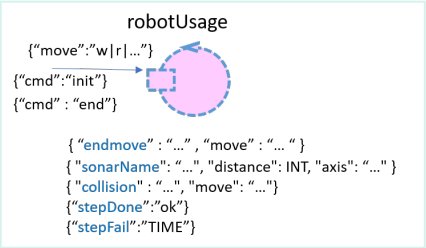
LabIss2021 | FirstActor : using KotlinActors to control a (virtual)robot

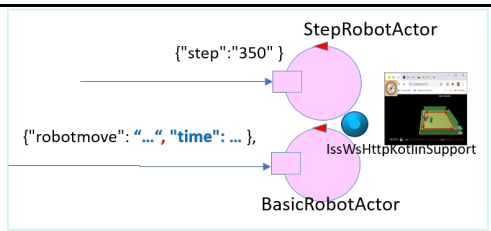
Divide et impera?

As usully happens in oop, the Single Responsibility Principle could induce the software designer to distribute different behaviours in different actors.
An accurate analysis is required to consider the effects of different architectural configurations.
Let us consider here what could happen if we want to introduce a StepRobotActor, as discussed in the project phase of cautiousExplorerActors.html.

> The main feature of a StepRobotActor is related to its capability of executing a **step** in 'atomic' way (i.e. all or nothing).
> A step is done when distance traveled by the robot is equal to the diameter of the circle in which it is supposed to be inscribed.

Step+Basic: case 0

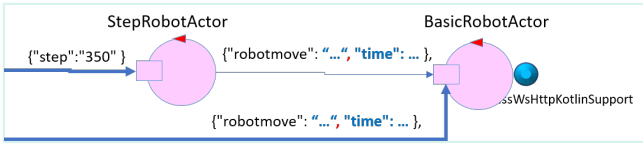




- **pros:** both the actors directly communicate with the support without any delay due to intermediate communications; every actor is dedicated to an own precise task
- **cons:** adding new features may need a large number of actor
- **caveat:** because of more that one actor directly uses the same support, beware of interferences

Step+Basic: case 1

- **pros:** no interferences because the support is only used by ***BasicRobotActor*** and, in addition to this, the managment of the support is simpler than the previous case
- **cons:** in many cases the communication pass between two actors (there is one level of indirection for ***StepRobotActor***);

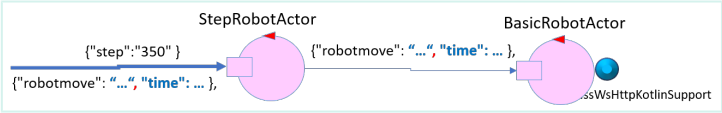


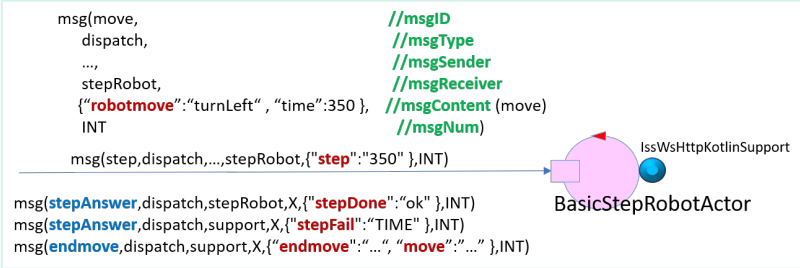
adding new features could make ***BasicRobotActor*** a bottleneck

- **caveat:**

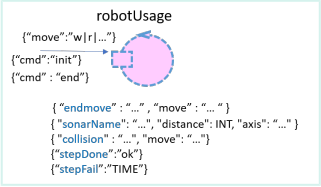
Step+Basic: case 2

- **pros:** like the previous case, there is no interference; it is possible to interface only with an actor for both functionality (encapsulation)
- **cons:** the communication always pass between two actor (***BasicRobotActor*** is only an intermediate and, in addition to this, the *robotmove commands* are only redirected by ***BasicRobotActor***); the principle of **single responsability** is not applied
- **caveat:** ***StepRobotActor*** must manage both functionality in the correct way





Step+Basic: case 3



- **pros:** there is one less actor then the previous cases so the performance is better; communication is direcly with support (no levels of indirection); the new structure of the message lets the possibility to make a logging and tracing system
- **cons:** the overall structure of the code is a little more complicated (particularly for the unique actor)
- **caveat:**appropriately manage the creation of messages

By student
Name: Luca Marchegiani
Email: luca.marchegiani3@studio.unibo.it
Git Repo: https://github.com/LM-96/MarchegianiLuca