Lab 1 (Java Generics)

CSC 172 (Data Structures and Algorithms)
Fall 2022
University of Rochester
Due Date: Monday, September 12 @ 8:59am

Introduction

Every student must hand in their own work, but every student must list the name of their lab partner if any on all labs. The textbook and lectures present examples of the code necessary to complete this lab. Collaboration is allowed. You and your lab partner may discuss the lab with other pairs in the lab. It is acceptable to write code on the white board for the benefit of other lab pairs, but you are not allowed to electronically copy and/or transfer files between groups.

This lab has six parts. You and your partner(s) should switch off typing each part, as explained by your lab TA. As one person types the lab, the other should be watching over the code and offering suggestions. Each part should be in addition to the previous parts, so do not erase any previous work when you switch.

Objective

The objective of this lab is exploring Java Generics in more details. We will also explore Functional interface, a new addition to Java. After completing the lab, you should be confident to undertake any project/assignment involving Java generics. Also, this lab will teach you how you can convert any non-generic method into a generic one. In the later part of the course, we will not use generic methods very often, but having this experience under your belt would help you to convert those methods into generic one with ease.

Tasks

Sooner or later, every Java programmer should work through the official Java Generics tutoria `https://docs.oracle.com/javase/tutorial/java/generics/index.html/` for an introduction to Java Generics. You are **strongly** encouraged to do so on your own time. It will help a lot. What you will find is that most of the textbooks and tutorials you read are derrived from this tutorial. However, there is no hand in associated with this recommendation.

Consider the following code snippet, where the `printArray()` method is used to print out arrays of different types:

```
Integer [] intArry = {1, 2, 3, 4, 5 };
Double [] doubArry = {1.1, 2.2, 3.3, 4.4};
Character [] charArray = {'H','E','L', 'L', 'O' };
String [] strArray = {"once", "upon", "a", "time" };

printArray(intArry);
printArray(doubArry);
printArray(charArray);
printArray(strArray);
```

1. Write a Java program with a main method that includes the example code above. Implement the static `printArray()` method using an array of Objects for the parameter. Compile and run your code.

2. Comment out, but do not delete, the previous implementation of the `printArray()` method and implement `printArray()` method using method overloading. Write four versions of `printArray()`, one for each appropriate array type. Compile and run your code.

3. Comment out, but do not delete, the previous implementations of `printArray()`. Now write a single method that uses the Generic programming technique so that you have a single method supporting all the types and maintain type safety.

4. Using non-generic techniques, write a method `getMax()` that takes an array of type `Comparable` and returns the maximum element in the array

   [i.e. `"public static Comparable getMax(Comparable [] anArray)"`].

   Add the following code to your main routine:

   ```
   System.out.println("max Integer is: " + getMax(intArry);
   System.out.println("max Double is: " + getMax(doubArry);
   System.out.println("max Character is: " + getMax(charArry);
   System.out.println("max String is: " + getMax(strArry);
   ```

   Compile your code. Copy and paste any compiler warnings you get into a comment block at the head of your implementation of `getMax()`. Run your code.

5. Comment out, but do not delete, your previous implementation of `getMax()`. Using the generic techniques to specify super-class relationships, implement a type safe version of `getMax()`.

6. For solving the same problem, now, you need to use Function interface. (Something really cool... but only availble in Java 8)

   Create a functional interface (`java.util.Function`) `findMax` that takes a Character array as input and returns the maximum Character. You can refer `https://www.javabrahman.com/java-8/java-8-java-util-function-function-tutorial-with-examples/` for an introduction to Java Functional Interface.

Submission

Hand in the source code from this lab at the appropriate location on the Blackboard system at `learn.rochester.edu`. You should hand in a single zip (compressed archive) `Lab1.zip` containing your source code, README, and OUTPUT files, as described below.

1. A plain text file named `README` that includes your contact information, your partner's name, a brief explanation of the lab (a one paragraph synopsis. Include information identifying what class and lab number your files represent.), and one sentence explaining the contents of any other files you hand in.

2. A single Java source code file representing the work accomplished for sections 1-5 of this lab. All source code files should contain author and partner identification in the comments at the top of the file.

3. Java source code files (you may decide how many you need) representing the work accomplished in section 6 of this lab. All source code files should contain author and partner identification in the comments at the top of the file.

4. A plain text file named `OUTPUT` that includes author information at the beginning and shows the compile and run steps

Grading (100 pts)

Each section (1-6) accounts for 15 pts. Total 90 pts
10 pts for corrently naming the files, proper commenting, README and OUTPUT file.

Notes:

All labs are open book. You can get code snippets from the internet if you want (make sure you cite those properly). But that is not the purpose. We want you to sit together, think about an algorithm, and then implement it together with your partner.