

Distributed: Monday, Feb. 5, 2007

Due: At the **start** of class, Monday, Feb. 12, 2007

Goal: To practice Java in the environment of your choice as a review of CSC 182, to think about how recursion and iteration can solve the same problem, and to acquaint yourself with your textbook and the on-line resources provided for this course, including my style and formatting expectations.

Before you begin:

- You have read Chapters 0 and 1 and Appendix B of your text and all Feb. 5 handouts (including this!).
- You have seen recursion before (for example, the standard way to teach quicksort or mergesort is by way of recursion in CSC 182).

Objectives:

- Successfully compile and run the Ratio program found in Chapter 1 of your text, with the following two changes. The code is at `q:\InstructorFiles\Chase_Gene\ds\labs\Ratio.java`.

1. Modify it to receive input from the keyboard, according to your “favorite” method from my handout on three methods for getting input from the keyboard.

2. Replace the gcd function that calculates the greatest common divisor of two numbers by an iterative solution (one involving looping instead of recursion). Mark clearly in your internal documentation and in your (external) report that you did so.

- Save the resulting Ratio.java and Ratio.class files in your student area on Drive Q: for this course. If you are using an Integrated Development Environment (IDE) like JCreator, then the class file and other files will be created automatically. Don't delete them!
- Submit a printed laboratory report using the format that I demonstrate with my “fake” lab report on the assignment that I describe in the file IOtest.java in that report. Notice that homework assignments in this course are not just program listings; the program listing is an appendix to the report.
- On another printed page, answer the following five questions. That document need not go on drive q:.

1. (Self Check Problem 1.1, page 26)

2. (Self Check Problem 1.2, page 26)

3. What do light bulbs and sockets have to do with the assignment to modify gcd? (See page 11.)

4. What in Java is the “contract” between a Java coder like you and the Java coder who uses your code? (See page 25.)

5. Why does the interface named Structure use arguments of type Object? (See pages 22-23, 25, and Self Check Problem 1.9 for hints.) Don't confuse the interface `Structure` with the package `structure`.

This is an **individual** assignment, not a team assignment.

[Find rubric on back!]

Rubric. I will be looking for the following things as I grade this assignment.

[40%] 1. Successfully write an iterative (i.e. looping) version of `gcd` for `Ratio.java`.

[5%] 2. Test all methods with a driver program, a *separate* class, say `RatioTest.java`. In other words, remove `main` from `Ratio.java` to `RatioTest.java` and expand it to test all methods of `Ratio`.

[5%] 3. Say `implements RatioInterface` in `Ratio`. Include in `RatioInterface.java` the methods that make a `Ratio` what it is: `equals`, `toString`, and so on. (See page 518 for interfaces.)

[10%] 4. **Use good style.** See the link to good style at the course web page! Do all your Java classes have author's name, date, and version? Do all Java methods have preconditions, postconditions, and side effects? [You may omit that this time, until I teach it.] If you modified software, say what you changed, and change the version number. If you found help on the Web or elsewhere, credit your sources. Use a monospaced font like `Courier` for computer programs. That allows you to align code vertically to be able to eyeball errors easily. I prefer output screens to be black on white, saving eyestrain and printer ink. You can change that from the `cmd` window's menu item `Properties`.

[7%] 5. **Present enough sample runs showing input and output.** Ideally enough to test everything. In practice, that's impossible. In this lab, at least one sample run should show that your `gcd` works by testing a fraction that needs to be reduced, like `10/15`.

[10%] 6. The five additional discussion questions on page 1 of this sheet are worth 2% each.

[1%] 7. Write dates with month names (e.g. 3 Feb 2006) because conventions differ by country about what `02-03-2006` might mean. Computer scientists should be aware of diversity issues, too! ☺

[2%] 8. **Comment your code.** For example, in `RatioInterface`, don't just list the methods. Say what they do. Later in the semester, we'll write preconditions (what assumptions on parameters of methods, and what do they mean) and postconditions (what does the return value mean) for each method, but I won't count that until I've done it explicitly in class. Without commenting, the interface is just syntax.

[0%] 9. Some of you may put `int compareTo (Object other)` in `RatioInterface`. Others may not, since `Ratio` already implement the `Comparable` interface, but if `Ratios` must have `compareTo`, then it would be better for `RatioInterface` to say so. I wish that `Comparable` were called `ComparableInterface` (my choice) or `IComparable` (Microsoft's choice), but I didn't invent the interface name.

[10%] 10. **Allow me to recreate your test environment.** Say where your Java files are found on Q:, and the name of your workspace if you used `JCreator`. If you used `JCreator`, I will run your workspace. If you do not, be sure to tell me which class contains `main()` and how you modified `CLASSPATH` to be what you needed. For those who use other environments, give me instructions for how to do things from the command line.

[10%] 11. **Your report** contains sections for problem statement, solution (results), limitations, and conclusions (discussion). Staple or clip your typed report together with the computer code in an appendix, as in my sample Lab 0. Put an electronic copy of the report on Q: