Assignment 4

Marking Guide

CompSci 230 S2 2015

Clark Thomborson

Total marks on this assignment: 20. This assignment counts 2% of total marks in COMPSCI 230.

Part 1: Reporting a Responsiveness Defect

1) **(8 marks)** *Assessed work:*

Compare the performance of A4v0 with Mandelscape, with respect to GUI responsiveness. You should test A4v0 in two modes. A4v0.0 is A4v0 being run with 0 worker. A4v0.1 is A4v0 with 1 worker.

If you notice a GUI-responsiveness defect, you should attempt to "reproduce" this defect, that is, you should attempt to find a reliable way to cause the application to exhibit this defect. For example, the spinner labelled "Max iter." may be unresponsive in either or both of these applications, and this unresponsiveness may be evident only in A4v0.1 but not in A4v0.0 (or vice versa). Note that a description of "what can go wrong" in a defective application doesn't reveal the conditions under which the defect can be reliably observed it – and developing a reasonably-concise but easily-understandable description of "how to observe" a responsiveness defect should be your primary focus when you are answering this question.

You may run across one or more correctness defects in this code. In particular, a SwingWorker may throw an uncaught exception which has the effect of killing off that worker but not the application, and the uncompleted task may result in visible defects in the display. Another known defect (which is also present in Tim's codebase) is that multiple mouse-clicks in rapid succession may not have the (desired and expected) effect of causing multiple zooms, but instead may have the effect of resetting the view to the default zoom. Please do your best to "work around" any correctness defects you observe. I will issue a new version of a4.jar only if it has fatal defects in correctness, and (after running it on a fast desktop and a slow laptop) I'm reasonably confident it has no fatal defects.

Rather than getting side-tracked into correctness defects, when answering this question you should focus your attention on responsiveness defects. Users generally expect the controls (= widgets or affordances) on their GUIs to always react, immediately, to their mouse-clicks and mouse-drags. In particular, it should always be possible for a user to "spin" a spinner, or to click a button – so you should concentrate your testing on the responsiveness of the three spinners and one button on the GUI of a4v0.

Submit: a paragraph discussing *one* responsiveness defect you discover in *either* version of the Mandelscape application. Your paragraph should

- describe a specific defect (2 marks), and
- it should also describe a reliable method for reproducing this defect in each of the three systems under test (Mandelscape, a4v0.0, a4v0.1; for 2 marks each).
- If the defect is not reproducible in a system under test, you should state this clearly.

Marking notes:

The defect description should be specific about *which* control of the display is unresponsive (1 mark); and it should include some description (not necessarily quantitative) of the *length of time* in which the app is unresponsive (1 mark).

The reproduction method should be described in enough detail that you'd be confident of applying it on any of the three versions (3 marks). For example, if the method involves multiple mouse clicks, then there should be some description (not necessarily quantitative) of the time between clicks. There should also be some description of the state of the application when the method is started, and what mouse click(s) are involved (e.g. attempting to increase the number-of-iterations spinner, after the initial display has been fully painted.

For each of the three versions, you should be in no uncertainty about whether the student's defect is reliably reproducible. No quantitative description is required however there should be some qualitative descriptor such as "reliable", "unreliable", "occasional", "unreliable" for each of the three versions (1 mark each * 3 versions = 3 marks)

Part 2: Thread Performance

2) **(6 marks)** *Assessed work:*

Discover an (approximate value for) the optimal number of workers for a4v0 on your platform, by determining the lowest value of "Number of workers" that reliably delivers near-optimal "mega-iterations per second" on the updates. Note that allocating slightly more than the optimal number of workers will not significantly affect the time required for an update, and allocating many more than the optimal number of workers will increase the overheads of task-formation and task-cancellation – possibly to the point of adversely affecting responsiveness.

