Important rules and remarks:

- You are allowed to use any source of information; **communication with third parties is forbidden**
- You must be able to explain your code and you must be able to perform modifications if asked; otherwise the exam is not passed
- If there is anything unclear, **ask** us
- You must use the technologies from the lab/lecture
- There are NO PARTIAL points, only the ones specified in the problem statement
- **The requirements MUST be solved in the order they appear in the problem statement; the points for any feature are given only if all previous features are completely correct**
- **You must accept the exam assignment (**available on the group/channel**) and work on the generated github repository**
- **Each feature (e.g. F1, F2 etc) will be developed on a different feature branch whose name contains the feature name; e.g: 'feature/F1', 'feature/F2' etc**
- **After solving each feature, the feature branch will be merged into the main/master branch;** push your changes to github as often as possible
- **After solving each feature, mark that feature as solved (with an 'x') in the attendance file** (the attendance file will be shared on the group/channel)
- **After solving each feature, notify the teachers for evaluation; *you must present each feature before proceeding to the next one***

Movie Rental

The following csv files are given (available on the group/channel). Use the provided files only without changing them.

- clients.csv (id, firstName, lastName, ssn); ssn is of type string and it is unique; id uniquely identifies the client, but it does not have to become the entity identifier
- addresses.csv (id, county, city, street, ssn); both id and ssn denote the related client; the relation between client and address is one-to-one, the addresses being kept is a different table, i.e, not in the client table
- movies.csv (id, title, description, genre); genre is an Enum with the values "Adventure" and "Thriller"

The following domain entities should be created based on the csv files: Client, Movie, Address, Rental. All of them will extend the BaseEntity from the lab and lecture examples (the id being annotated with "@GeneratedValue"). The Rental (client, movie, rentedDate, dueDate) also extends BaseEntity. A client can rent the same movie several times. The client and movie attributes from Rental are references to Client and Movie.

At the repository level, ClientRepository and MovieRepository extend
BaseRepository<T extends BaseEntity<ID>, ID extends Serializable> extends JpaRepository<T, ID>
There will be no other interfaces or classes at the repository level.

Application logic will be implemented at the service level. A REST API will expose certain endpoints for an Angular front-end, as specified in the requirements form below.
At the address http://localhost:4200/ a navigation bar will be displayed at the top of the page and the following links will be provided: “Movies”, “Data Import”, “Clients”.

Movies screen
[**F1 20p**] When the “Movies” link is followed, at the address http://localhost:4200/movies , an ordered list (HTML element, not sorting) of movies is displayed (title, description, genre).
[F2 10p] The data will be paginated on the backend.

Data Import screen
[**F3 10p**] When the “Data Import” link is followed, at the address http://localhost:4200/data-import , the following elements are shown:

- A label with the text “Choose files (addresses, clients, movies):”
- An input that will allow to browse the file system for selecting several files (addresses.csv, clients.csv, movies.csv)
- An “Import” button
- A “Delete all” button

[**F4 20p**] The “Import” button will upload the provided files and will import all data into the application (saved in the db) by accessing an endpoint exposed by the rest controllers: http://localhost:8080/api/upload
[**F5 10p**] After the “Import” button is pressed, a “Data import started” message is immediately returned (from the backend) and displayed, i.e., the api call will not be blocked until the data is imported
[**F6 10p**] The Delete all” button will delete all the data from the db

Clients screen
[F7 10p] When the “Clients” link is followed, at the address http://localhost:4200/clients , the following elements are shown:

- A “County” label
- A “County” dropdown where the first element is the string“---”
- A “City” label
- A “City” dropdown where the first element is the string“---”
- An ordered list of clients (<firstName> — <lastName>)
- A “More…” button

[**F8 10p**] The list of clients uses pagination and displays the first 10 clients. The “More…” button brings the next 10 clients and adds them to the existing ones. Only the client data is fetched (without address).
[F9 10p] The “County” dropdown displays the list of counties from the db. The first element from the dropdown is initially the string “---”.
[**F10 10p**] When a county is selected, the “City” dropdown is populated (from the db) with the cities from the selected county. The first element from the dropdown is still the string “---”.
[**F11 20p**] When a county is selected, the list of clients is updated accordingly, i.e., only the clients from that county are shown, but if there were 30 items in the list before selection then

there should be still 30 items after selection (and these items should match the filter criteria). The “More…” button will continue to bring the next 10 clients according to the filter criteria and add them to the existing ones.
[F12 20p] When a city is selected, the list of clients is updated accordingly - matching both filters, as described in the previous feature. The “More…” button will continue to work as before - adding 10 more items to the list according to the filter criteria.

[**F13 10p**] On the same page, there will be a “Movie title” label, a “movieTitle” input and a “Search” button. The “Search” button will bring all the movies whose title contain the string provided in the “movieTitle” input and display them in a list.
[F14 20p] The user can select a movie and a client from the corresponding lists and using a “Rent” button a rental will be added into the db.
[**F15 10p**] The rentedDate will be the current date and the dueDate will be 30 days after the current date.