**MICROSERVICES**

**MANDATORY HANDS-ON QUESTIONS**

**Question 1**

Creating Microservices for account and loan

In this hands on exercises, we will create two microservices for a bank. One

microservice for handing accounts and one for handling loans.

Each microservice will be a specific independent Spring RESTful Webservice

maven project having it's own pom.xml. The only difference is that, instead of

having both account and loan as a single application, it is split into two

different applications. These webservices will be a simple service without any

backend connectivity.

Follow steps below to implement the two microservices:

Account Microservice

 Create folder with employee id in D: drive

 Create folder named 'microservices' in the new folder created in

previous step. This folder will contain all the sample projects that we will

create for learning microservices.

 Open https://start.spring.io/ in browser

 Enter form field values as specified below:

o Group: com.cognizant

o Artifact: account

 Select the following modules

o Developer Tools > Spring Boot DevTools

o Web > Spring Web

 Click generate and download the zip file

 Extract 'account' folder from the zip and place this folder in the

'microservices' folder created earlier

 Open command prompt in account folder and build using mvn clean

package command

 Import this project in Eclipse and implement a controller method for

getting account details based on account number. Refer specification

below:

o Method: GET

o Endpoint: /accounts/{number}

o Sample Response. Just a dummy response without any backend

connectivity.

{ number: "00987987973432", type: "savings", balance: 234343 }

 Launch by running the application class and test the service in browser

Loan Microservice

 Follow similar steps specified for Account Microservice and implement a

service API to get loan account details

o Method: GET

o Endpoint: /loans/{number}

o Sample Response. Just a dummy response without any backend

connectivity.

{ number: "H00987987972342", type: "car", loan: 400000, emi: 3258, tenure:

18 }

 Launching this application by having account service already running

 This launch will fail with error that the bind address is already in use

 The reason is that each one of the service is launched with default port

number as 8080. Account service is already using this port and it is not

available for loan service.

 Include "server.port" property with value 8081 and try launching the

application

 Test the service with 8081 port

Now we have two microservices running on different ports.

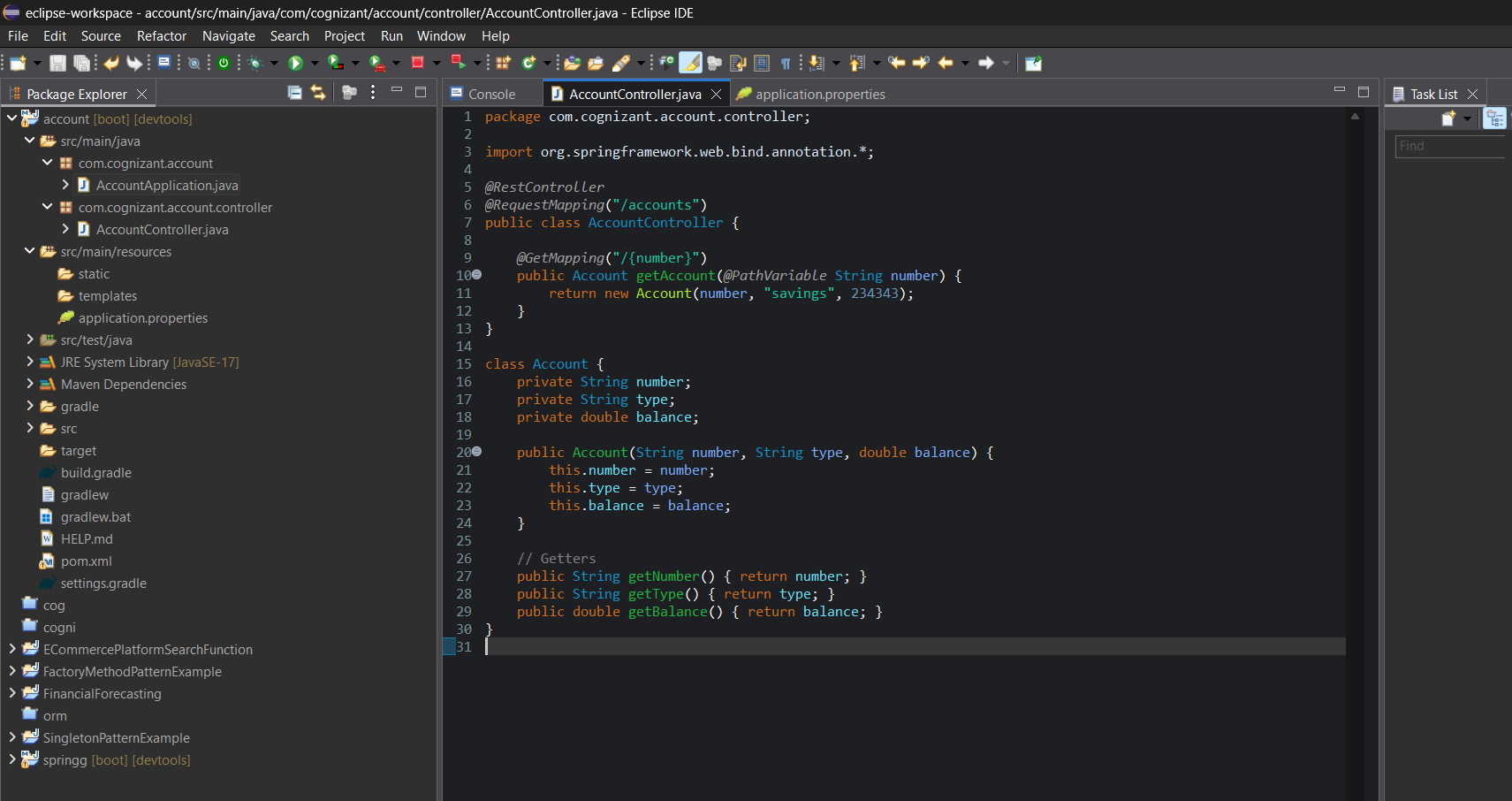
NOTE: The console window of Eclipse will have both the service console

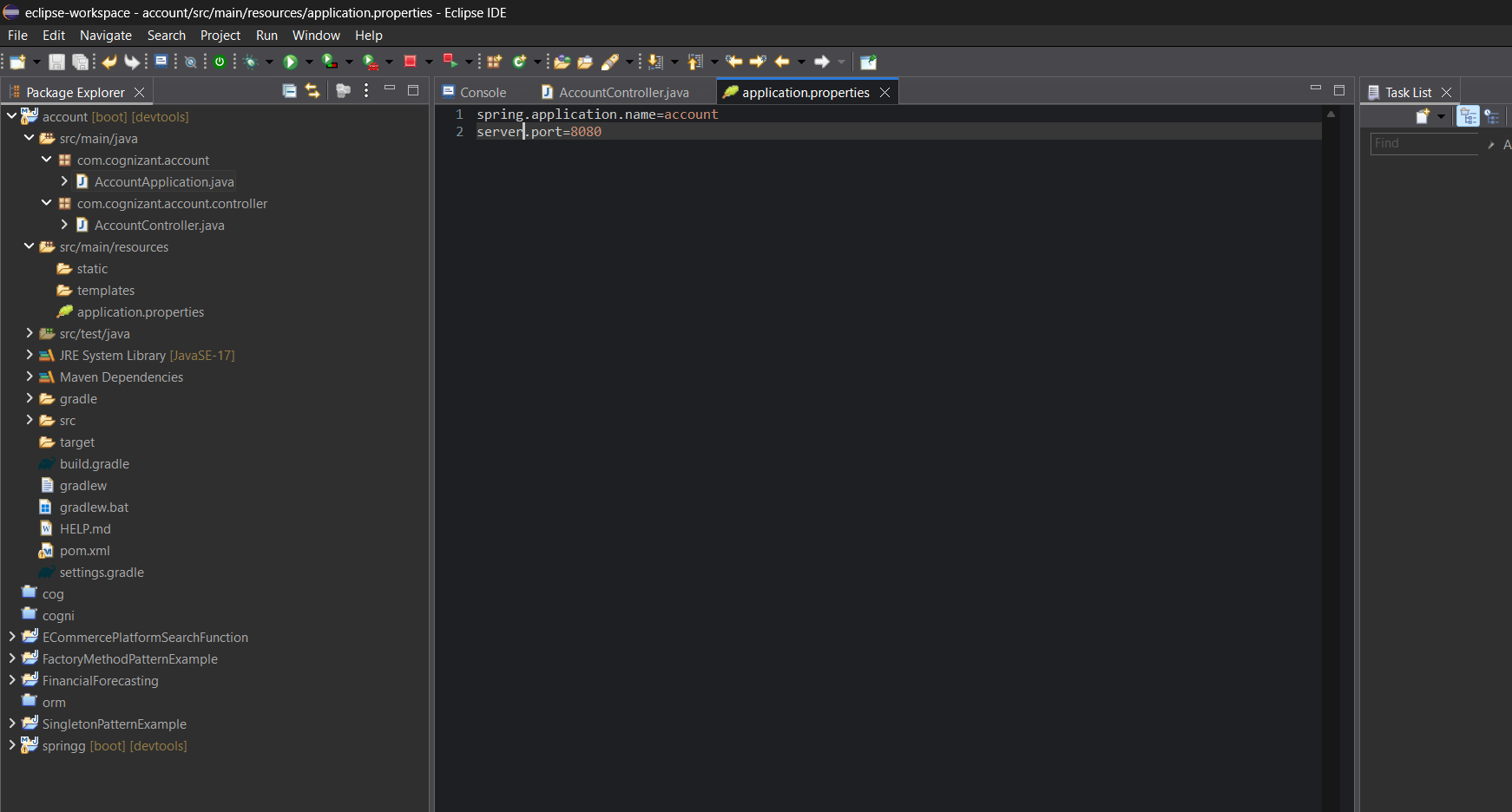
running. To switch between different consoles use the monitor icon within the

console view.

