Lab 0A: Java

CS 205

Lab objectives:

- To help you make the transition from writing programs in Python to writing object-oriented programs in Java.
- Import, edit and run a Java application in Eclipse.
- Use class fields, methods, and constructors.
- Use the ArrayList class.
- Use a static field shared by all class objects.
- Write test code to ensure that your code behaves as required.
- Apply pair-programming techniques to a task

Before starting this assignment, you should have read:

- *java4python*, skipping Sections 2.3 and 3.3, all of Chapter 4, and Sections 4, 5, and 8 of Chapter 5.
- Chapter 1 of *Object Oriented Design and Patterns*.

Follow the steps in this lab carefully and complete the assignments.

Transitioning from Python to Java

Java and Python have similarities and differences, so we first have to recognize that we're making a transition. Making this transition includes such things as recognizing:

- All code must reside within a class.
- Java is a typed language; a variable and its type must be declared before use. These declarations are somewhat different for primitive types versus objects.
- White space has no syntactical meaning. The beginning and end of classes, methods, and blocks of code are indicated by using braces (`{ ... }`). The end of a statement is indicated with a semicolons (`;`) (White space *is* important in helping to make your programs readable by others.)

1. Using the NoMachine software, Login to phoenix. Download the lab0aStarter zip file from the course web site on phoenix.

2. Launch Eclipse (In NoMachine, select Applications-Programming-Eclipse) and import two Java projects by selecting File-Import-General-Existing Projects into Workspace. Click Next. Make sure Select archive file is selected and browse to the downloaded zip file. Click Finish.

3. In the Average Project, open the two class files, `Average` and `AverageTester` and study each one.

   There are multiple errors in each class file. Starting with the `Average` class first, and then moving on to the `AverageTester` class, find and correct the errors in the program.

   You will find that there are:

   - Compile-time errors (syntax), such as missing semicolons, braces, and `import` statements, among some others.
   - Run-time errors, such as `NullPointerException`.
   - Logic errors, such as the average in `Average`'s `getAverage` method being computed incorrectly.

   Find and fix each error, writing a comment for each of them explaining what the error was and how you fixed it.

4. Once you have corrected the syntax errors, you can run the AverageTester main program by right-clicking AverageTester and select Run As-Java Application. The output will appear in the console.

5. Include the names of both members of the programming pair as a comment at the beginning of each of the two class files.

6. When you are finished, export the project (right-click on Average in the Package Explorer pane, select Export-General-Archive File) and submit the project's zip file in Canvas. Only one member of each programming pair has to submit the project.

Extending an Existing Java Program

1. Examine the two source files in the Lab0a project and verify that you understand all the code.

2. Run the GreeterTester main program by right-clicking GreeterTester and select Run As-Java Application. The output will appear in the console.

3. Observe the JavaDoc comments that are in the form `/** ... */`. These comments contain tags like `@param` and `@return` which will generate the appropriate documentation. We won't be actually generating the JavaDoc, but I want you to write your documentation in JavaDoc format as much as you can.

4. In the Greeter class add a sayGoodbye method. You can easily add the javadoc comments by right-clicking in the declaration line of the method and selecting Source-Generate Element Comment. Sorry, but you still have to type in the content of the comment. Add code in GreeterTester to test this method and also generate the new javadoc documentation that includes this method.

5. Add a method void swapNames(Greeter other) in the Greeter class that swaps the names of this greeter with another. What could go wrong with this method? Test that! Complete the javadoc.

6. Modify the code so that instead of always saying "Hello", the code will cycle through several separate greetings. To accomplish this you will add a private field which is an ArrayList of Strings. This ArrayList will contain the separate greetings. You will also add a count generator which is an instance of the ModCounter class included with the project. Because each greeter object doesn't need its own count generator, make the generator static so that it is shared among all Greeter objects. How do you know that the count generator is being shared among all your Greeter objects? Test that! How do you know that the count resets to 0 after the last greeting has been used? Test that! Complete the JavaDoc.

7. Make sure that the javadoc comments in your files include your names. (The javadoc `@author` tag is handy for this.) When you are satisfied with your project and documentation, export the project (right-click on Lab0a in the Package Explorer pane, select Export-General-Archive File) and submit the project in Canvas.