CMPSC 111
Introduction to Computer Science I
Fall 2016

Practical 4
September 23, 2016
Due in Bitbucket by midnight of the day of your practical
"Checkmark" grade

Summary

To practice using interactive programs implemented in the Java programming language. Additionally, to learn more about how to declare variables of different data types and then to change values of these variables using expressions. To also further explore how to display graphics using the applet environment provided by Java. Finally, to continue practicing the use of the `git add`, `git commit`, and `git push` commands to upload Java source code and graphics to your Git repository.

Review the Textbook

To successfully complete this practical assignment, you should review Section 2.2's content that explains how to declare variables and to assign values to these variables. Since the source code that is provided to you as part of this assignment also solicits input from the user, students should also re-read Section 2.6's exposition of the methods provided by the `java.util.Scanner` class. Finally, study Sections 2.7 through 2.9 and their explanations of how to create static graphics in the Java programming language. Please see the instructor if you have questions about these topics.

Preparing for the Programming Task

Before you start enhancing the Java classes required by this assignment, you should separately type the commands "`cd cs111F2016`", "`cd cs111F2016-<your user name>`", and "`cd practicals`" in your terminal window. Once you are in the `practicals/` directory of your Git repository, you can type the command "`mkdir practical04`" to create a new directory for this assignment. You can run the `gvim` command from this directory when you are ready to begin to implement the required Java program. Please see the instructor if you have problems with these preparatory steps.

Now, in the course's "share" repository, after you type "`git pull`" command, go to the `practical04/` directory, where you will find two Java classes: `DisplayDrawingCanvas.java` and `PaintDrawingCanvas.java`. Using the graphical file browser or the terminal, copy the `practical04/` directory from the "share" repository to your own `cs111F2016-<your user name>` repository inside the `practicals/` directory. Before you move on to the next steps, make sure that your repository is clearly organized with separate directories for labs and practicals and each assignment.

Reading Input from the User

Please use the `gvim` text editor to carefully study the code in the `DisplayDrawingCanvas.java` file. What are the lines in this program that read input from the user and store the input values

in variables? How do these lines of source code work? Next, find the line of code that creates the `JFrame` and modify it so that it prints your name and the date. This is the only change that you need to make to this file! Please see the instructor if you do not understand the code in this file.

Painting with Complementary Colors

In this phase of the assignment, you will need to load the file `PaintDrawingCanvas.java` file into `gvim` so that you can understand the existing code in this file and then add your own enhancements to this class. It is important to note that this code refers to a variable declared in the `DisplayDrawingCanvas` class through notation such as `DisplayDrawingCanvas.redValue`.

The purpose of this program is to create a canvas that contains two colors in it. The first colored rectangle—which should take up exactly half of the page—should display the color that was requested by the user. For instance, if the user inputs the RGB value of $(255, 0, 0)$, then this region of the graphic should be filled with the color red. The second half of the image should be filled with the "complement" of the color requested by the user. For example, the complement of the red value of 255 is the value $255 - 255 = 0$ and the complement of the green value of 0 is $255 - 0 = 255$. Intuitively, taking the complement of an RGB should give you the color that is on the "opposite" side of the color wheel that you can view in the `gimp` program. So, what is the RGB value of the color red's complement? Does it appear on the opposite side of the color wheel?

As a means of practicing the use of variables and expressions, and the creation of graphics in Java, you should make some small additions to this program so that it fulfills its intended purpose. To accomplish this task, the first thing that you must do is add a call to the `page.fillRect` method so that it paints a rectangle of the color requested by the user. Next, you should store in the variable called `userComplementaryColor` a color that is the complement of the one requested by the user. This means that you will have to write in the form of a Java expression the simple equations that were intuitively explained in the previous paragraph. Now, the source code line "`page.setColor(userComplementaryColor);`" will cause the next shape to be filled with the complement of the color chosen by the user. Finally, you will need to again call the `page.fillRect` method with the parameters that will yield a rectangle correctly drawn in the second half of the image. Please see the instructor or a teaching assistant if you have questions about these steps.

Testing Your Finished Program

Once you think that you have added the necessary lines to the `PaintDrawingCanvas.java` file, you should repeatedly test you program to make sure that it is creating the correct graphical output. For instance, if the user inputs the RGB value of $(255, 0, 0)$, then what are the colors in the left and right sides of the graphic? To test your program, you may want to try several different RGB values that are listed in Figure 2.10 of your textbook. More advanced testing of your program is possible by picking an RGB value from one side of the color wheel in `gimp` and then checking that your program correctly displays its complement (the one "across" the color wheel) in the graphic. Does your testing reveal any limitations to our approach to calculating the complementary color?

Students who are interested in an additional challenge can investigate other ways to calculate complementary colors. Another option for enterprising students is to further investigate the theory of color and actually implement ways to display other types of colors derived from the one input by the user. For instance, you might try to display a color that is "analogous" to the user's.

Using Version Control Correctly

As you are typing your program in the `gvim` text editor, you should regularly save your files. Once you have created a preliminary version of your program and it compiles and runs as anticipated, you should use the "`git add`" command to stage it in your Git repository. Next, you can use the "`git commit`" command to save it in your local repository with a version control message. Finally, you can run "`git push`" to transfer your file to the Bitbucket servers. For this practical assignment, you do *not* have to hand in a hard copy of anything—just upload your Java program and an output produced by your program to Bitbucket using the appropriate `git` commands. Please review your "Git Cheatsheet" and talk with a member of the class, the course instructor, or a teaching assistant if you do not understand how to correctly use the Git version control system.

As in the past practical assignments, you should submit the final versions of your Java code and your output by midnight on the day of your practical. For this assignment in particular, you must turn in the completed versions of the `DisplayDrawingCanvas.java` and `PaintDrawingCanvas.java` files along with screenshots and terminal outputs associated with running the program three times. Please pick RGB values for input to the `DisplayDrawingCanvas` program that reveal that your calculation of the complementary color is working as well as is possible.

Reminder About the General Guidelines for Practical Sessions

Since this is one of our first practical assignments and you are still learning how to use the Java programming language, don't become frustrated if you make a mistake. Instead, use your mistakes as an opportunity for learning both about the necessary technology and the background and expertise of the other students in the class, the teaching assistants, and the course instructor.

- **Experiment!** Practical sessions are for learning by doing without the pressure of grades or "right/wrong" answers. So try things! The best way to learn is by intelligently experimenting.
- **Submit *Something*.** Your grade for this assignment is a "checkmark" indicating whether you did or did not complete the work and submit something to the Bitbucket-hosted Git repository using the "`git add`", "`git commit`", and "`git push`" commands.
- **Practice Key Laboratory Skills.** As you are completing this assignment, practice using the `gvim` text editor and the terminal until you can easily use their most important features.
- **Try to Finish During the Class Session.** Practical exercises are not intended to be the equal of the laboratory assignments. Even if you are simply a slow typist, you should still ideally upload a file, even a partially working one, by the end of the class period.
- **Help One Another!** If your neighbor is struggling and you know what to do, offer your help. Don't "do the work" for them, but advise them on what to type or how to handle things. If you are stuck on a part of this practical assignment and you could not find any insights in either your textbook or online sources, formulate a question to ask your neighbor, a teaching assistant, or a course instructor. Try to strike the right balance between asking for help when you cannot solve a problem and working independently to find a solution.
- **Review the Honor Code Policy on the Syllabus.** Remember that while you may discuss programs with other students in the course, programs that are nearly identical to, or merely variations on, the work of others will be taken as evidence of violating the Honor Code.