Objectives
To continue practicing the use of GitHub to access the files for a practical assignment and the use of a “terminal window” to compile and run Java programs. To get a deeper understanding of how to create your own classes and methods in Java. To experiment with the structure of different classes by studying the given Java programs and modifying them by adding more functionality to the given methods.

Reading Assignment
You should review Sections 4.1 through 4.3 of your book to learn more about writing your own classes, constructors and methods. As you review this material, try to make a list of questions about concepts that you do not yet fully understand. In addition to discussing these questions with the teaching assistants and the course instructor, try to answer them as you complete this assignment.

Accessing and Studying the Example Programs
To access the practical assignment, you should go into the #practicals channel in our Slack team and find the announcement that provides a link for it. Unless you provide the instructor with documentation of the extenuating circumstances that you are facing, not accepting the assignment means that you automatically receive a failing grade for it.

Now, navigate the src/ directory to find files titled “Octopus.java”, “Utensil.java,” and “Practical9.java”. Study these programs by following through with the program execution flow manually. Compile and run these programs, and make sure you understand why and how the output is produced.

Modify the Example Programs
This practical assignment invites you to modify the given programs in the following manner.

1. Edit the file Octopus.java and find the constructor in this class. In the constructor, there is one parameter octopusName, which contains a String. Change this by adding one more parameter, octopusAge, of type int. This is the age of the octopus. Assign this to the appropriate instance variable (imitating what was done for the name).

2. Edit the file Practical9.java and look for the place where variable ocky is defined to be a new Octopus. Add an “age” to this so that we are specifying two things, not one, in the construction. Delete or comment out the “ocky.setAge(10)” method call in the next line.

3. Recompile and re-run the program and see if it correctly provides the age you specified.
2. • Edit the file Practical9.java. Declare a second Octopus variable (don’t just change the name of the one that’s there—create another one) and assign it any name and age that you want.

• Create a second Utensil of any type you wish, imitating the declaration and initialization of spat. Assign a cost and a color to this utensil. Assign this utensil to the new Octopus you created.

• Print out the name, age, weight, and favorite utensil of your new octopus. Print out the type, cost, and color of your new utensil.

3. If you finish with the tasks above you can also change values for your new octopus and its new utensil by using set methods on the new octopus object.

Checking the Correctness of Your Programs

As in the past assignments, you are provided with an automated tool for checking the quality of your source code. To check your Java source code you can start with the use of GatorGrader, type the command “gradle grade” in your terminal window. If you do have mistakes in your assignment, then you will need to review GatorGrader’s output, find the mistake, and try to fix it.

Once your programs are building correctly, fulfilling at least some of the assignment’s requirements, you should transfer both of your files to GitHub using the “git commit” and “git push” commands. For example, if you want to signal that the src/main/java/practicalnine/Practical9.java file has been changed and is ready for transfer to GitHub you would first type “git commit src/main/java/practicalnine/Practical9.java -m “Your commit message” in your terminal, followed by typing “git push” and checking to see that the transfer to GitHub is successful.

When you use the “git push” command to transfer your source code to your GitHub repository, Travis CI will initialize a “build” of your assignment, checking to see if it meets all of the requirements. If your source code meets all of the established requirements, then you will see a green ✓ in the listing of commits in GitHub after awhile. If your submission does not meet the requirements, a red ✗ will appear instead. You should aim to finish practical assignments on the day that they are assigned; please see the instructor if you do not understand this policy.

Since this is another challenging practical assignment and you are still learning how to use the Java classes and objects, 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.

Deliverables

This assignment invites you to submit, using GitHub, the following deliverables. Because this is a practical assignment, you are not required to complete any technical writing.

1. A correct version of src/main/java/practicalnine/Practical9.java that meets all of the established source code requirements and produces the desired text-based output.

2. A correct version of src/main/java/practicalnine/Octopus.java that meets all of the established source code requirements and produces the desired text-based output.

Evaluation of Your Practical Assignment

Practical assignments are graded on a completion — or “checkmark” — basis. If your GitHub repository has a ✓ for the last commit before the deadline then you will receive the highest possible grade for the assignment. However, you will fail the assignment if you do not submit any work by the set deadline for completing the project. Please see the course instructor if you do not understand how practical assignments are graded or you do not know how to complete one of the specific tasks in this assignment.