To estimate the optimal number of workers, you should select a display size and "Max iter" value which causes your platform to spend about 4 seconds (= 4000 milliseconds) when computing an update. You should then adjust the Number of workers from 1 to 16, then down from 16 to 1, then from 1 up to 16 again, slowly enough to ensure that each update is completed before the next update is requested. Your console listing will be quite long – you should cut-and-paste it into a word-processing document, retaining the entire listing as a single document (for reference). Then you should edit-down the listing until it consists solely of reports on update-deletions of the following form: "Update number X deleted. There are 0 pending updates. Latency Y. Work rate = Z mega-iters per second." Note that if there are any pending updates in the completion reports on your experimental trace (for X = 1, 2, 3, … 15, 16, 15, 14, 13, …, 2, 1, 2, …, 15 16) then the reported work-rates are unreliable – because some `SwingWorker`s are doing useless work, thereby consuming CPU resources that are unavailable to the usefully-working `SwingWorker`s. Also note that any report of a 0 work-rate is referring to an update which had been cancelled. If your first attempt does not provide you with a complete experimental trace (due to some pending updates), you should collect a second experimental trace, adjusting the number of workers then waiting until the update is completed (as indicated in its console report) before making another adjustment to the number of workers. If your second attempt fails, you will have reproduced a serious correctness error in a4v0. If you have discovered a reproducible correctness defect, you should document it briefly, then you should construct a (partial) experimental trace which contains deletion records of updates which contain performance records of updates which weren't cancelled, weren't running concurrently with pending updates, and which didn't immediately follow an update that completed while another update was pending.

Produce two scatterplots from the (X, Y, Z) triples in your experimental trace. Your (X, Y) plot will indicate how update-completion latency varies as a function of the number of workers. Your (X, Z) plot will indicate how the max-iter performance varies as a number of workers. Consider what these plots tell you about the optimal number of workers (for this particular view of a Mandelbrot set, on your particular platform). Now shift your viewpoint on the Mandelbrot set, and choose a somewhat smaller or larger "Max iter" value, selecting the optimal number of workers, to get a rough indication of whether or not the "mega-iters per second" performance of your platform (when running with the optimal number of workers) is reasonably constant over viewpoint settings. (This is called a "sensitivity analysis" – you're discovering whether your finding is "sensitive" to parameters you weren't directly testing. In this case your experimental trace varied only the number of workers, but there are many other parameters which *could conceivably* have a significant effect on performance.)

Submit: your two scatterplots (1 mark each), accompanied by your discussion (4 marks). Your discussion should briefly describe any experimental difficulties (such as a correctness defect), and it should focus on *your* interpretation of your experimental findings regarding the optimal number of workers.

Marking notes:

The scatterplots should either be captioned, or have axis labels and a title, so that it is immediately apparent to the marker what experimental measurement (e.g. a latency or a workrate) is being plotted on the abscissa/vertical axis (1 mark), and what experimental factor (e.g. the number of workers) is being displayed on the ordinate/horizontal axis (1 mark)

The discussion should include some analysis of each scatterplot (1 mark * 2 plots = 2 marks), it should come to some reasonable and understandable conclusion about the optimal number of workers (1 mark), and it should make some comment about the unevenness or non-linearity of the plots (1 mark) or about some experimental difficulty such as an uncaught exception (resulting in a stack trace being printed to the console).

Many students will, I suspect, conclude that "the more workers you allocate, the faster this application will run". This is an accurate interpretation of the first derivative of the experimental data, in both plots. If a student notices that there is a "knee" in their performance plots – typically at 4 workers for a 4-core CPU – commend them for their insight. As discussed during my last lecture, the apparent performance gain of this application when there are more workers than cores is solely because these additional workers are able to "crowd out" the daemons in the JVM and the operating system. Heavily loading a CPU by "crowding out" the daemons is generally *not* a good idea when tuning an application for performance, as it will tend to decrease responsiveness (and even the reliability) of the runtime system; however such performance tuning is quite an advanced topic, and the mark for a "reasonable and understandable conclusion" should be awarded to students who assert that 16 workers is optimal.

Part 3: Injecting a Defect

3) **(6 marks) Submit:** A paragraph discussing the presence or absence of your part-1 responsiveness defect in `a4v1` (3 marks), and a paragraph discussing your performance findings on `a4v1` (2 marks) which refers (in some relevant and clear way) to a table or plot of your performance measurements (1 marks).

Marking notes:

The first paragraph (on responsiveness) should be easily understandable (1 mark), make a definite statement about the presence or absence of the defect (1 mark), and contain at least one qualitative or quantitative description of the

defect or the detection method (1 mark). For example, the defect may be described as being "reliably" exhibited by the method, or the response-latency might be characterised as being "several seconds".

The second paragraph (on performance) should understandably refer to (tabular or graphical) data from both scenarios (r1 and r2) (1 mark), from both versions (a4v1 and a4v1.1) (1 mark), for four levels of the "number of workers" factor (= 1, 5, 11, 16). The paragraph should explain how they went about drawing a conclusion from this complex, 3-dimensional, dataset (1 mark). For example, if a student presents the required data, and if they assert that some conclusion "is obvious" but they do not provide any explanation of their reasoning, you should award two marks for their data presentation and zero marks for the explanation.