Petflix

Petflix is a Movie Store implemented with Java 8 & Java Remote Method Invocation (RMI).

There are two packages - client and server, which means client-side and server-side, accordingly.

All classes are well documented inside the code.

Server-Side

Server-side of Petflix Store contains following classes: Buy, ErrorCodes, History, Movie, PetflixServer, PetflixServerImpl, PetflixServlet.

ErrorCodes is an enum class contains mostly error codes and also success codes.

Movie, **Buy**, **History** are entity classes. **Movie** is a main entity containing information about movies. **Buy** is an operational entity containing the client and movie as parameters. **History** is also operational entity containing 2 string parameters: Client's name and the name of the movie.

PetflixServer is an interface that extends from **Remote** (RMI).

PetflixServerImpl is a class that implements core functionality of the server.

PetflixServlet runs the whole thing. I/O operations, connectivity.

Client-Side

Client-side of Petflix have some variety, containing such classes as ClientServlet, ConnectionLayer, FailureDetector, PetflixClient, PetflixClientImpl.

ConnectionLayer is a class that wrap the server objects and an instance of the **FailureDetector**.

FailureDetector is a class that responsible for failure detection and additionally measure performance.

PetflixClient is an interface that extends from **Remote** (RMI).

PetflixClientImpl is a remote class that has a name and a callback.

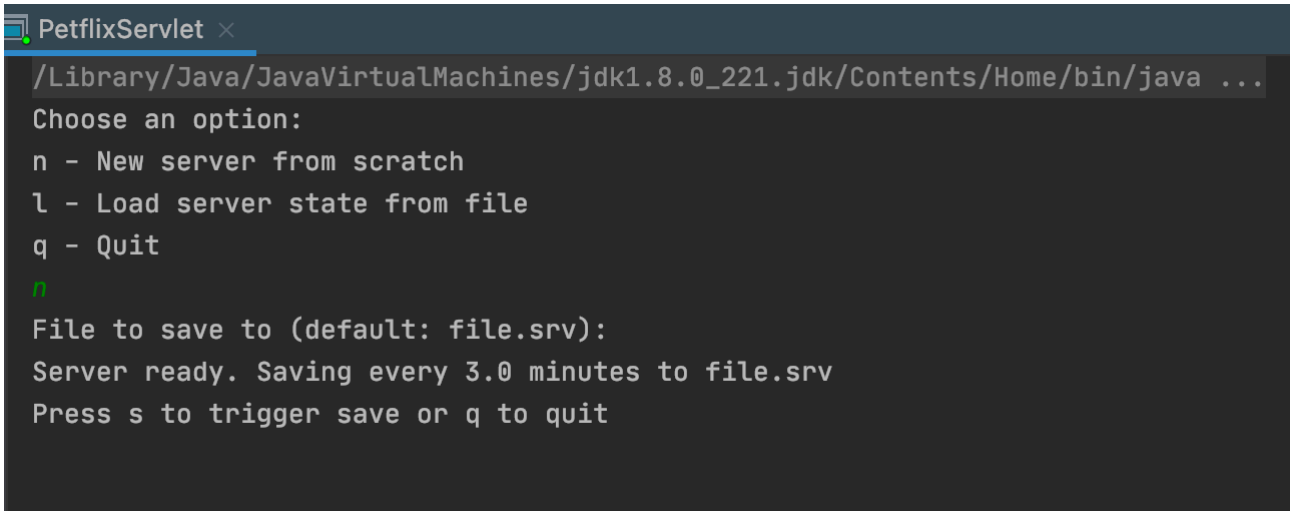
ClientServlet runs the whole thing of the client-side.

Testing

For this test I'll run 2 different users simultaneously and show how operations are work.