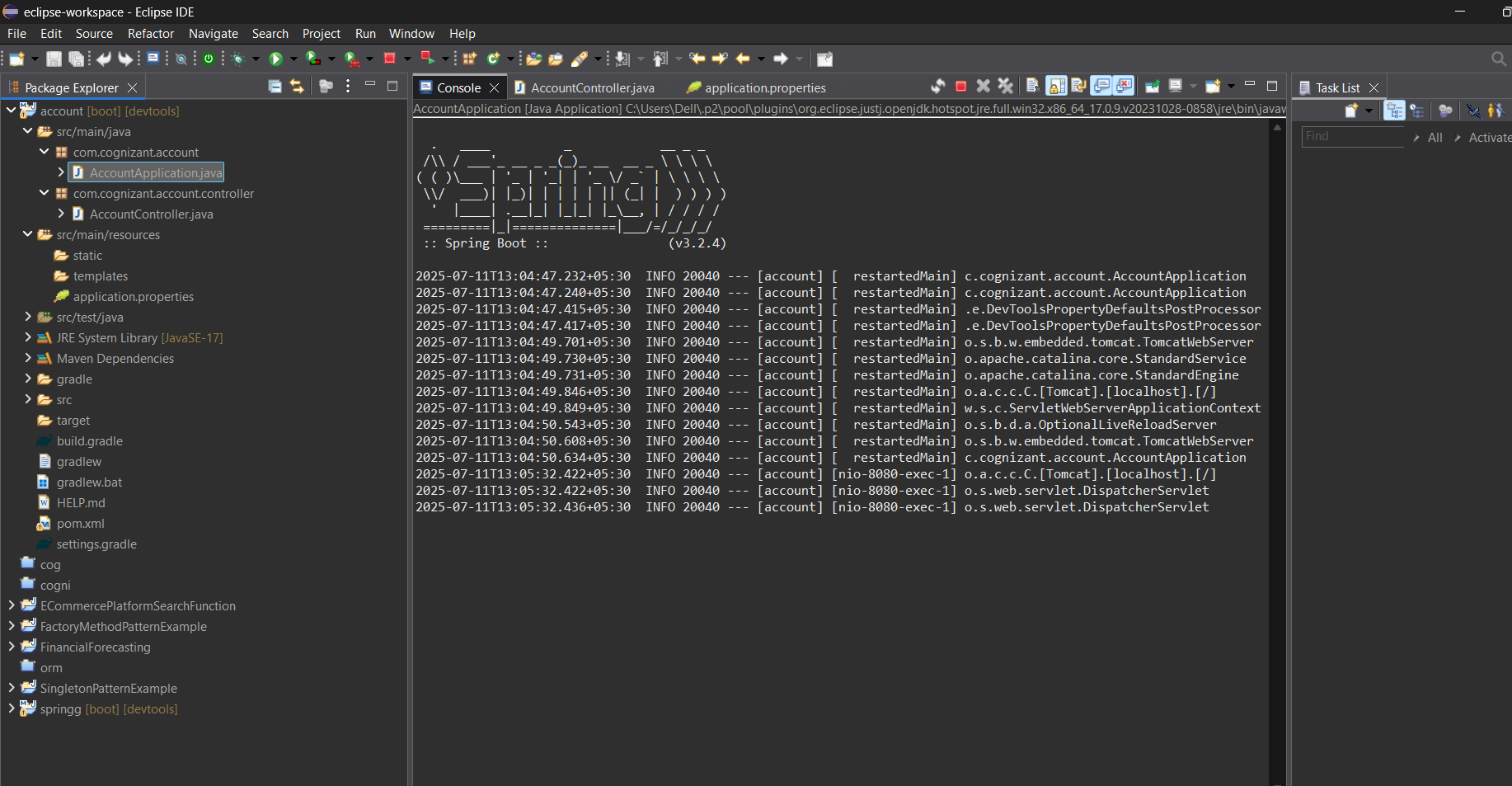
Answer and output:

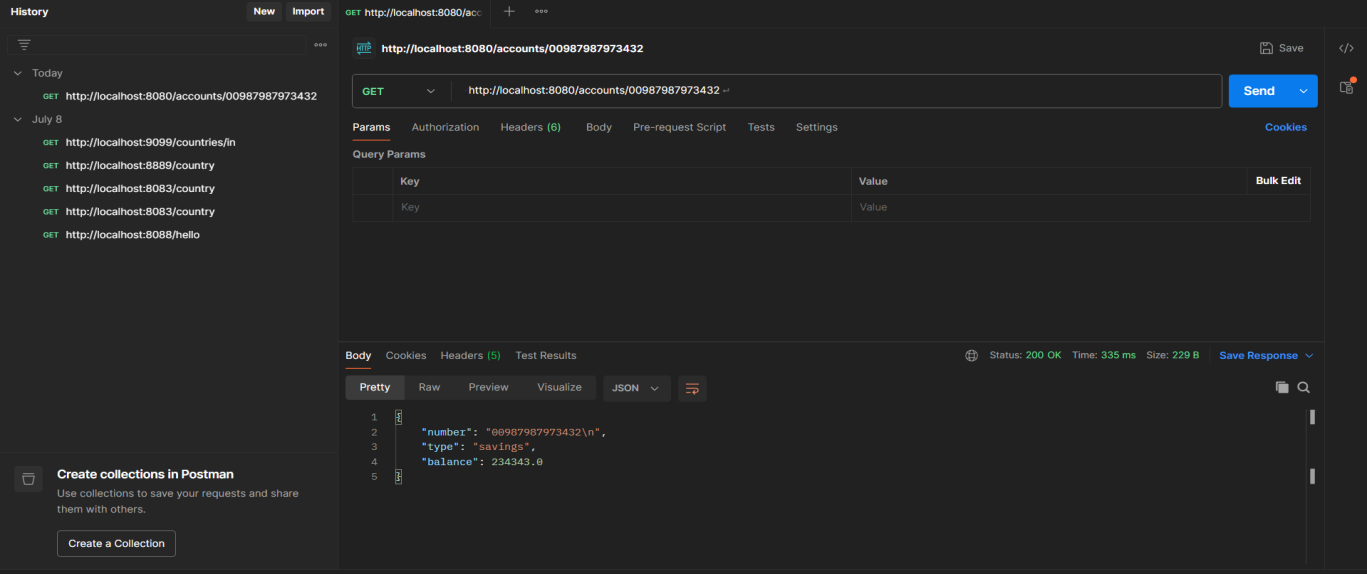
Account Service CODE:



