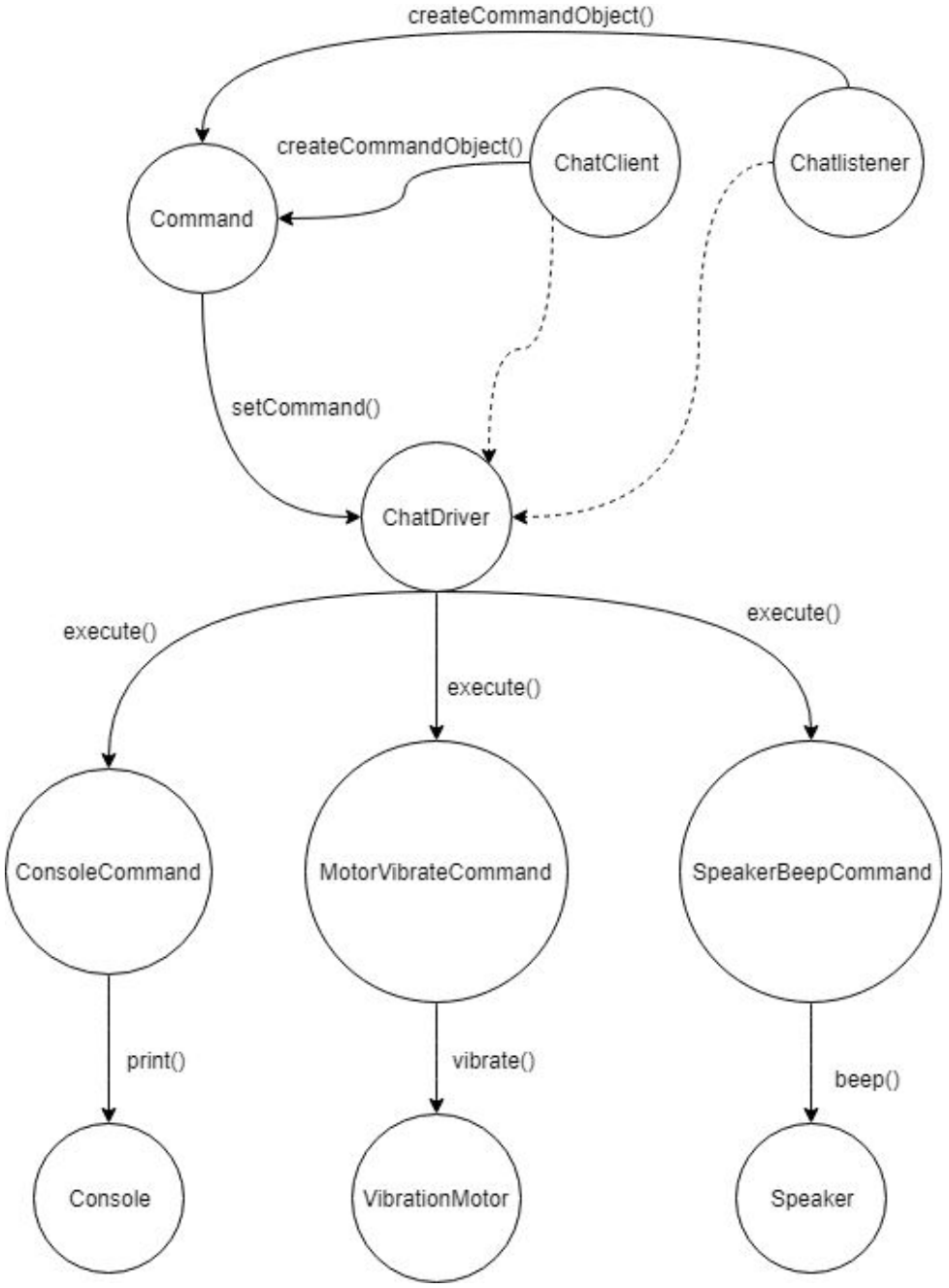
Zack Campbell
Zcc254

Lab 8: Command and Singleton Patterns Report

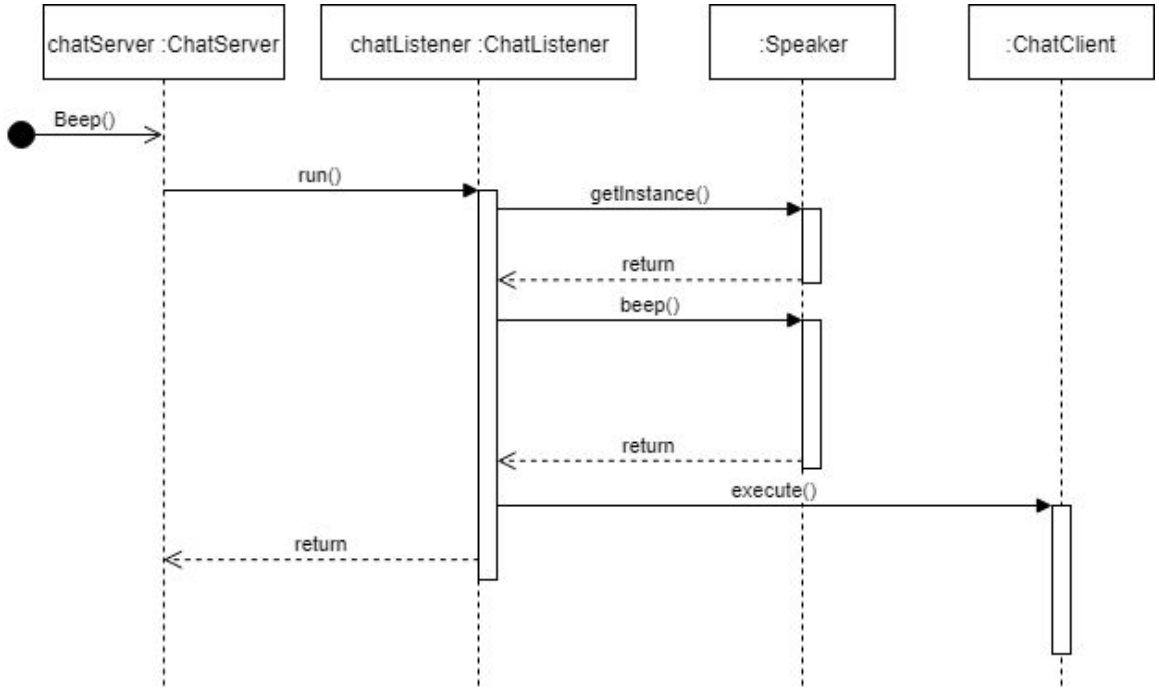
1.



Class	Command Pattern Element

ChatClient	Client
ChatListener	Client
Command	Command
ConsoleCommand	Command
MotorVibrateCommand	Command
SpeakerBeepCommand	Command
ChatDriver	Invoker
Console	Receiver
VibrationMotor	Receiver
Speaker	Receiver
execute()	execute()
setCommand()	setCommand()

2.



3. We implemented Console, VibrationMotor, and Speaker as singletons because there shouldn't be more than one instance of them ever in the program. If there was more than

one console, the clients wouldn't know which console to look at, so we have to use the singleton pattern to limit it to one console for everyone to share.

4. I would change the code in ChatClient in the while statement of the run() method to have an if statement hanging on a boolean. If the meeting boolean is set to true, then the notification can be ignored and we can skip execution of the command.
5. You don't just call execute() because the command object needs to determine if it has the instance of the object that it is calling or not. If two threads attempt to get the speaker, one will succeed and complete it's task successfully, but the other will attempt to execute a task on a null object.
6. The queue is thread-safe in ChatClient because it uses a LinkedBlockingQueue, which is a data structure inherently thread-safe in that it blocks threads blocks threads from trying to take out the same item from the queue at the same time and it prevents threads from adding two objects to the queue at the same time, but guarantees that one thread will succeed in both cases to avoid starvation.