Engineering of Domain Specific Languages (Fall 2025)

Dr. Andrea Mocci
Assignment 01: Domains & Semantic Models (100 pts + 10 bonus points)
Deadline (Hard): October 16, 2025 at 16:59
Version: v1.1, Released on October 2, 2025 at 12:00

Assignment Instructions

Please read carefully this section.

Assignment Group. This is a **paired** assignment. Please indicate your group in the Sharepoint Excel file that you will find on iCorsi. Please contact the instructor **ASAP if you are alone because there are no remaining students for a pair.**

Grading. Grade is based on the following criteria:

- Submitting this assignment satisfying the **basic requirements** (listed at the end of the assignment) will normally **guarantee a base grade of 60%**. Any violation of the basic requirements will imply a penalty (your base grade will be below 60%).
- If you are fine with your base grade, you can opt out the oral discussion, and you will get that grade. You can opt out as a pair, or as a student alone.
- Extra 30% grade (thus, up to 90%) is obtained through discussion. You must answer 4 among the possible questions listed in the **advanced requirement** section, as chosen by the instructor. These questions will not involve in any form the bonus exercise(s). The quality of the submission on that specific aspect and your answer during the discussion will determine the grade. The discussion must be done in pair and each student will answer 2 out of 4 questions. The other student may complement the answer. The grade will be shared if no student will opt out. In exceptional and motivated cases, the students may discuss the assignment separately and get separate grades.
- If you want a grade over 90%, or if you are unsatisfied of the grade you obtained up to now and want to improve it, you will need to answer 2 additional arbitrary questions done by the instructor on your submission, including the bonus points. Any of the two students can opt-out on this additional discussion.

Dates and times for the discussions will be partially flexible and notified soon.

Plagiarism. Any evidence of plagiarism will be contested through mandatory discussion on arbitrary aspects of the assignment. The submission will be potentially void or considered as negative grade. Do not share solutions with other groups of students.

Policy for the use of LLMs. You are allowed to use AI tools based on Large Language Models (LLMs) like CoPilot or ChatGPT for this assignment, but you must ensure the following aspects:

- You must clearly indicate in the submission that you have used an LLM, which one, and how. You may share the interactions done with it.
- You acknowledge full ownership of the solution you provide. The solution needs to use only constructs and syntax forms of Scala that were explained during the lectures. Anything else should be properly justified and motivated, and this motivation could be invalid or contested: Any unnecessary construct will be considered invalid. In particular, in this Scala feature you should not use any advanced programming language feature that has not been explained during the lecture. If you are not sure, please ask upfront.
- In any case, undeclared or unjustified use of AI assistants (using constructs not explained during the lecture) will be considered plagiarism (see above).

Slack Day. Each of you has two slack days. You should communicate your intention to use the slack days **before** the assignment deadline via email. Since the assignment is paired, a request for a slack day will consume a day for both students.

Submission Instructions

Please use the template in iCorsi/Gitlab. Code should compile without errors, *i.e.*, `sbt compile` should run successfully.

Use comments / documentation whenever needed to clarify your choices.

For other exercises, submit a pdf, a markdown, or a text file, with images when needed. Please organize folders in a reasonable way, and submit a zip/tar file. In the rare occurrence that your submission exceeds the size of 20 MB, you have definitely included something unnecessary. *Do not include executable or compiled files (*e.g., *.class files):* you should run `sbt clean` before zipping and submitting the assignment.

Submit your solution via iCorsi as a zip file. You can submit multiple times, but only the last submission will be considered. Late submissions will not be considered.

1 Domain Modeling: Flight Bookings (40 points)

Please submit a diagram in the form of a PDF file for this exercise. Use your favorite software tool that supports shapes to build diagrams (*e.g.,* OmniGraffle if you have it, Apple Pages, Microsoft Word), online service (*e.g.,* draw.io), or you can also manually sketch it with an analog medium (*e.g.,* paper) and scan it (but please, let it be clear and readable). You can also optionally use the special DSL included in the template.

Note: If you think there is any ambiguity, or some alternatives to be preferred when modeling the domain above, illustrate your thinking and always justify your specific choice. For any doubt, the instructor is available for clarifications.

1.1 Domain Modeling (40 points)

Model a domain to support the description of a flight booking, *i.e.,* a reservation for some flights.

A booking has a code, composed of 6 uppercase letters and digits. It is composed of a set of trips, *e.g.,* , an outbound trip and an inbound (return) trip. Each trip is composed by a set of segments, *i.e.,* a flight between two airports.