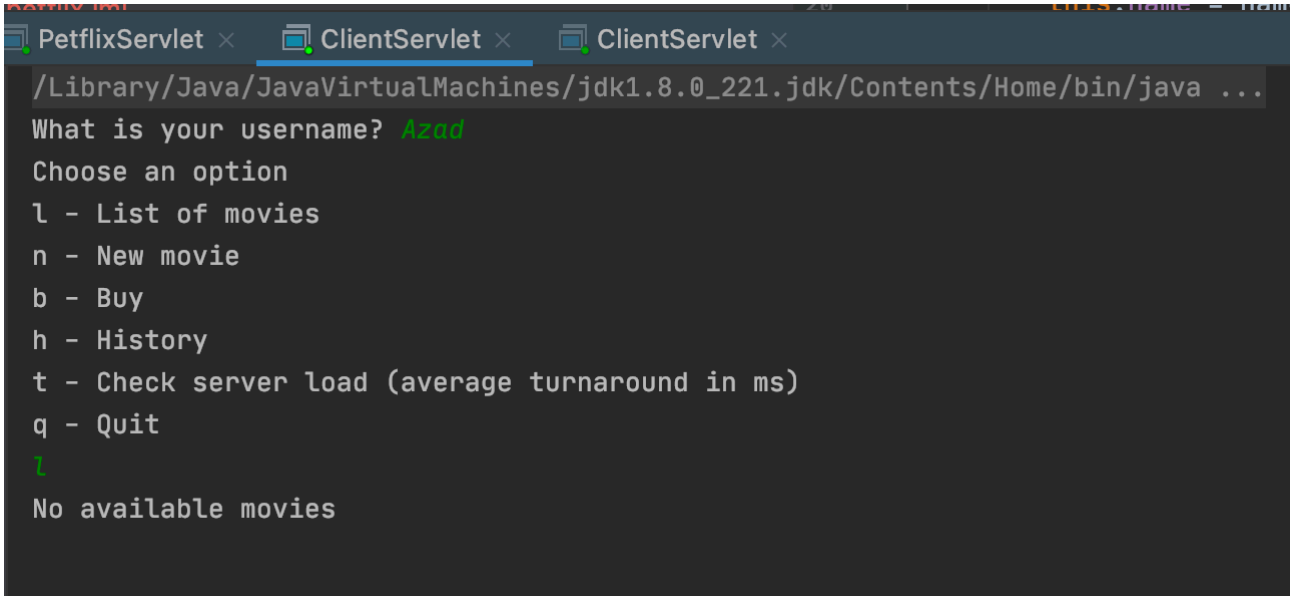
Step 1.

First of all, we need to run Server-side (PetflixServlet)



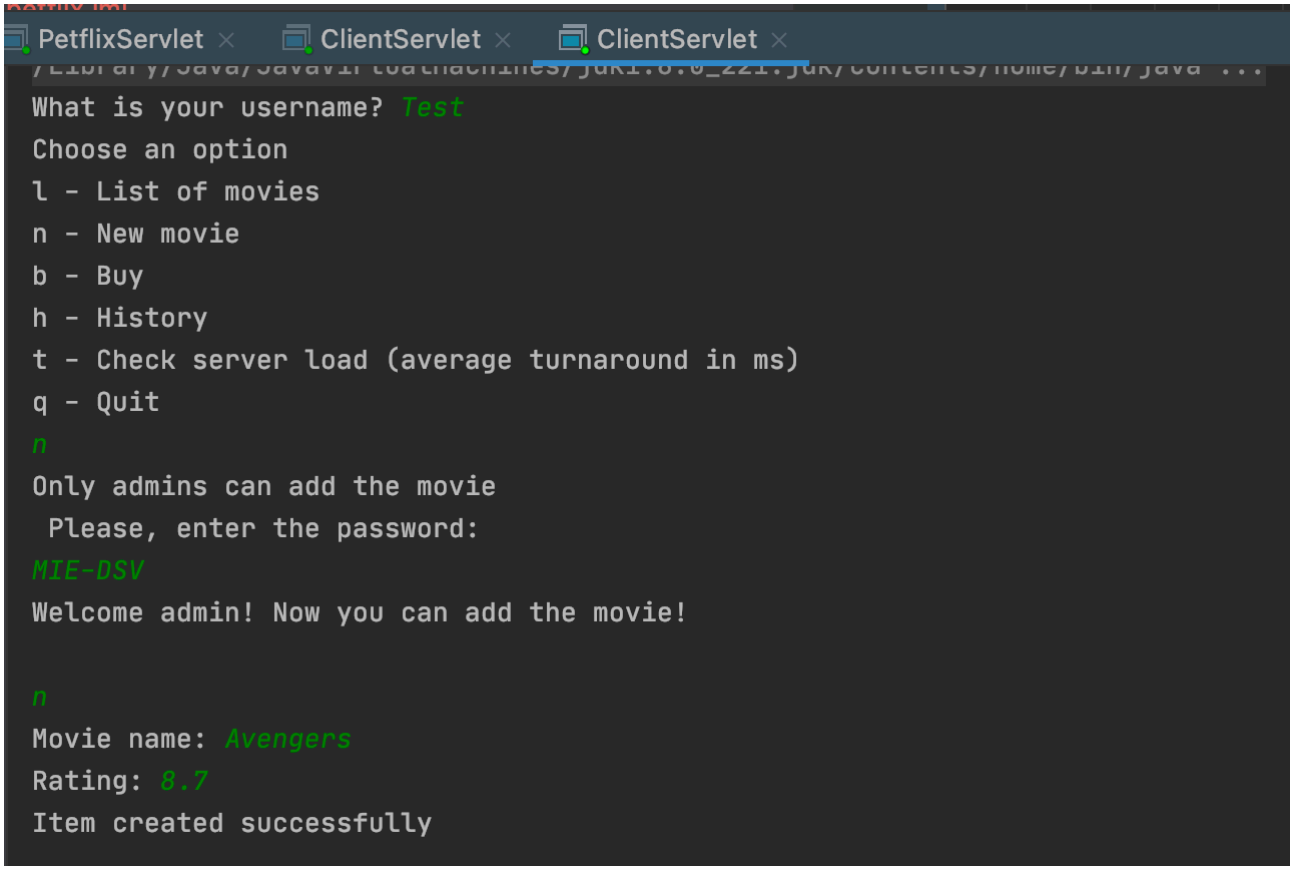
Here we created a new server and left the file name by default. After each 3 minutes server will save the file automatically.

