CMPSC 111
Introduction to Computer Science I
Spring 2016

Lab 10
Assigned: March 30, 2016
Due: April 6, 2016 by 2:30pm

Objectives

To learn more about how to use iteration constructs, such as the `for` loop, when writing Java programs. In addition, to explore some of the advanced graphical features that Java provides. Finally, to learn how to conduct an empirical study that uses operating system timers to evaluate how the number of loop iterations influences the execution time of a Java program.

General Guidelines for Labs

- **Work on the Alden Hall computers.** If you want to work on a different machine, be sure to transfer your programs to the Alden machines and re-run them before submitting.
- **Update your repository often!** You should `add`, `commit`, and `push` your updated files each time you work on them. I will not grade your programs until the due date has passed.
- **Review the Honor Code policy.** You may discuss programs with others, but programs that are nearly identical to others' will be taken as evidence of violating the Honor Code.

Reading Assignment

Review Section 6.4 to learn more about the structure and behavior of the `for` loop construct provided by the Java programming language, paying close attention to the three key parts of the `for` loop's declaration. Additionally, please review Section 5.4 to refresh your knowledge of the `while` loop. To learn more about fractals, and the Mandelbrot set that we will visualize in this laboratory assignment, please study the following Web site available at `http://jonisalonen.com/2013/lets-draw-the-mandelbrot-set`. Students who have never explored the concept of fractals are also encouraged to review `http://en.wikipedia.org/wiki/Mandelbrot_set`.

Creating a New Directory and Starting the Project

After changing into the "`cs111S2016-share/`" directory, which contains our course's version control repository, you should type the command "`git pull`" to download the source code for this laboratory assignment. In your own "`cs111S2016-<your user name>`" repository inside the "`labs/`" directory, create a directory called "`lab10`". Using the method described in a previous laboratory assignment, please copy the source code from the share repository to your own repository. Now, change into the "`labs/lab10/`" directory, in your own Git repository, and use "`gvim`" to study the source code of the provided files. What methods do these classes provide? How do they work? While you do not need to understand the details of the provided source code, you should be able to add explanatory comments that highlight the basic points about how this program operates. Students who do not understand these two programs should ask the instructor for assistance.

Understanding and Empirically Evaluating a Fractal Generator

Once you have developed a basic understanding of the `MandelbrotBW.java` and `MandelbrotColor.java` files, you should try to compile and run these programs. Please notice that these programs do not produce any output in the terminal window. Instead, you should see that they create a graphics file in the same directory where they are stored. For instance, if you run the `MandelbrotColor` program you will see that it creates a `mandelbrot-color.png` file. If you want to view the contents of this file, you can type the command "`xdg-open mandelbrot-color.png`" in your terminal window. What do you now see on the screen? What are the characteristics of this image?

Please use `gvim` to display and edit the source code of the `MandelbrotColor.java` file, noticing that it declares an `int` variable called `max` and initializes it to the value of 1000. How does the value of this variable influence the time taken to create the Mandelbrot graphic? To answer this question, you can change the value of max to, for instance, 100, and recompile and run "`/usr/bin/time java MandelbrotColor`" in your terminal. After running this command, the number that is postfixed with the "`user`" label will give the amount of time need to create the fractal. As part of this assignment, you should set the `max` variable to take on the values of $10, 100, 1000,$ and, 10000, timing the execution with `/usr/bin/time` and using `xdg-open` to see how this changes the resulting visualization. Along with recording the execution time for each program configuration, you should save each distinct graphic with a unique name and upload it to your Bitbucket repository.

Now, find the line in the `MandelbrotColor.java` file that is written in the following fashion:

```
colors[i] = Color.HSBtoRGB(i/256f, 1, i/(i+8f));
```

This line of code configures the way in which `MandelbrotColor` will display the colors in the fractal. What happens when you change the value of "`256f`" to a different floating point value? Will changing this value influence the efficiency of the fractal generator? To answer these questions you should set this value to the values of $32, 64, 128, 256, 512,$ and, 1024—while keeping the value of `max` set to the default of 1000—and then again use `/usr/bin/time` to measure the program's performance. In addition to recording the running time for each program configuration, you should save each graphic with a unique name in your repository. As you vary both this value and the value of the `max` variable, you should determine how execution time varies and how the image changes.

Required Deliverables

For this assignment you should submit versions of the following deliverables through both the Bitbucket repository and in a signed and printed format; do not print the color Mandelbrot graphics.

1. Completed, fully commented, and properly formatted versions of the two source code files.
2. The black and white version of the Mandelbrot graphic, saved with the default file name.
3. Appropriately named versions of all of the color versions of the Mandelbrot graphics.
4. Data tables that show the performance timings associated with varying the two parameters.
5. A written report, saved in `time`, explaining the trends in your Mandelbrot data tables.
6. A written report, saved in `color`, discussing how the Mandelbrot graphic changes.
7. A written reflection, saved in `reflection`, about the challenges you faced during this lab.

Please make sure that you use the appropriate `git` commands to save all of these required deliverables in the "`lab10/`" directory of your Git repository hosted by Bitbucket.