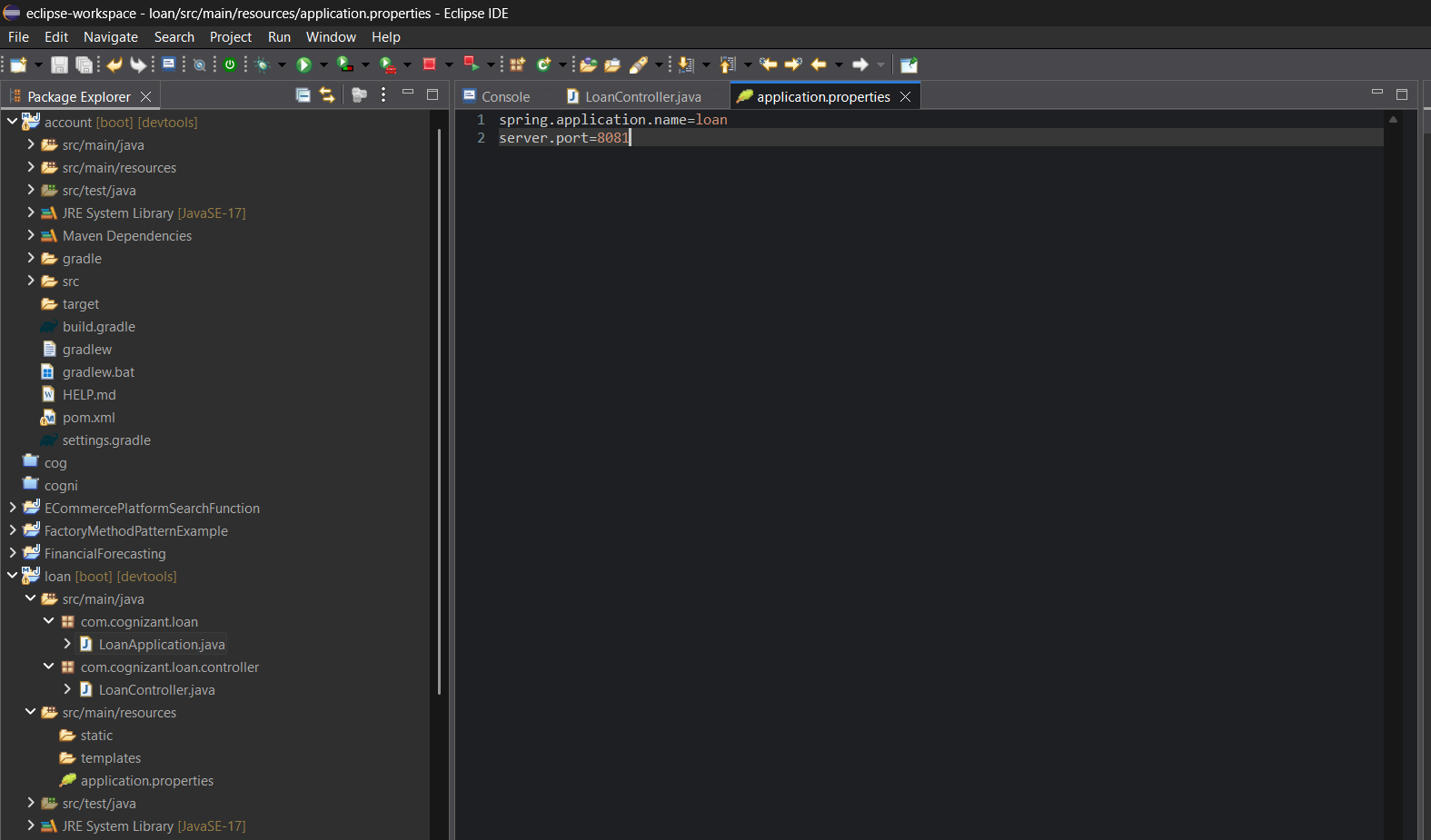


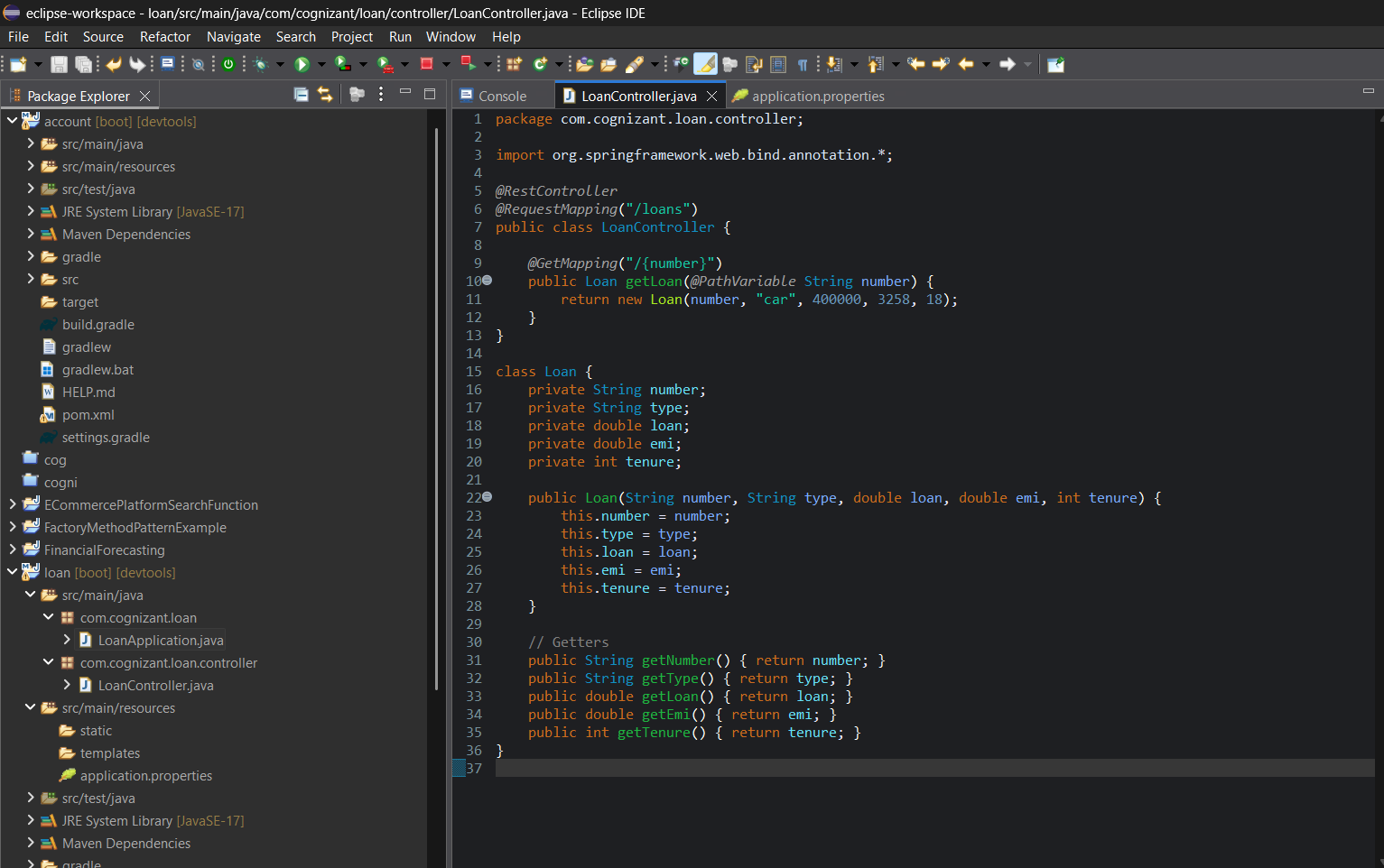
**OUTPUT:**

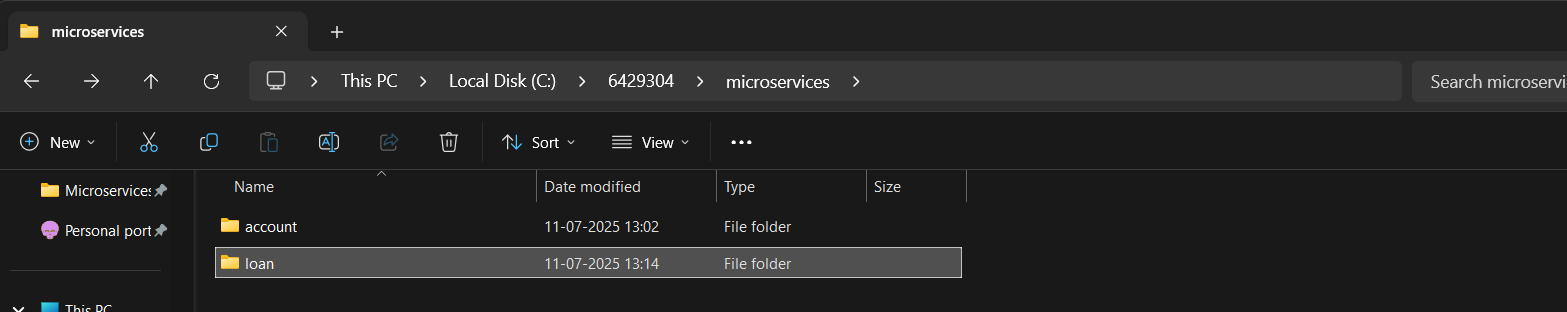
****

****

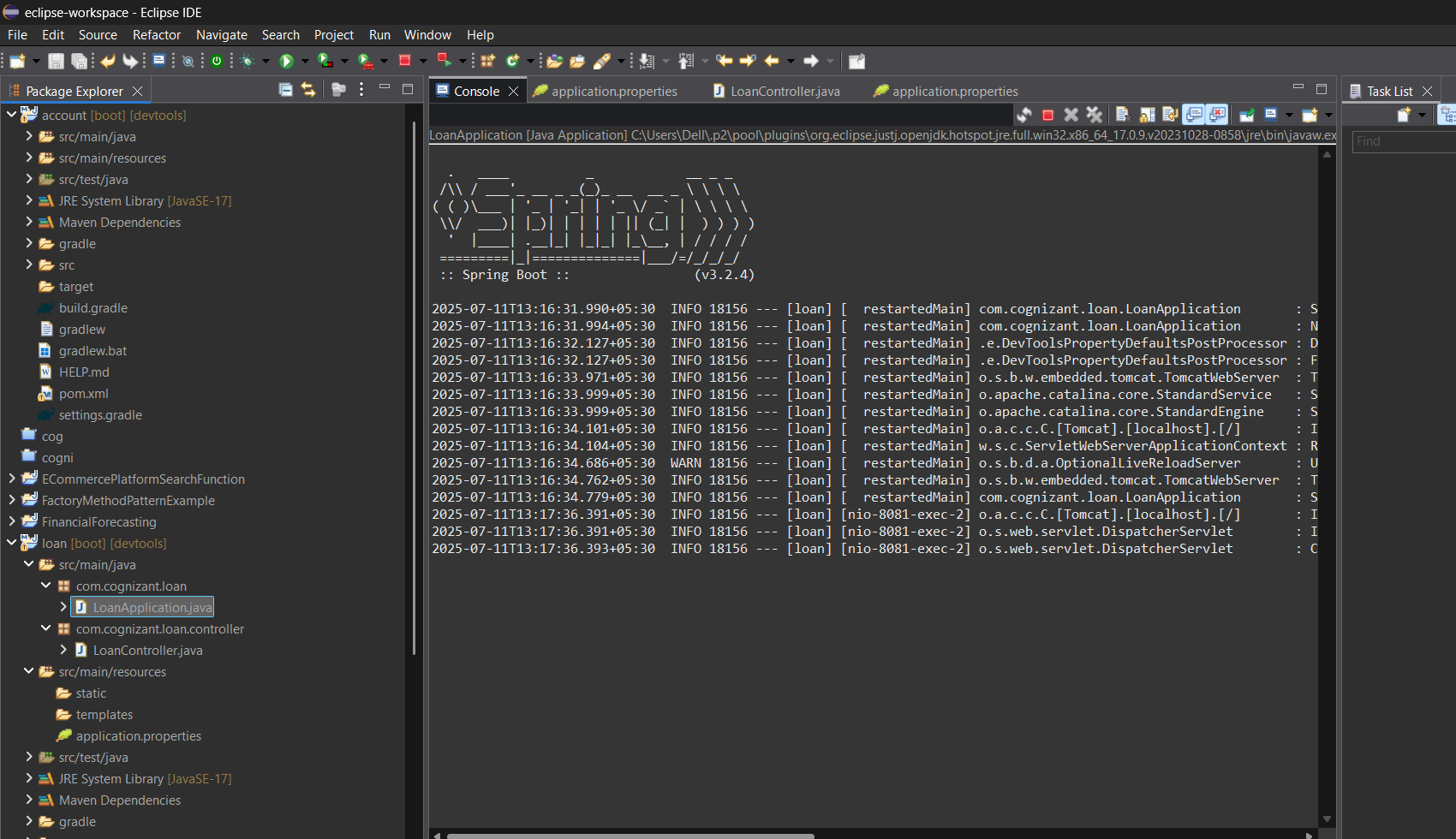
Loan Service CODE:

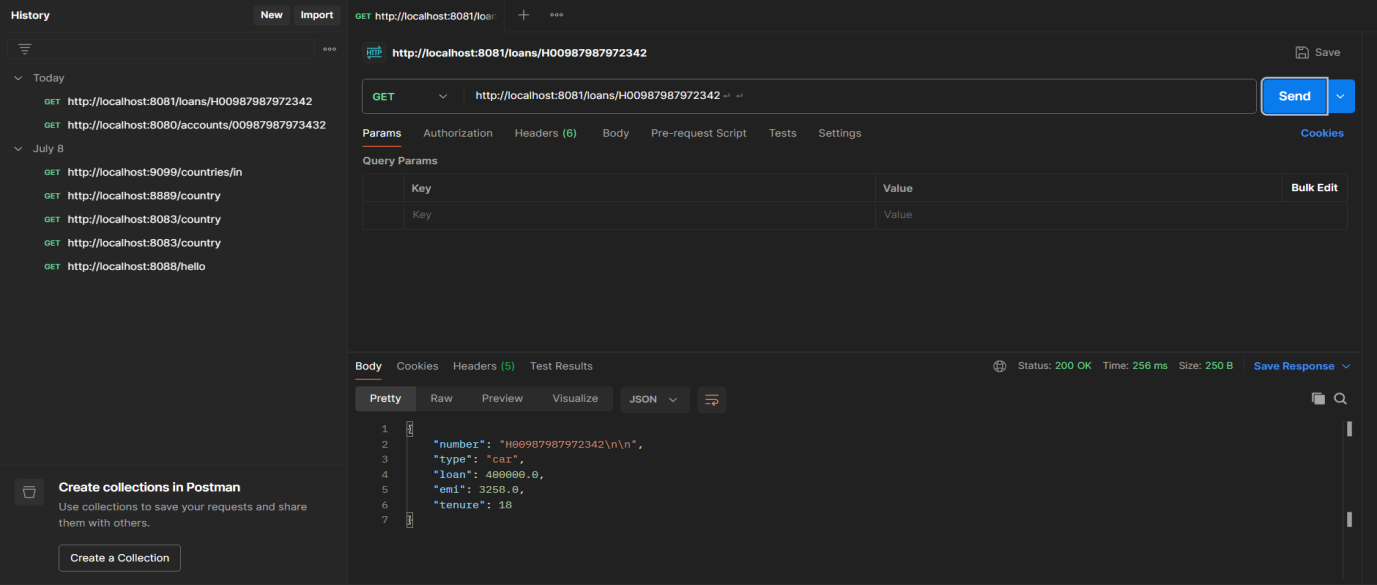
****

****

****

**OUTPUT:**

****

****

**ADDITIONAL HANDS-ON QUESTIONS**

**Question 1**

Create Eureka Discovery Server and register

microservices

Eureka Discovery Server holds a registry of all the services that are available

for immediate consumption. Anybody whom wants to consume a RESTful

Web Service can come to the discovery server and find out what is available

and ready for consumption. Eureka Discovery Server is part of spring cloud

module.

Follow steps below to implement:

Create and Launch Eureka Discovery Server

 Using https://start.spring.io generate a project with following

configuration:

