Software Architectures

Assignment 4: Design patterns for micro-service architectures

Assistants: Humberto Rodriguez Avila, Kennedy Kambona
Email: {rhumbert, kkambona}@vub.be
Office: {10F719, 10F732}

Deadline: January 14th, 2018, 23:59

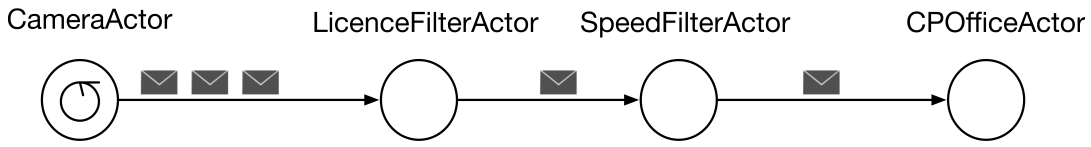
Description

A traffic enforcement camera[1], is a camera that is usually mounted beside or over a road to detect traffic violations. These cameras can take photos and measure the speed of a vehicle. Before a photo message (e.g., speed infraction) is sent to a central processing office (e.g., in Brussels), a number of *filters* need to be applied. For example:

- If no license plate is found in the photo, the system is unable to process the message any further, and it will be discarded.
- If the speed is below the maximum legal speed (e.g., 70 km/h), it will also be discarded.

In other words, only messages that contain the *license plate* or a *speeding vehicle* end up getting to the central processor.

For this assignment, you need to implement an *Actor System* that simulate the processing flow explained above. Your solution should enable use any filter in any particular order. In other words, the result of applying the *license plate* filter and later the *speed filter*, or viceversa, must be same. If the photo message is indeed valid, the *last* filter in the processing flow has to send it to the *central processing office*.



For your solution consider the speed camera, license plate filter, speed filter, and the central processing office as an *Actor*. The *speed camera* actor is to emit every 30 seconds

[1] `https://en.wikipedia.org/wiki/Traffic_enforcement_camera`

a new photo message. The photo message contains only two attributes *license* (String) and *speed* (Int).

Deliverables A short *report* (in English) explaining your solution to the problem. The report *file* should follow the naming schema `firstname_lastname_SA4.pdf`, and it should be handed in as a PDF file. For example: `Humberto_Rodriguez_SA4.pdf`.

Submit the *report* and the *source code* of your solution as a single *ZIP* file on the Software Architectures course page[2] in PointCarré, by clicking on *Assignments (Opdrachten) > Assignment 4*.

Grading The assignment will be graded and can become subject of an additional defense.

Notes

- Team work is not allowed.
- The solution to this assignment can be implemented in *Scala* or in *Java*.
- Copying – from other students, or from the internet – will be considered as plagiarism and be reported to the faculty.
- If you use any other resources besides those provided in the lectures and in this document, remember to cite them in your report.

[2] Use the English variant "Software Architecture**s**", rather than the Dutch one "Software Architecture**n**".