CS 2ME3 – Assignment 1

Available from: 25 September 2023
Deadline: 21 October 2023

Basic expectations

- Teams of 2 or 3. Teams are formed by 29 September, latest. Report your teams to your TA.
- Use proper OO principles in models.
- (Some) code must be generated. (Document everything.)
- Code must compile.
- Documentation must be concise.
- No freeriding. Also, standard academic dishonesty rules apply and will be rigorously enforced.
- Must meet 50% overall, and must have more than 0% in each task.
- Oral presentation is mandatory for everyone. Again, academic dishonesty rules will be rigorously enforced.

Failure to meet any of the above points will result in a failed assignment without a chance of re-submission.

Scenario

A startup company plans to launch a service reservation application. Their current services include car rentals and vacation travel packages. The company has hired you to oversee the new service reservation application.

Clients typically book either a car or a vacation, but sometimes, they book both to be able to drive around their vacation spot. The platform stores the following information about cars: license plate, year, make, number of doors; and the following information about vacations: country, city, season. Bookings have a start and end date and a booking ID generated upon creating the booking in the system.
The platform stores the following information about clients: client ID, name, birth date, and contact information. Clients also have memberships that can be regular, silver, or gold. Each subscription has a different discount % associated with it that reduces the costs of reservations. Regular: 0%, Silver: 10%, Gold: 20%.

Tasks

Task		%
Task 1	Modeling and code generation	40
Task 2	Implementation	40
Task 3	Evolution	10
Task 4	Documentation	10

Task 1. Modeling structure and generating code (40%)

Task 1a. Modeling (30%)

Model your software using a UML Class Diagram. Use proper OO mechanisms, such as abstraction, inheritance, references, aggregations, etc.

Task 1b. Code generation (10%)

Generate Java code from the model. Refine your models until you get sufficiently detailed class skeletons you can work with in later tasks.

Task 2. Implementation (40%)

The company asks you to develop a console-based REPL program in Java. The program prints a menu on the console and waits for user input. The following actions must be supported:

- Create a new item
  - Car (license plate, year, make, number of doors)
  - Vacation (country, city, season)
- List items
- Book a new item (in: from date, to date, user ID; out: generates a booking ID)
- Cancel a booking (by referring to a booking ID)
- List bookings
- List bookings by user in the following format:
  - | BookingID | Items in the booking | From-to | Price
- Show revenue of the business
  - For simplicity, revenue is defined as the sum of money clients pay for bookings

Implement the application using the generated skeletons (Task 1). Try to avoid altering generated code as much as possible, although this is not a must.

The entry point of the program should be the `public static void main(String[] args)` method of the `Program.java` class.
Tip: Use a while(true) loop to keep the application running in an infinite loop that prints the menu, waits for an action, and executes it.

Task 3. Evolution (10%)

The company wants to add motorcycles (bikes) to its portfolio and make them available for booking. Bikes have the following attributes: license plate, make, year.
Modify your models, re-generate your Java skeletons (Task 1), and modify the implementation (Task 2) accordingly.

Task 4. Documentation (10%)

Document activities and main findings.
Every team member's contributions should be briefly described in a table.
1 cover page + max 5 pages of content.

One section should be about reflections. What are some of the lessons learned? Some ideas:

- Did modeling help understand the domain better? (Task 1a)
- Did the modeling tool generate good enough code you could use or did you have to touch it up? (Task 1b)
- Did the generated code enforce some of the OO principles? Did this help in organizing your implementation better? (Task 2)
- Did modeling help manage evolution or was it more of a burden? (Task 3)