Introduction to Java

CS9053 Section I2

Spring 2025

Assignment 5

Wednesday 6:00 PM – 8:30 PM

Prof. Dean Christakos

Feb 21st, 2025

Due: Feb 28th, 2025 11:59 PM

Part I: Abstracts

Create an abstract class Task that contains:

* Fields: String module, int time.
* An abstract method void performTask() executes the task.

Create three subclasses implementing the abstract method:

* DevelopModule
* DocumentModule
* TestModule

The performTask() method for each subclass prints out a message that says “Developing Module [module]”, “Documenting Module [module]”, and “Testing module [module]”, respectively, where [module] is the value of the module field in Task

Next, create an ArrayList of Task objects called jobQueue, using parameterization. This should ONLY be able to accept objects of type Task or a subclass of Task

Then, create an abstract class called Person. The Person class has an abstract method called startTask(String moduleName, int time) and a field called jobQueue, which is an ArrayList of Task objects.

The Person class will have a constructor that takes an ArrayList of Task objects, and you will pass in the jobQueue that you created earlier.

Finally, create three concrete subclasses of Person:

* Developer
* Tester
* TechWriter

Each has their own implementation of startTask(String moduleName, int time), which creates a DevelopModule, DocumentModule, and TestModule task, respectively, and puts it on the jobQueue. They will obviously need their own constructors that call the Person constructure with the jobQueue.

Create one Person of each subclass. Create a loop that creates 5 random tasks for each Person subclass (randomly generate a name that is a 4-digit numeric string and randomly generate a time that ranges from 1 to 10)

Loop through the jobQueue, called performTask() on each task in the queue, and then print out the total amount of time to process all of the tasks by summing up the time each task takes.

Part II: **Lambdas**

* Use a lambda expression with the Comparator<Task> interface to sort the jobQueue by age time.
* Use a lambda expression and Predicate<Task> to filter the list and create a new list containing only Tasks whose time is larger than 5.
* Print the sorted list and the filtered list.

Part III: Generics & Lambdas

* Create a class Triplet<A, B, C> parameterized with three values, first, second, and third with getters and setters as a toString representation. first ,second, and third should be fields of type A,B,and C, the parameters of the Triplet class
* Create an ArrayList of Triplet objects where first, second, and third are of type Double
* Sort the ArrayList by the value of the magnitude of the Triplet, , using a Lambda function
* Make Triplet Comparable. This means that Triplet should implement the Comparable interface where the compareTo method should take a Triplet and, based on the magnitude (in the case of numeric values) or some other metric (in the case that A, B, and C are themselves Comparable)
  + I admit that this is a bit weird when you get beyond numeric parameterizations of Triplet. Assume all parameters are Comparable when implementing compareTo in this case
* Sort the ArrayList using the sort method will null as an argument, so the Comparable implementation will be used