o Group: com.cognizant

o Artifact: eureka-discovery-server

o Module: Spring Cloud Discovery > Eureka Server

 Download the project, build it using maven in command line

 Import the project in Eclipse

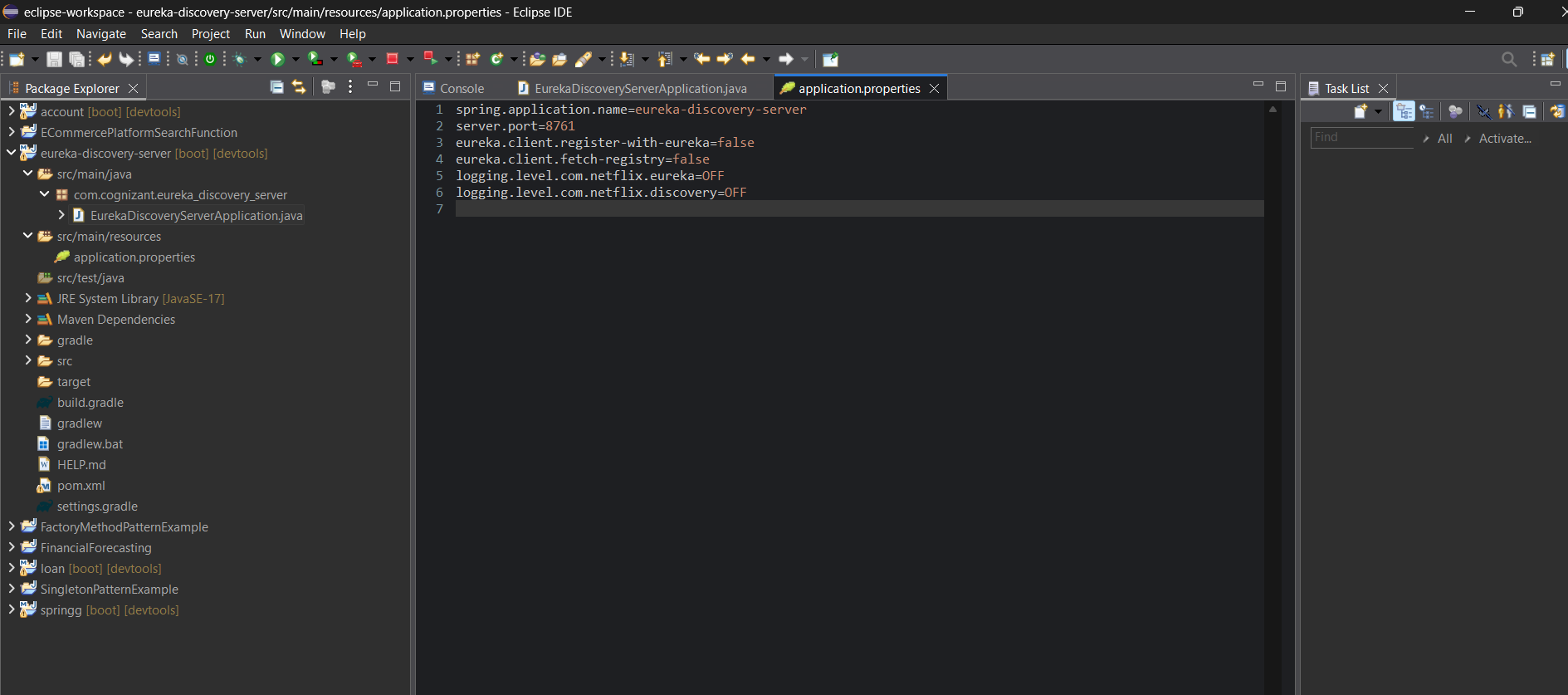
 Include @EnableEurekaServer in

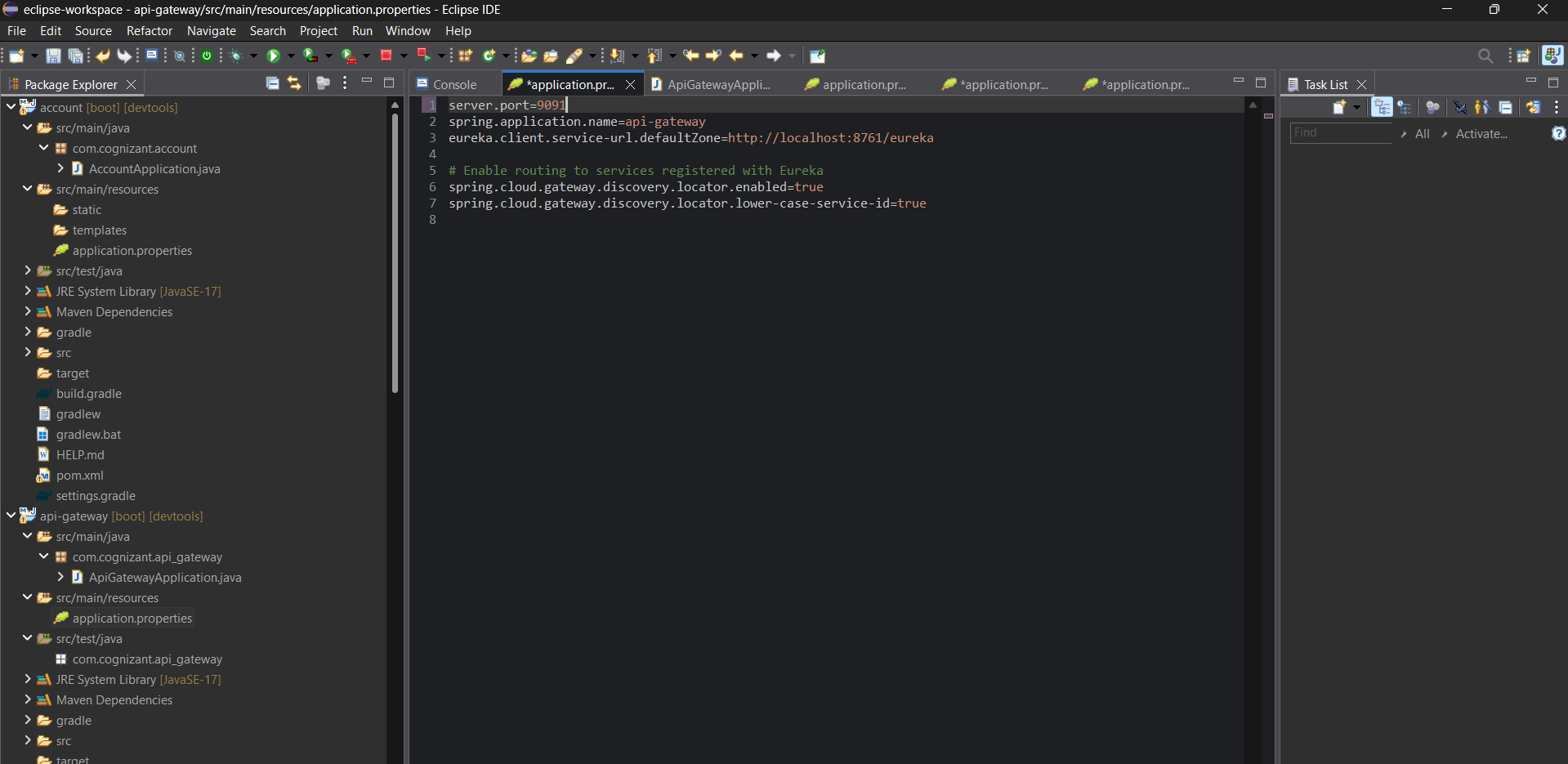
class EurekaDiscoveryServerApplication

 Include the following configurations in application.properties:

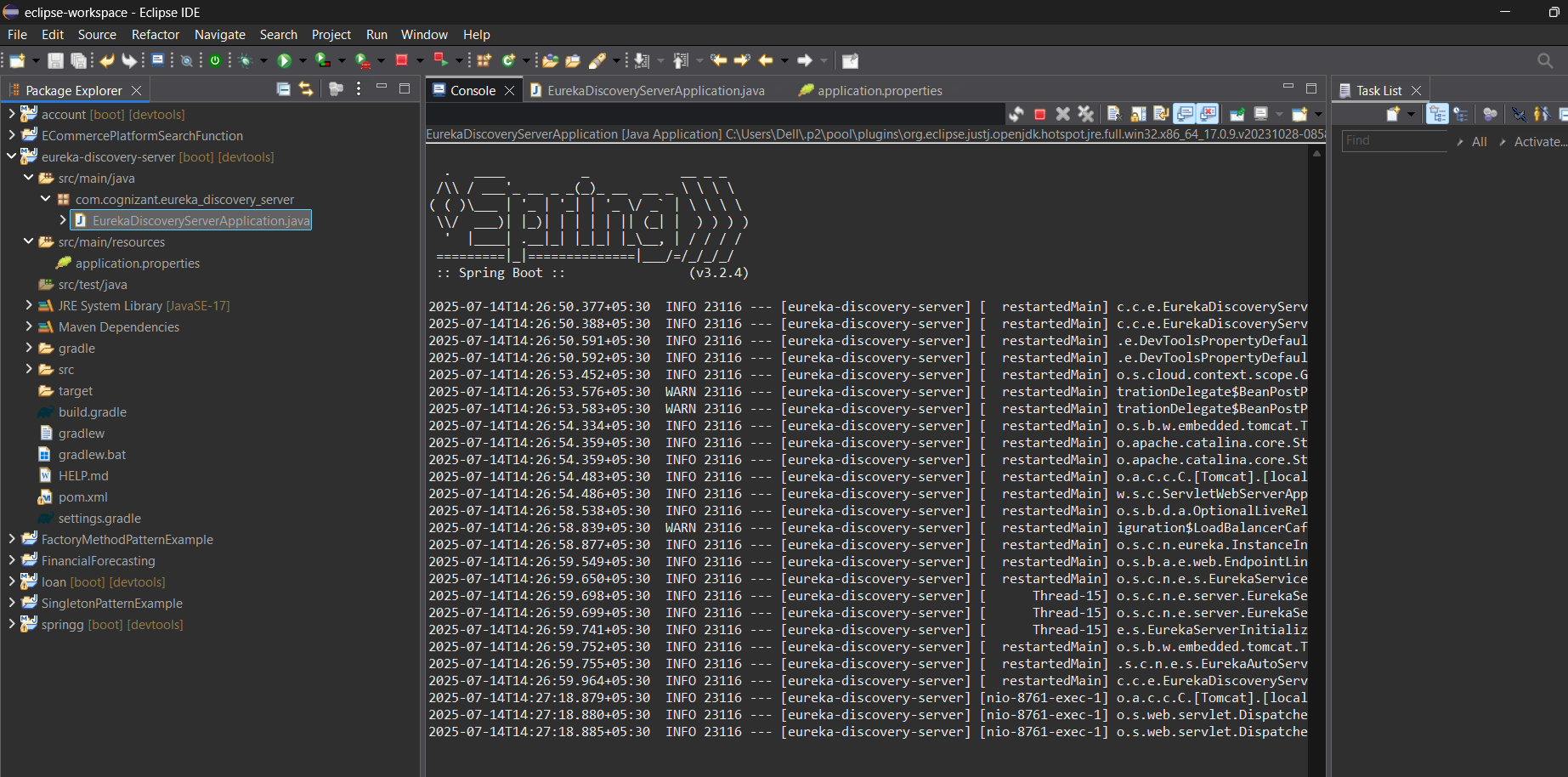
Answer and output:

