Assignment 1: Fotoshop image manipulation workbench

This assignment must be submitted by through Moodle by **23.59, Monday Week 7, 9 November 2015**. Make certain that, when unpacked, your files will compile and run in any environment (e.g. avoid absolute links to files that exist only on your machine).

Background

The small software company that you work for has decided to enter the image manipulation market with a product imaginatively named *Fotoshop*. The company hopes that this will be the first of a series of products, including "lite" and "pro" versions, all sharing an underlying structure. It is therefore essential that the code produced is *easy to maintain* and easy to *extend*. They also hope to sell the products into other, *non-English speaking* countries.

The company's initial plan is to produce a simple initial prototype that uses command line instructions to manipulate images. The commands are:

`open <file>`	Load an image in <file> as the current image.
`save <file>`	Save the current image to <file>.
`help`	Offer help (list of instructions, etc).
`quit`	Quit the workbench.
`look`	Report the status of the workbench: for each image loaded, gives the name of image, its input file, a list of operations that have been applied to it.
`mono`	Convert the current image to monochrome.
`rot90`	Rotate the current image 90°**.**
`script <file>`	Run a script of commands from <file>.

Future plans include:

- A cache of images being worked on. Each image in the cache should include the list of filters applied to it so that they can be undone. The interface to the cache is

`put <name>`	Add a copy of the current image to the cache as <name>.
`get <name>`	Replace the current image with a copy of the image <name> from the cache.

- An `undo` operation that removes the last filter (where applicable) from an image. Note that some operations such as `mono`, `save` etc cannot be undone.
- Support for internationalisation (see the Java Internationalization trail, http://docs.oracle.com/javase/tutorial/i18n/).
- Further filters such as `flipH` (flip the current image horizontally), `above` (place one image above another), `sidebyside` (place one image next to another), etc.
- A graphical user interface.

As team leader, you asked an intern from Parkwood Metropolitan University to produce the initial prototype with a simple command-line interface. The module page contains a link to a jar file of his prototype, `fotoshop-bad.zip`. You save a copy of this file to your file space, read his code and despair: it is dreadful. It clearly cannot be extended in its present form, let alone support any of your company's ambitions. Clearly, you are going to have to refactor it before it meets many of their goals.

Your tasks

There are three parts to this assignment. First, write a detailed critique of the prototype. Second, refactor the prototype and implement the missing features listed below in order to remedy the prototype's shortcomings. Third, write a report explaining your improvements. It is important that, as computer scientists and hopefully as future leaders in companies etc, you can write clear English as well as well structured code. Marks will be awarded for each of these three parts.

Read these instructions very carefully. In particular, pay attention to the requirements and future plans of the company explained in the brief above. Your solution *must* support these.

1 Critiquing the prototype (15%)

Reading and critiquing other people's code is a useful exercise. Your first task is to read the existing code, understand what it does, and where and why it does it badly. You may consult Chapter 7 of *Objects First with Java*, Barnes and Kölling, for ideas about good design. Bearing in mind the goals of the company as well as good software engineering practice, write a detailed report identifying the design and implementation flaws in the prototype. Structure your report as a list.

2 Refactor and add features (75%)

2.1 Refactoring

Refactor the workbench to remove the flaws of the prototype. Some of these have been mentioned in the lectures or in *Objects First with Java*: you may implement these. **Your implementation should demonstrate good design throughout and your framework must support all the company's plans for future developments**. Your code must be professionally written and will be assessed for:

- correctness
- design (particular consideration will be given to cohesion, coupling, maintainability, extensibility)
- support for the company's ambitions for its products.
- appropriate use of language constructs
- style (commenting, indentation, etc.)

2.2 New features

Add the following features from the company's wish-list.

- A cache of images
- An `undo` command
- Internationalisation support
  (see the Java Internationalization trail, http://docs.oracle.com/javase/tutorial/i18n/)

You don't need to add any further functionality (you will not get any extra marks) but you may do so if it helps to test your redesign. For example, after you have refactored the code, can you add a "`flipH`" command without making changes to *any* of the existing classes except for `CommandWords`?
Note: You must adhere to the user interface specified.

3 Report (10%)

Write brief notes on each of your improvements that explain why they improve over the original. For example, how do they reduce coupling or increase cohesion? How do they make the code more flexible or easier to extend? How do they support the company's plans? [I suggest that you keep a log book of what you have done and why as you go along.]

Submission and Assessment

You must submit the project through Moodle by **23.59, Monday Week 7, Monday 9 November 2015**.
You must include:

1. Your PDF critique of the prototype, `critique.pdf`
2. The source code and Javadoc documentation for all your code. If you submit a Netbeans project, make sure that it has included the source code, and not just the .class files.
3. Your PDF report on your improvements, `improvements.pdf`. This should also mention any known bugs in your solution (you will lose more marks for bugs that I find than ones that you report).

You must adhere to the submission requirements. Do not change the names of the files. Adhere to the user interface specified, i.e. do not change command names, or add a graphical user interface. **Submit the reports as PDF using the names specified above.** If you ignore these requirements, it is likely that my test and printing scripts will not work and you will lose marks.

Marking scheme

Marks will be awarded as follows: Critique 15%, Refactoring 75%, Report 10%.

Plagiarism and Duplication of Material

The work you submit must be your own. We will run checks on submitted work in an effort to identify possible plagiarism, and take disciplinary action against anyone found to have committed plagiarism.

Some guidelines on avoiding plagiarism:

One of the most common reasons for programming plagiarism is leaving work until the last minute. Avoid this by making sure that you know what you have to do (not necessarily how to do it) as soon as an assessment is set. Then decide what you will need to do in order to complete the assignment. This will typically involve doing some background reading and programming practice. If in doubt about what is required, ask a member of the course team.

Another common reason is working too closely with one or more other students on the course. Do not program together with someone else, by which I mean do not work together at a single PC, or side by side, typing in more or less the same code. By all means discuss parts of the assignment, but do not thereby end up submitting the same code.

It is not acceptable to submit code that differs only in the comments and variable names that have been used, for instance. It is very easy for us to detect when this has been done.

Never let someone else have a copy of your code, no matter how desperate they are. Advise someone in this position to seek help from their class supervisor or lecturer. Otherwise they will never learn for themselves.

The University of Kent provides guidance concerning plagiarism at http://www.kent.ac.uk/ai/students/whatisplagiarism.html.

There is also a web page provided by the School of Computing concerning plagiarism at http://www.cs.kent.ac.uk/teaching/student/assessment/plagiarism.local that answers some frequently asked questions concerning plagiarism and computing assignments.

You are reminded of the rules about plagiarism that can be found in the Programme Handbooks. These rules apply to programming assignments. We reserve the right to apply checks to programs submitted for assignment in order to guard against plagiarism and to use programs submitted to test and refine our plagiarism detection methods both during the course and in the future.

Richard Jones, October 2015