A flight is identified by a flight number, composed of a two-letter IATA code that identifies the airline that markets it (*e.g.,* LX), and 4-digit flight number (*e.g.,* 0123).

A flight may be operated by a different airline that sells it: there are pure flights or a codeshare flights. A pure flight is one where the operating airline is the same of the marketing airline. In a codeshare flight, the flight is operated by another airline, with a different 4-digit flight number.

An airport is identified by a three-letter code, and it is located in a city (which is identified by its name). For example, LIN and MXP are two different airports in Milan, and ZRH identifies the airport in Zurich.

A booking involves a set of passengers, identified by their first name and last name. A passenger may have a frequent flyer number.

Examples

- **Example 1.** Passenger Alex Geek, with frequent flyer number OX89156273, has a booking with two trips, from Sydney - SYD to Paris - CDG and back (CDG to SYD). The first trip is on December 27 2025, and has only one segment, from SYD to CDG, marketed and operated by Oceanic Airlines (IATA code OX) with flight number 0815.

  On the way back, on January 15th 2026, the trip has two segments, from SYD to SIN (Singapore) and from SIN to CDG. The first segment is marketed and operated by Oceanic Airlines with flight number 0700, and the second segment is marketed by Oceanic Airlines with flight number 0403, but operated by Pan Am Airlines with flight number 0100.

2 Semantic Model (40 points)

Translate the domain data model from the previous exercise into an adequate semantic model in Scala (using case classes, traits). Use require clauses only for simple constraints that are not too complex to represent. Use the template provided on iCorsi / Gitlab, and reuse the code that creates URLs. To model dates, please use the JDK classes specified in the template. Represent Example 1 by just invoking the constructors.

3 Fluent API (20 points)

Sketch a fluent API with the same techniques that we have seen during the Lecture 3, using nested functions, method chaining, and function sequences. Represent the same example as in Exercise 1, plus another example that you describe in a comment as Example 1. Do not implement the methods but leave them unimplemented like in the code for Lecture 3.

Bonus Exercises (10 points)

Extend the domain model, the semantic model, and the fluent API to include the following additional features. You do not need to replicate any part of the previous exercises, just put comments that clearly indicate which parts involve the bonus exercise.

A passenger can be, depending on their age in years, child, young, adult, or senior.

For each segment and each passenger in the reservation, there is a seat assignment in a given cabin of service. A seat is defined by a letter and a 2-digit number, and the cabin of service can be *economy, premium economy, business, or first*. In any segment, the passenger can potentially order a special meal. Do your own research on the kind of special meals that are typically available with airlines.

Basic Requirements for Assignment 01

- You have attempted all exercises (penalty proportional to exercise weight).
- The notation used for domain data models is consistent with what has been shown at the lecture (Penalty: 10%).
- There are not major bad smells on domain data models (primitive types over concepts, unjustified duplicate relations, boolean flags) (Each bad smell: 5%).
- The code compiles (Penalty: up to 20%).
- The code contains only programming language constructs and features as explained at Lecture 03. Anything else is justified properly (Penalty: up to 20%).
- All classes and traits in the code are consistent with the domain data model (Penalty for each inconsistency: 5-10%).
- The fluent API has the characteristics of a fluent API and uses the mechanisms explained during the lecture (Penalty: 5-10%).

Questions for Advanced Requirements for Assignment 01

Whenever the questions mention a *specific* element (e.g., concept, relation, class, trait), the instructor will point out to a specific instance of that element in your assignment.

Questions on domain data models

- Why did you choose to represent this specific concept as a set?
- Why did you choose to represent this specific relation?
- Why you did this specific generalization as such? (i.e. why it involves these specific concepts)
- Why a specific set of relations has been modeled as such (i.e. their name, their domain, and their range)?
- What is the reason of some specific (or potentially missing) multiplicities of these relations?
- Why you named this specific concept / relation as such?
- Can you explain if a specific part of the model has a specific bad smell?

Questions on semantic models

- Why a specific case class has this name and these fields?
- What are the elements in the domain data model that correspond to these specific entities in the code?
- Why did you use this require clause?
- Why there is no require clause in this case class?
- Why did you use a trait / case class to model this entity?
- Why this trait is (not) sealed?
- Why this field has this type?
- Which relation is modeled by this field?
- Why did you represent this specific part of the example as such?

Questions on fluent APIs

- Is this fragment of code fluent, and in which sense?
- Would a particular mechanism for fluent APIs (function sequence, method chaining, nested functions) work better in this context instead of the one you used?
- Why did you represent this specific part of the example as such in the fluent API?