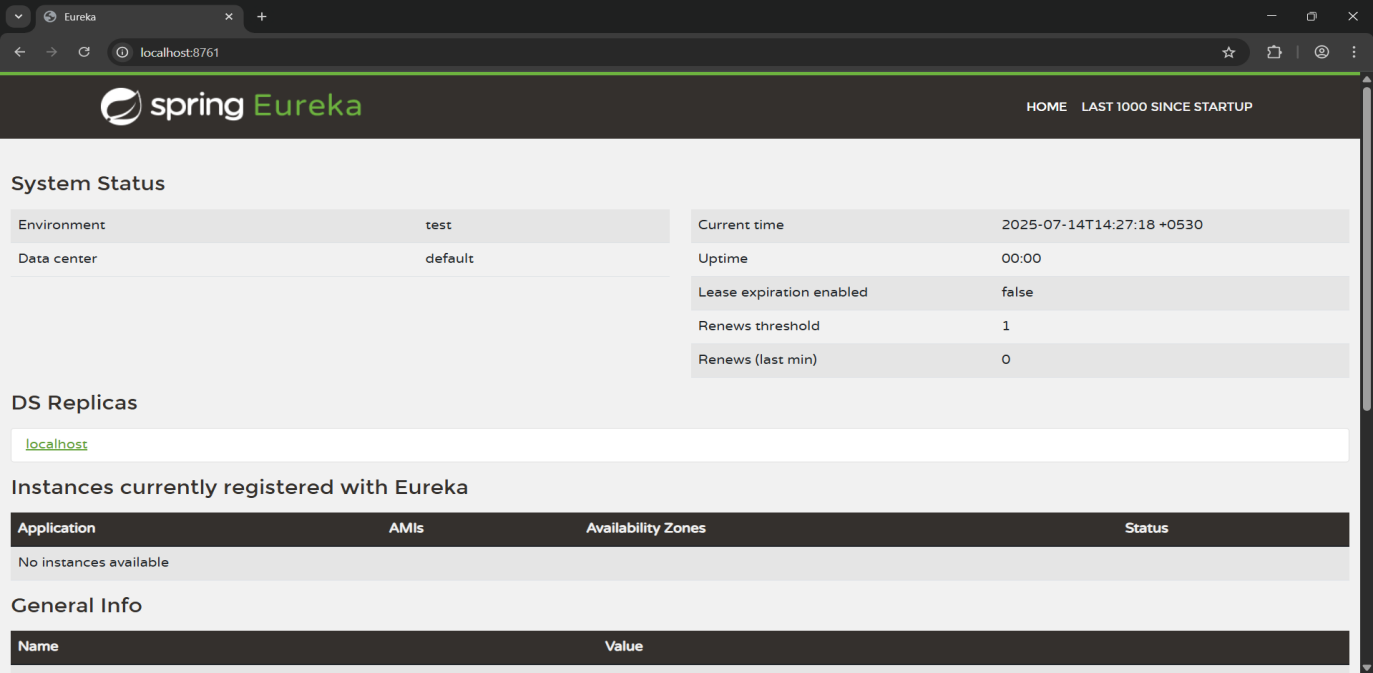
Eureka CODE:

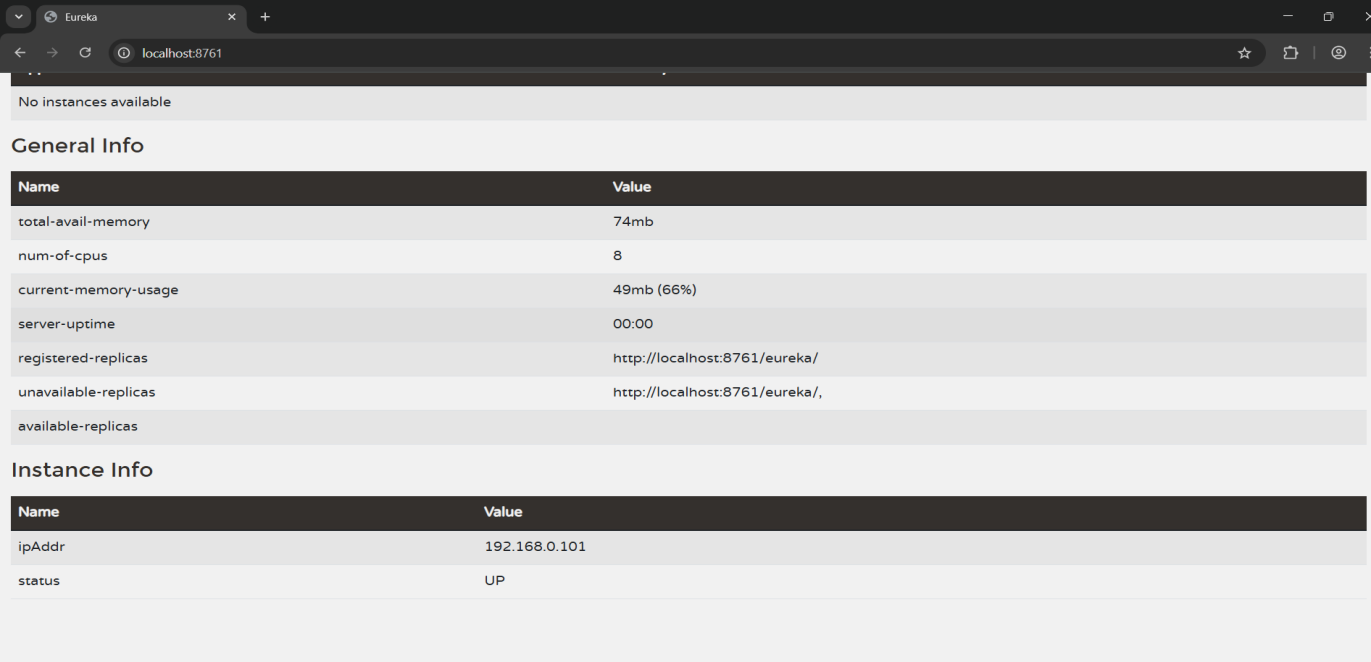




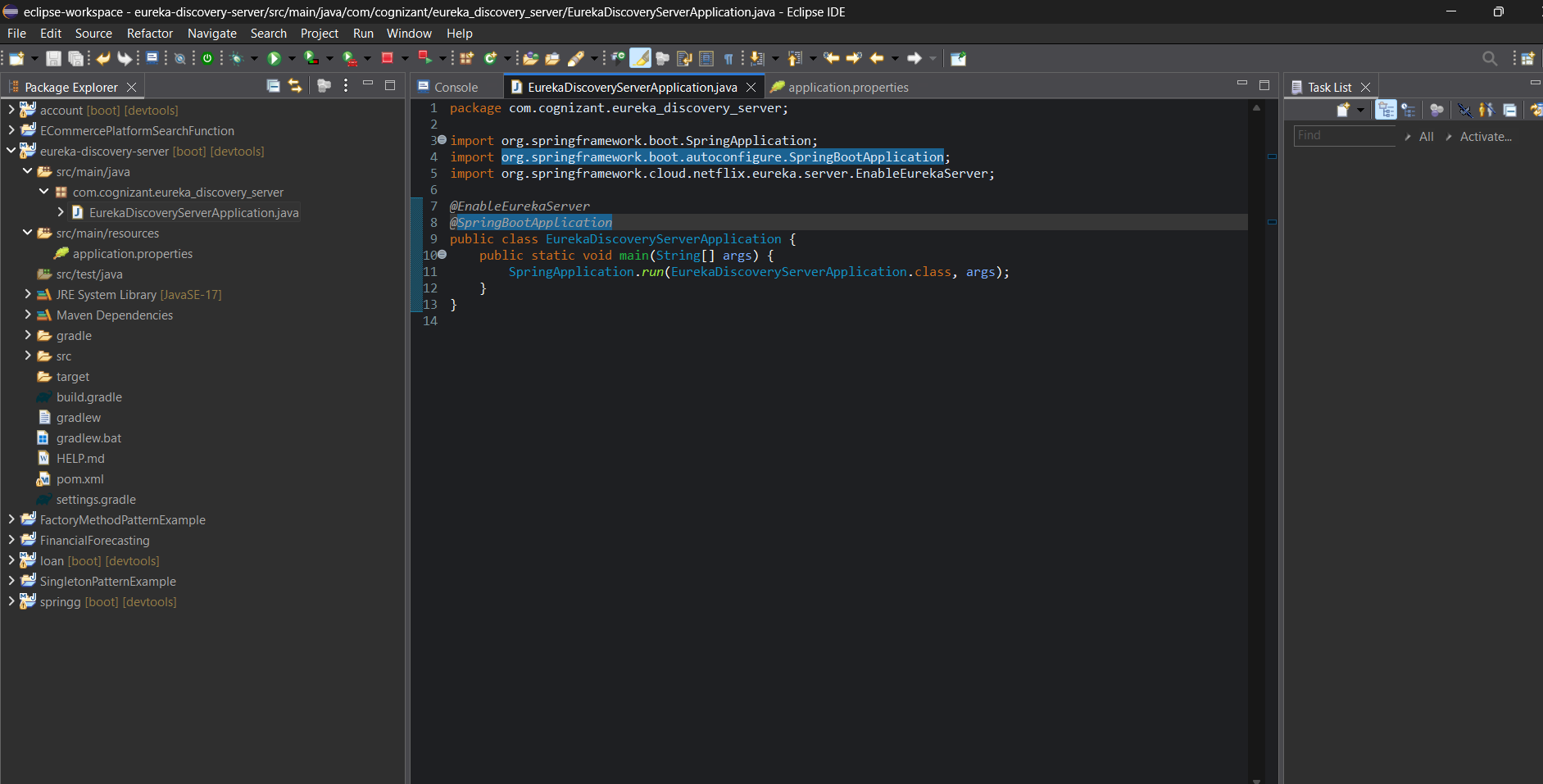
Eureka OUTPUT:

