Tutorial 8: Command And Singleton Design Patterns

The Command and Singleton Design Patterns

Executive Summary

In this assignment, we'll investigate a variation of the Command pattern that is widely used, especially in distributed programming situations. In this task, you're going to fill in some missing pieces of a simple "chat" application. Here's the idea of the application:

You have a "client-side" application that can connect to a set of specified "friends." Once you're connected to the friends, each of them can do (arbitrarily) any of the following: (1) send you a text message; (2) make your device "beep"; (3) make your device "vibrate"; or (4) break off the connection.

This programming assignment requires some network programming (using sockets). Since this isn't a networking course, I've implemented these pieces for you. Yes, I'm well aware that many of you are not familiar with socket programming. I do believe that it is a skill you *should* have, so please do review what I've provided. You need to review this code, figure out what's going on, and be able to answer questions about it. Ask if you have questions.

Provided Files

You'll need to clone the GitHub repository below, which should contain ten java source files, all in the command package.

(https://utexas.instructure.com/courses/1266665/files/51930735/download?wrap=1)
https://github.com/ut-ee461l/tutorial-command-singleton (https://github.com/ut-ee461l/tutorial-command-singleton) (https://utexas.instructure.com/courses/1266665/files/51930735/download?wrap=1)

There are several provided files for this assignment, but most of them are REALLY simple. You will need to modify several of these files, but you shouldn't need to create any new classes or files. In fact, the amount of programming you have to do here is pretty minimal, but that doesn't mean the assignment is trivial. I've put comments in the code itself, but here's an overview of what you're looking at:

- ChatDriver.java -- This is your launcher. It's going to start up your chat client and all of your friends' chat servers. You invoke it (after a successful compile) as: "java command.ChatDriver <names>" For example, I might create connections to three friends by saying "java

command.ChatDriver Alice Bob Carole" You should be able to connect yourself to arbitrary numbers of friends, but you must provide at least one friend's name. **There is no need to modify this file in any way.**

- ChatServer.java -- This is the implementation of your friends. Read through it. Figure out what's going on. If you have questions, ask. **There is no need for you to modify this file in any way.**
- ChatClient.java -- This is the implementation of your client-side code. It sets up the connections to your friends using Sockets, but then delegates handling received messages to the ChatListeners (described next). **You will need to make one small change to this file.** The location of this needed change is marked.
- ChatListener.java -- One of these gets created on the client side for each friend connection you have. The ChatListener receives String messages from the friend. It needs to turn those messages into one of three types of commands (beep, vibrate, or console printing). You need to implement this. The locations of the needed additions are marked.
- Speaker.java, VibrationMotor.java, and Console.java -- These are object representations of "devices" on your client device. They're pretty self-explanatory (and really simple). However, we're asking you to refactor them. Specifically, they should be singletons. **You'll need to make changes to the classes and to anyone who references them** (i.e., ChatListener.java in this case).
- Command.java -- The Command interface. Exactly as presented in class. **There is no need for you to modify this file in any way.**
- SpeakerBeepCommand.java, MotorVibrateCommand.java, and ConsoleCommand.java - These are the command classes that implement the Command pattern. We've done SpeakerBeepCommand.java for you. You need to do the other two.

Your Tasks

1. Modify the provided files to implement the Command pattern. No, I do not recommend just cloning the repository and starting to code. I recommend reading through the provided files, reading through the below questions and exercises (and doing most if not all of them) before finalizing the implementation.
2. Modify the provided files to make the Speaker, VibrationMotor, and Console classes Singleton objects.
3. Provide a written report that answers the following questions/exercises.

The Report

Your report should (briefly but completely) answer the following questions or perform the stated exercise:

1. One of your tasks was to make the Speaker, VibrationMotor, and Console classes Singletons. Why did we do this (other than to exercise the Singleton pattern)? Asked another way, what could potentially happen if we did *not* implement these as Singletons?

2. OK. Let's say we decide we want to make the Chat application a little more "context-aware" and we want to implement a functionality that makes it possible for a user's preferences to prevent "notifications" (i.e., beeps, vibrations, or texts) while the user is in an important meeting. Describe (perhaps with code snippets) what changes you would make and where you would make them. **Do not implement this solution, just explain your design.**
3. Inside the run() method of ChatListener, you have to fill in the creation of the commands and do something with them that ultimately results in their execution. Why do you not just call execute() on the command that you just created? Hint: consider your answer for the previous question.
4. In *your own words*, what does it mean when I say (in ChatClient) that the queue is "thread-safe"?

What to Submit

Zip the whole thing back up and turn it in **here (https://utexas.instructure.com/courses/1266665/assignments/4876005)** . Name your file with FIRSTNAME_LASTNAME.zip, for example BRUCE_WAYNE.zip. You should turn in a zip file that's formatted like mine; when we unzip it, the top level directory should contain two things: your report, in a PDF document, and a single folder called "command" with all of your java source files inside of that folder (we don't want your class files). ***Without changing directories, I should be able to type the following two commands and have everything work properly:***

javac command/*.java

java command.ChatDriver Alice Bob Carole Dennis Emily

I should be able to type as many or as few names as I want (the program already supports this; just don't change anything in ChatDriver.java and you should be fine).