the data, without throwing errors, within the time limit.

Since there could be dozens of links on a page, we'll want to download them in parallel. Because it's considered bad form to monopolize concurrent connections on web servers, we'll have a parameterized cap on the number of threads that can be downloading simultaneously (default to 5).

While a command-line program is a great starting point, you'll probably want a GUI to increase usability. As your downloads succeed or fail, you'll want to update the UI in real time with the status of the links. Dropping links into working and broken lists would be sufficient. Alternatively, you could track links in a table.

We'll also want an indication of progress. We'll get that through a pie chart, segmented by the number of links on the original page. This will update live, colorizing a slice when a download completes. You could also have it reflect status: starting with a white pie, as a thread begins to download, a slice is colored yellow; on completion, the slice turns green or red for working or broken, respectively.

Example UI:



This specification is intentionally short on details, leaving you tremendous flexibility in implementation. Java provides a number of technologies suited to the task, so consider this an opportunity to explore and have fun.

Also, you don't need to be constrained to the minimum requirements (a command-line program with a progress-indicating GUI). You have several pieces of information that could also be presented to the user (and analyzed), such as a link's content type, size in bytes, time to download, and status (malformed URL, downloading, timeout, success, etc.). Also, Java provides ways of rendering link targets, including HTML and images.