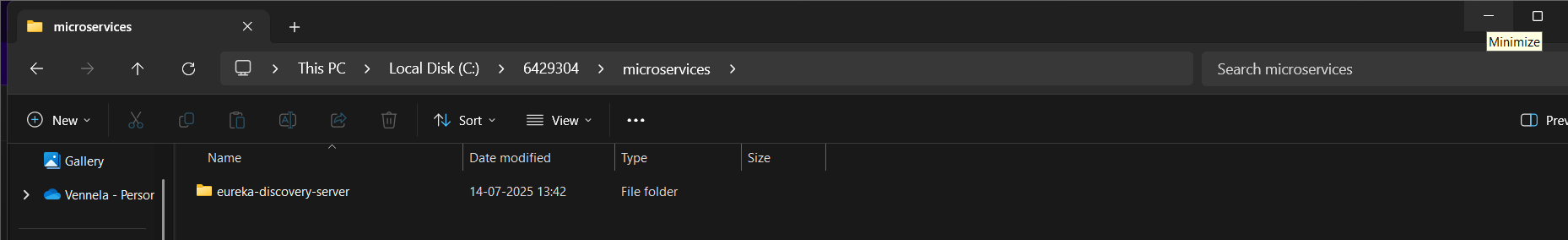




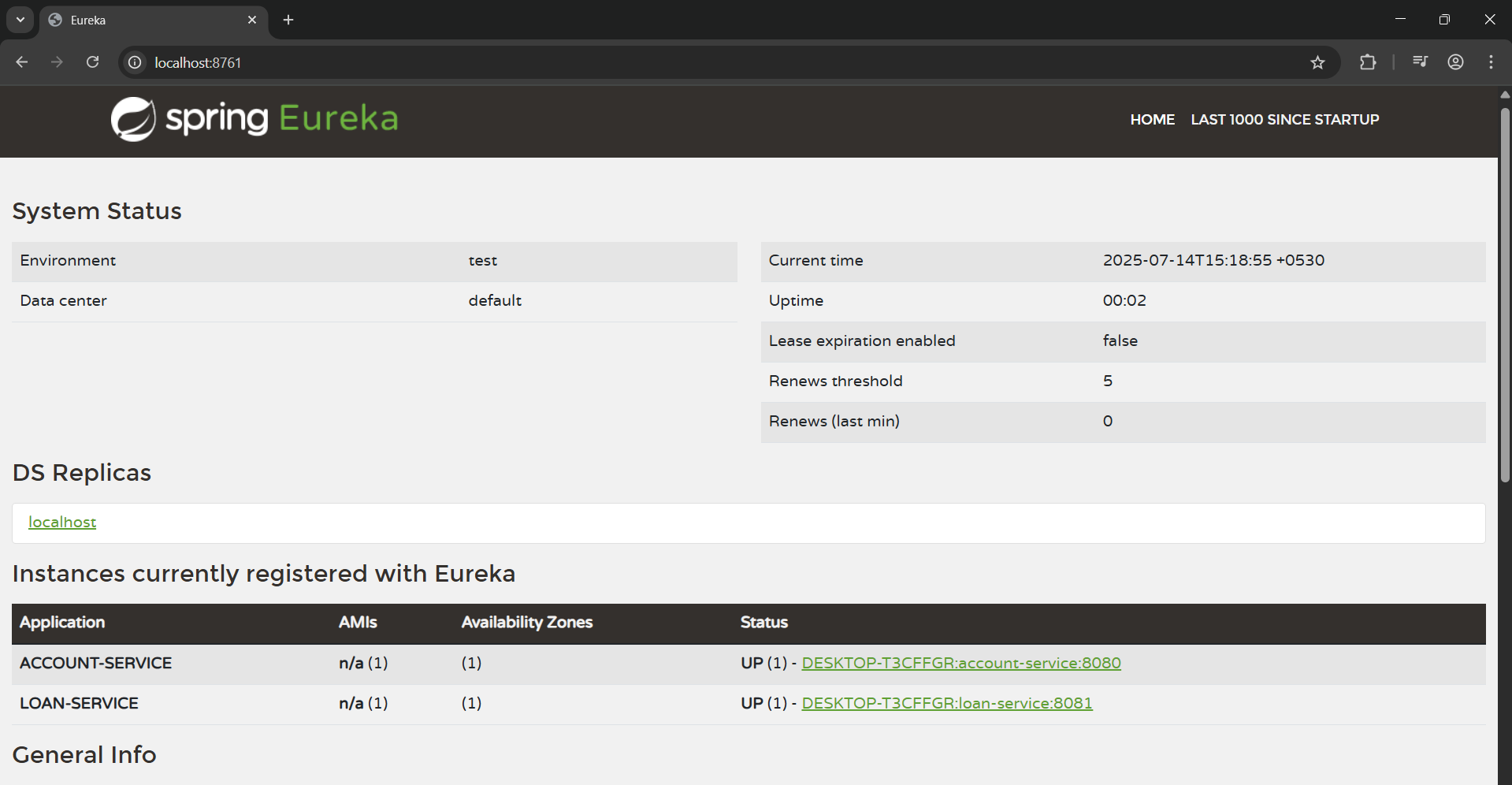


Loan and Account CODE:

