Assignment 2: AspectJ and Code Generation

This piece of coursework will contribute with **20%** to the module mark. This is an **individual** assignment. Do not work together to avoid plagiarism concerns.

Return your answers through Blackboard by **Friday 5th of April (extended by a week)**, 23:59 UK time.

Marking will be done anonymous (your identity will only be revealed to the marker after all marks have been assigned). Please do not include your name in your submission.

Submit all answers in the filename indicated in each task, as a zipped file containing the AspectJ project. Code that does not compile cannot be marked. Please read the sections on late submission of coursework and plagiarism in the student handbook.

Assessed learning outcomes:

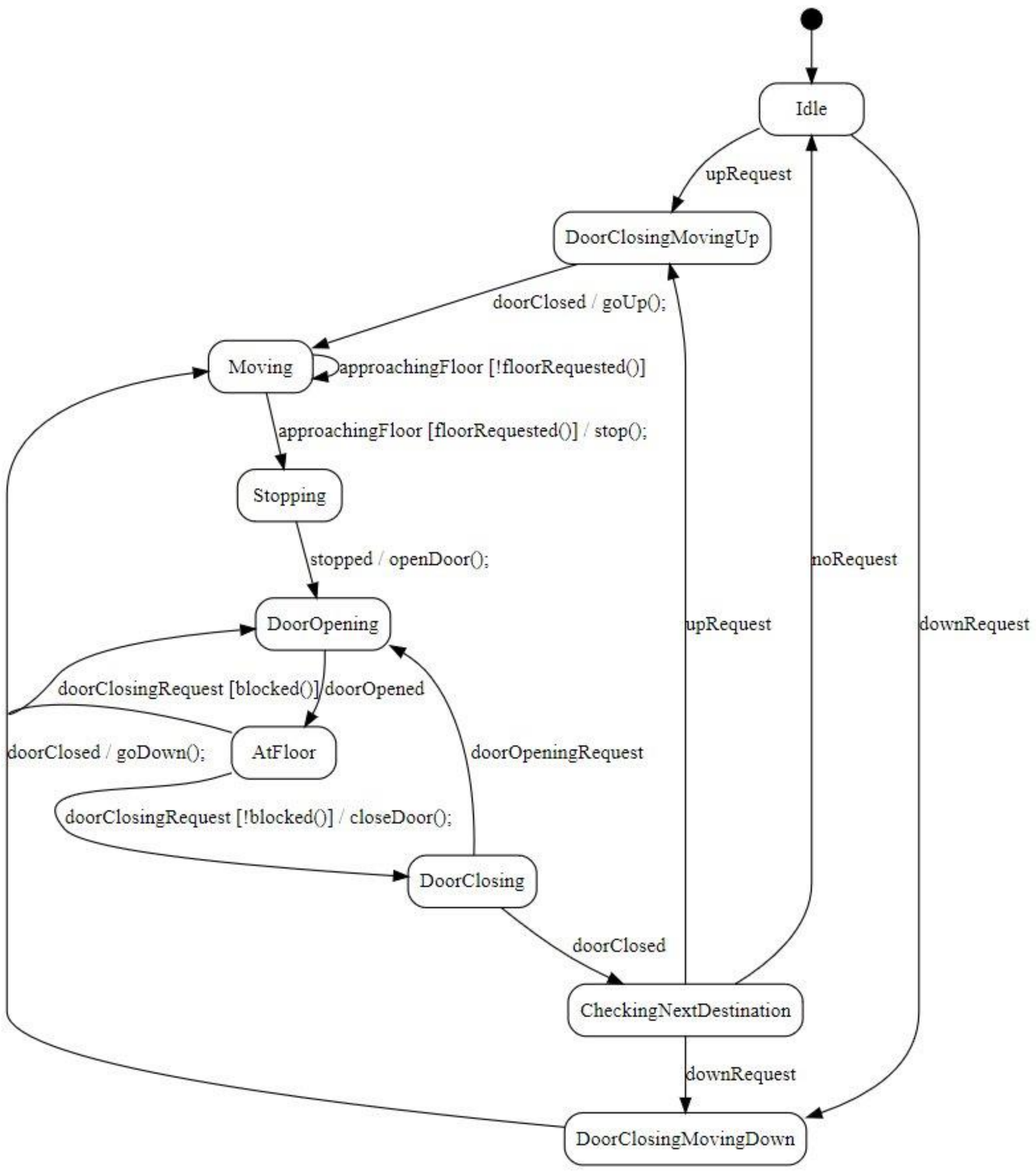
- Use techniques of generative software development.
- Critically evaluate the role of modelling and code generation in software development.