Step 2.

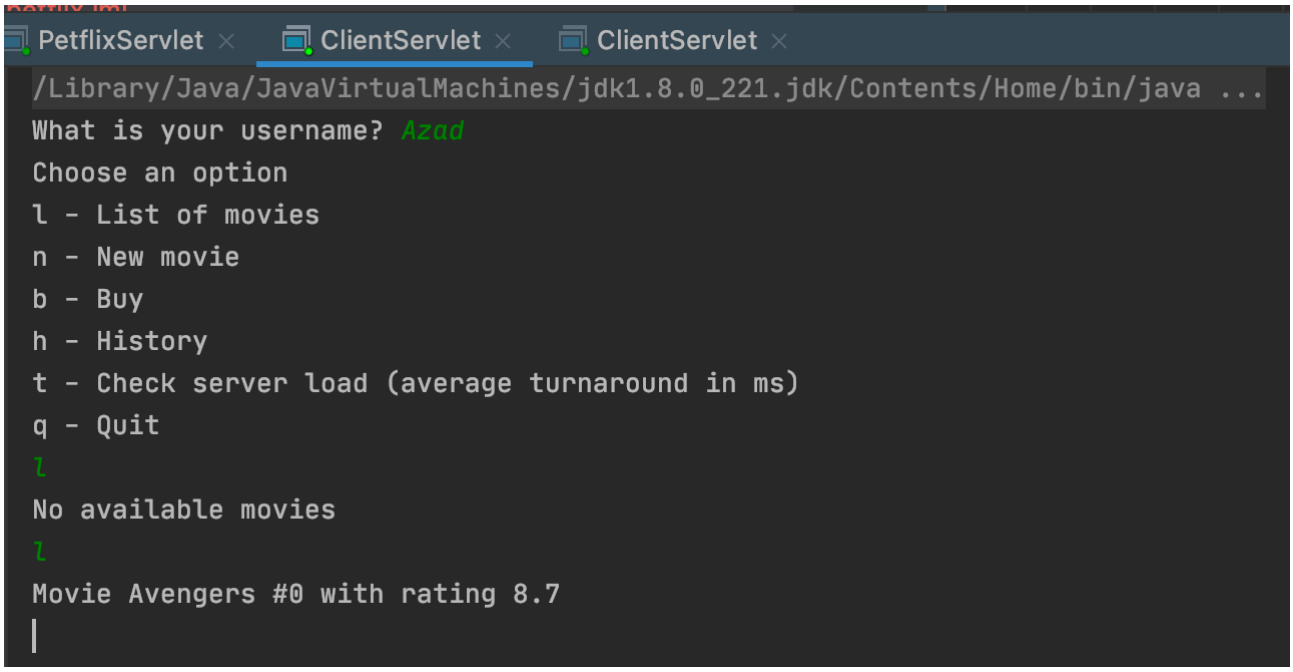


Now we will run 2 clients. One of them will be me - Azad and the second - Test. Let's look at the first start of these clients.

From the cold start client Azad cannot see any movies in the store.
Let's think that the client Test has rights to add the movie.



As we can see, the client Test (also as an Admin in this case) added the movie. And going back to client Azad we can see the result.



Step 3.

Now let's add some more movies and look at Buy and History operations.

```
l
No available movies
l
Movie Avengers #0 with rating 8.7
l
Movie Avengers #0 with rating 8.7
----------------------
Movie Thor #1 with rating 8.1
----------------------
Movie Iron-Man #2 with rating 7.8
b
Movie ID: 2
Item was bought successfully
h
User Azad bought movie Iron-Man
```

Here for buy operation we need to write the movie id (#ID) to buy a movie item. And using history operation we can see the result.

For client Test it looks like this:

```
b
Movie ID: 2
The movie does not exist
b
Movie ID: 1
Item was bought successfully
h
User Test bought movie Thor
----------------------
User Azad bought movie Iron-Man
```

He cannot find the movie with id 2 anymore, thereby bought the movie with id 1. Looking up a history he can observe who bought the movie with id 2 (Iron-Man).