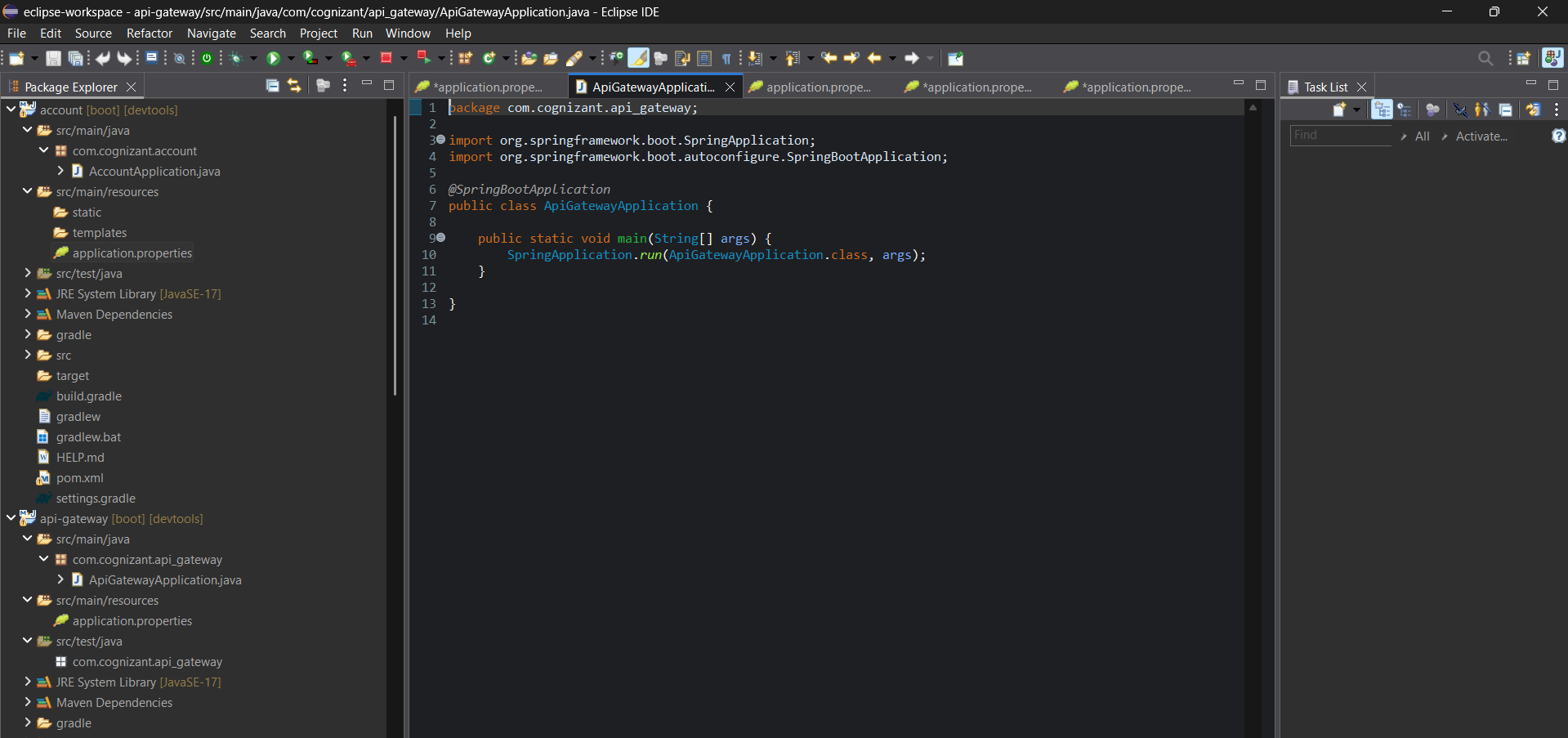




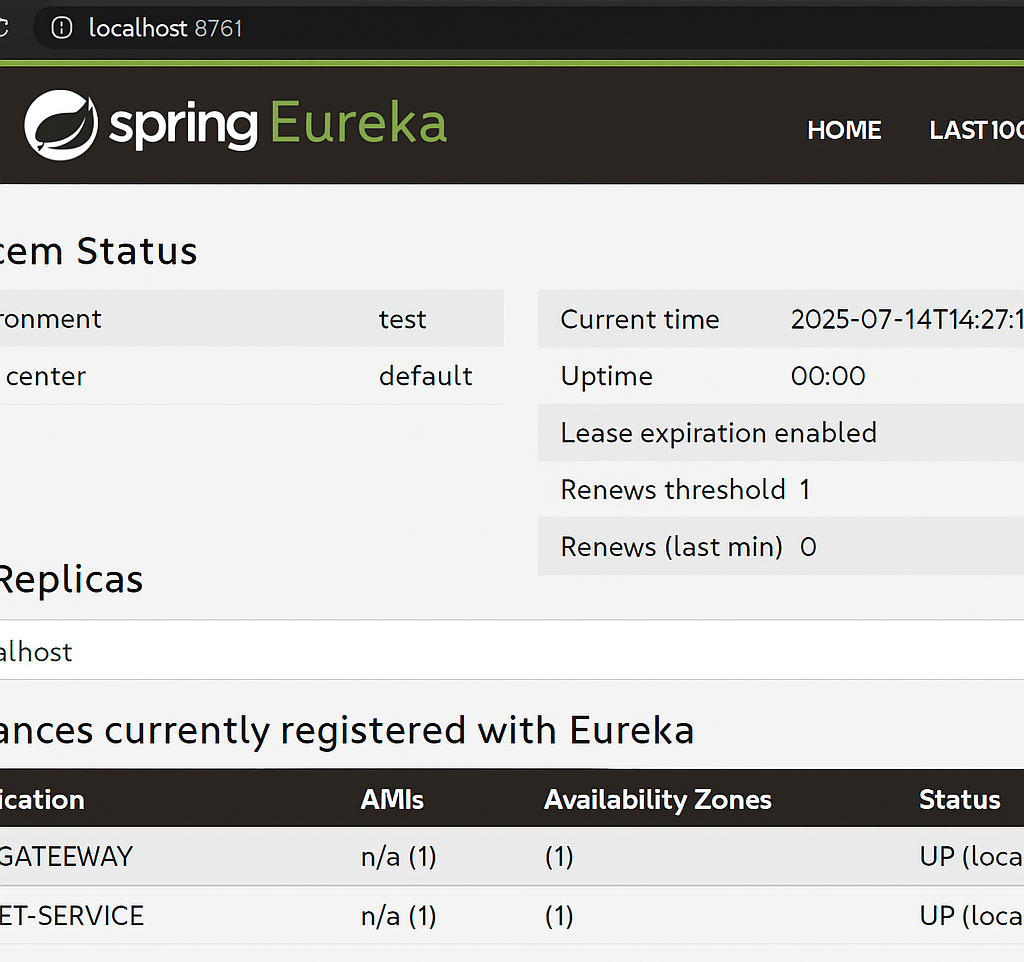
Loan and Account OUTPUT:



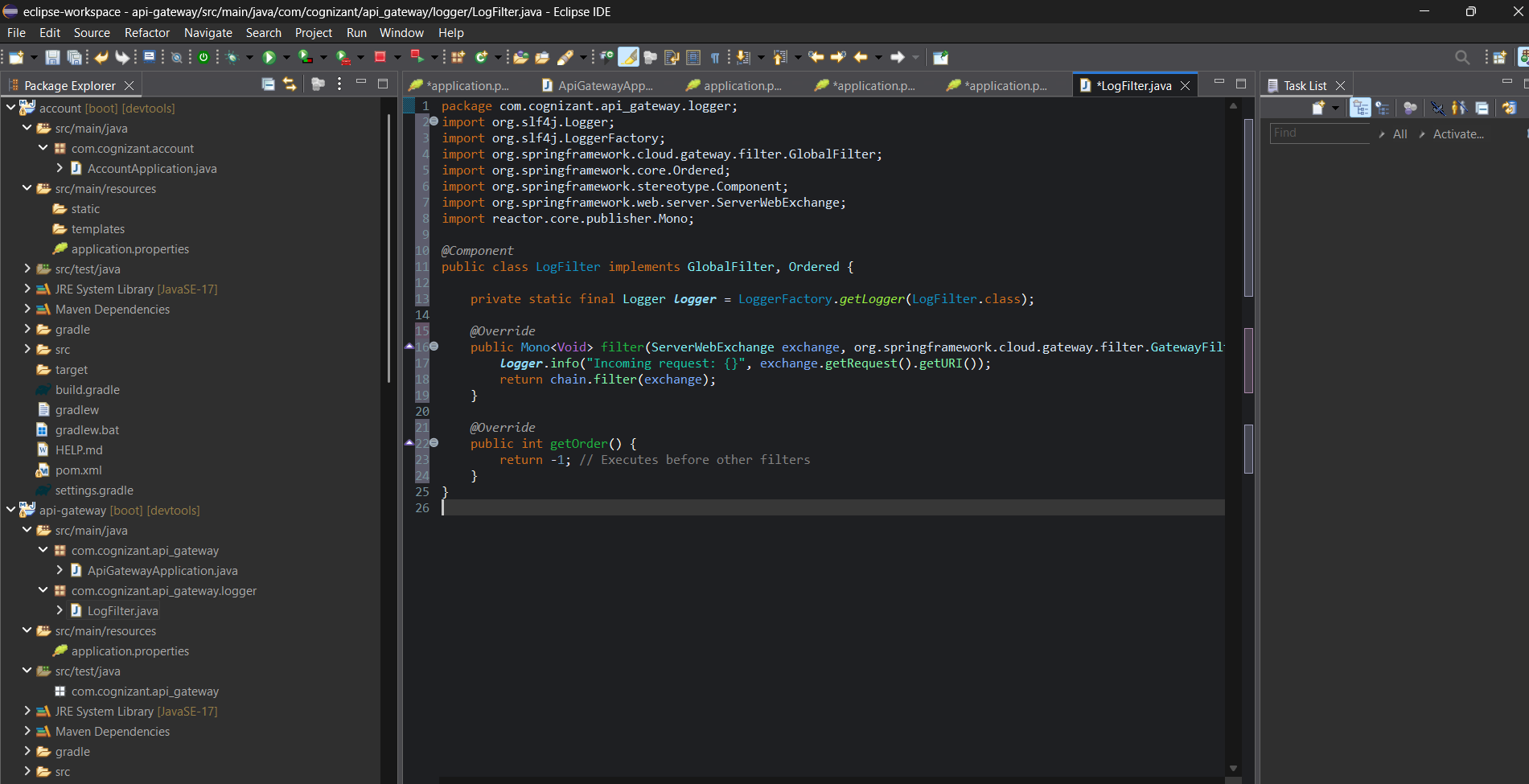
Api-Gateway CODE:



Api-Gateway OUTPUT:



Greet Service CODE:



Greet Service OUTPUT:

