**Smart house software engineering exercise**

In this exercise the goal is to implement a smart house system using high level languages: C#/Java/Python/TypeScript/Scala/etc.

The exercise will be divided into check points. Please use Github to keep track of changes and progress from check point to check point, each checkpoint should have a Git tag associated with its commit.

**High level description:** The smart house system exposes an HTTP API to receive queries & commands to either retrieve smart devices info & status or submit a command to a device, e.g. turn off/on. The smart house has the following devices:

* Bedroom TV
* Microwave
* Computer
* Living room TV
* Air Conditioner

Design your solutions well, there may be more devices and more queries & commands, adding them should result in minimal changes to existing code.

**Checkpoint 1**

Design and Implement a console app that receives one of the following CLI commands, design the commands syntax as you see fit:

* Query Bedroom TV status on/off
* Query Microwave status on/off
* Query Computer status on/off
* Query Living Room TV status on/off
* Query Air Conditioner status on/off
* Turn on/off Air Conditioner
* Turn on/off Bedroom TV
* Turn on/off Microware
* Turn on/off Computer
* Turn on/off Living room TV
* Turn on/off Air Conditioner
* Query Bedroom TV channel
* Query Living room TV channel
* Switch channel in Bedroom TV
* Switch channel in Living room TV
* Query Microwave degrees
* Query Air Conditioner degrees
* Set degrees and timer to Microwave, no more than 30 degrees
* Set degrees to Air Conditioner, between 10 and 30

The commands should be sent to the appropriate device and trigger the appropriate action, currently implemented as printing an appropriate console message stating the action executed for simplicity.

Notes:

* Do not concern yourself with concurrency at this point.
* Treat the console app execution as one-time CLI command execution, no session.

**Checkpoint 2:**

Design and implement an HTTP API to submit the same queries & commands to the smart house devices. Now instead of printing a console message when a command executes an HTTP response body with the same message should be returned. Devices now have a limit to #commands to be executed in each time window, and HTTP 429 should be returned when command is rejected for this reason:

* Bedroom & Living Room TV can receive at most 1 command every 3 seconds.
* Computer can receive at most 10 commands every 5 seconds.
* Air Conditioner can receive at most 2 commands every 4 seconds.
* Microwave is limitless, it’s an expensive microwave.

Notes:

* The HTTP API can run on localhost URLs without security considerations.
* Now do concern yourself with concurrency, the APIs may be invoked on same device concurrently. Consider carefully which behavior to implement and which API responses to return in such cases, after all HTTP has well known ways to handle concurrency conflicts.

**Checkpoint 3:**

Add logging to the execution of every query & command, both in console app and in the HTTP API, and add additional logs to facilitate troubleshooting of the smart house. A log record should have timestamp, severity, and message. The allowed severities are Error/Warning/Info/Verbose. The logging modules should be chosen differently for both:

* The console app should use console logging & text file logging.
* The HTTP API should use text file logging & event viewer (Windows only) logging.

**Checkpoint 4**:

Add configuration to control the following settings, so they can be modifying by changing a simple configuration file:

* Degrees settings of Microwave & Air Conditioner
* Commands rate limit of devices
* Logging types to use, e.g. console/text/event viewer; note not all can be used depending on execution in console app or in HTTP API
* Minimal logging verbosity to emit, e.g. we can set only logs that are Warning or more severe to be logged. The default should be to log with Info severity or higher.

You’re encouraged to explore and add additional features/infra: more devices, graceful API shutdown (waiting for all queries & commands to return a response while blocking new requests), more logging options (e.g. Splunk sink), whatever comes to mind.