Introduction:

This project combines code generation from a state machine (UMPLE as presented in lecture 15 and tutorial 5) with AspectJ (presented in lectures 16-18, tutorial 6, and labs 6-7).

You can download the file elevator.zip if this assignment and import it as a project into Eclipse or STS as shown in quick explanation of the code: https://leicester.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=668a818a-f59c-47f6-972c-aa0c00fca866

You will work on an elevator control program. The code for the following state machine of the elevator is already generated into class Elevator:



A GUI of the elevator is implemented in class ElevatorVis.

An example of how to instantiate the elevator visualization and how to execute the state machine is provided in class ElevatorRunner. You can modify the execution steps here.

It is your task to develop aspects using AspectJ to improve the behavior of the generated code. You are **not allowed to modify any code other than the contents of ElevatorRunner.java and Task1.aj to Task4.aj**.

Task 1: Tracing [20 marks]

a) Whenever the state (attribute Elevator.state) of the state machine changes, print the previous (from) and next state (to). **[10 marks]**
As an example, the execution of class ElevatorRunner should show a trace starting like this:

```
From null to Idle
From Idle to DoorClosingMovingUp
From DoorClosingMovingUp to Moving
…
```

b) Whenever an event occurs but is not processed by the state machine, print the name of the event and print that it hasn't been processed. **[10 marks]**
Assume that all public methods of class Elevator that return a Boolean value are processing events. The return value is false if the event has not been processed.
As an example, the execution of class ElevatorRunner should show a trace starting like this:

```
From null to Idle
From Idle to DoorClosingMovingUp
From DoorClosingMovingUp to Moving
Event upRequest not processed.
…
```

Implement pointcuts and advice for both a) and b) in aspect Task1.aj. Ensure that this aspect will still work when the model changes, e.g., when adding or removing states or when adding or removing events the new functionality should apply.

Marking:

- Each correct pointcut [5p]: A pointcut that is too strict or too permissive will lose marks.
- Each correct advice [5p]: An advice that doesn't print the required information (from and to state / event name) will not receive marks.

Task 2: New transition from all states [15 marks]

Implement advice Task2.aj to add a transition from any state to state Idle when the event noRequest occurs.
Hint: The event noRequest occurs whenever the public method Elevator.noRequest() is executed.

Marking:

- Correct pointcut [5p] (for the specific event only)
- Correct advice [10p]

Task 3: Transition from all states [50 marks]

Implement advice Task3.aj to keep track of the current floor that an elevator is on. Make sure your advice will work when there are multiple elevators, i.e., do not store the current floor inside the advice.

a) Add appropriate fields for tracking the current floor and whether the elevator is going up or down. **[10 marks]**

Marking:

- Correct addition of field for tracking floor [5p]
- Correct addition of field for tracking direction [5p]

b) Whenever the actions goUp() or goDown() are executed, store the direction information in the field created in a). **[10 marks]**

Marking:

- Correct pointcuts [3p] each
- Correct advice [2p] each

c) Whenever the event approachingFloor is successfully processed (i.e., the method Elevator.approachingFloor() returns true), increment or decrement the current floor based on the direction information from b). **[10 marks]**

Marking:

- Correct pointcut [4p]
- Correct advice [6p]

d) If the elevator is on floor 0 it should not process the event downRequest, i.e., the execution of method Elevator.downRequest() should not proceed and should return false. **[10 marks]**

Marking:

- Correct pointcut [3p]
- Correct advice [7p]

e) If the elevator is on floor 20 it should not process the event upRequest, i.e., the execution of method Elevator.upRequest() should not proceed and should return false. **[10marks]**

Marking:

- Correct pointcut [3p]
- Correct advice [7p]

Task 4: Improve visualization [15 marks]

Implement aspect Task4.aj to show an image a blocked elevator door (by calling method ElevatorVis.show(“open_blocked”)) whenever the state machine is in state AtFloor, the event doorClosingRequest occurs, and the elevator is blocked.

Marking:

- Correct pointcut [5p] (for the specific event only)
- Correct advice [10p]

Submission:

You need to submit via Blackboard. Once you select this assignment from "Assessment and Feedback" you will be taken to a different screen that allows you to submit your solution.

Your submission should include:

- a zip file containing the complete, exported Eclipse project with aspects Task1 to Task4 implemented according to the specifications above.

See this video on how to export projects: https://leicester.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=d3c3c8ce-2882-42d5-a5dd-aa0300d44c9e

Once the file is uploaded and you are happy to proceed with the submission, click the Submit button (Save will not transfer your submission, but allow you to store work at intermediate stages). You can re-submit your assignment as many times before the deadline as you like, but only the last submission will be marked.

If you do not follow the submission guidelines your work will not be marked.