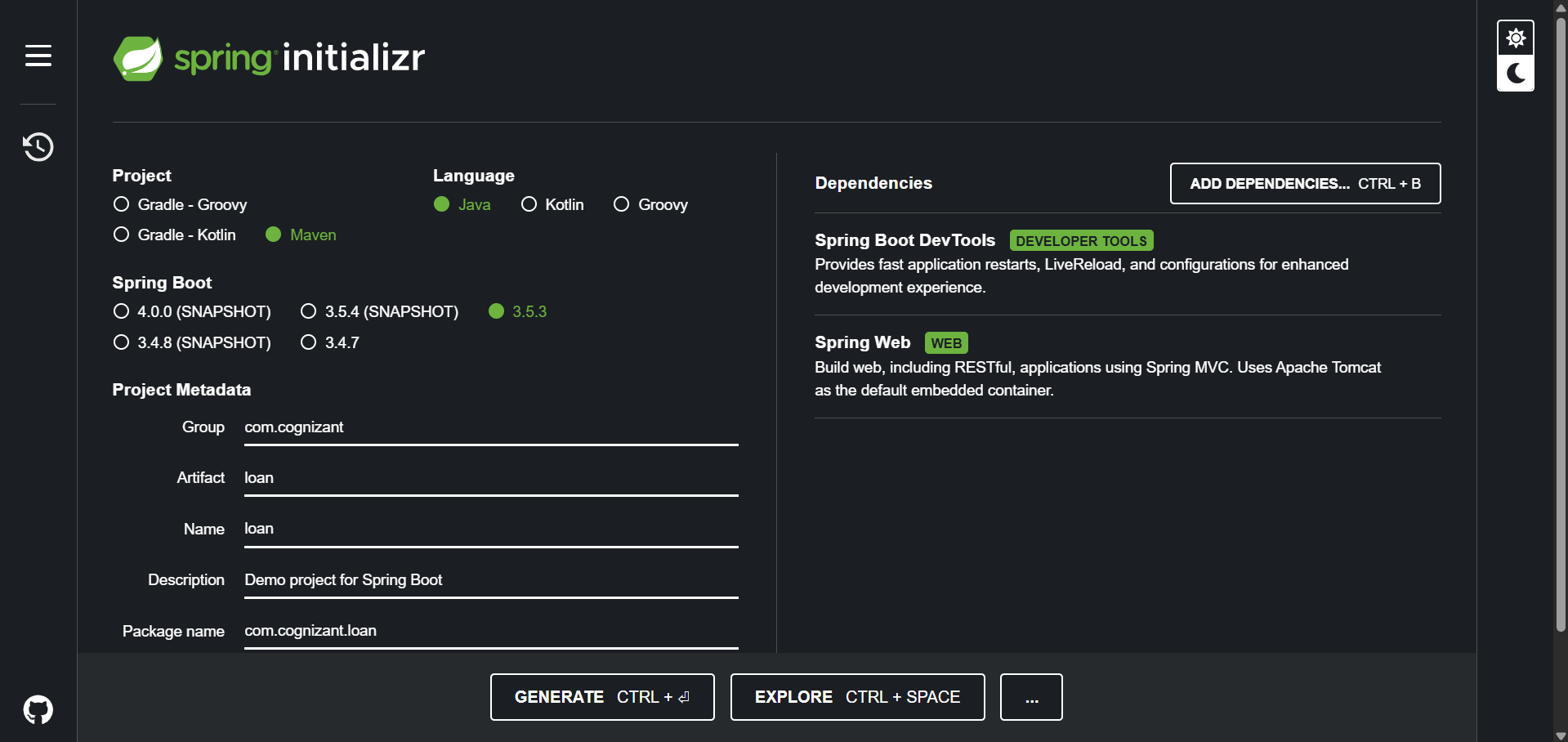
**Microservices with Spring Boot 3 and Spring Cloud**

**WEEK-5:Mandatory and additional questions**

**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.**



**LOAN SERVICES:**

**Application.properties:**

spring.application.name=loan

server.port=8081

LoanController.java:

package com.cognizant.loan.controller;

import org.springframework.web.bind.annotation.\*;

*@RestController*

*@RequestMapping*("/loans")

public class LoanController {

*@GetMapping*("/{number}")

public Loan getLoan(*@PathVariable* String number) {

return new Loan(number, "car", 400000, 3258, 18);

}

public static class Loan {

private String number;

private String type;

private double loan;

private int emi;

private int tenure;

public Loan(String number, String type, double loan, int emi, int tenure) {

this.number = number;

this.type = type;

this.loan = loan;

this.emi = emi;

this.tenure = tenure;

}

public String getNumber() { return number; }

public String getType() { return type; }

public double getLoan() { return loan; }

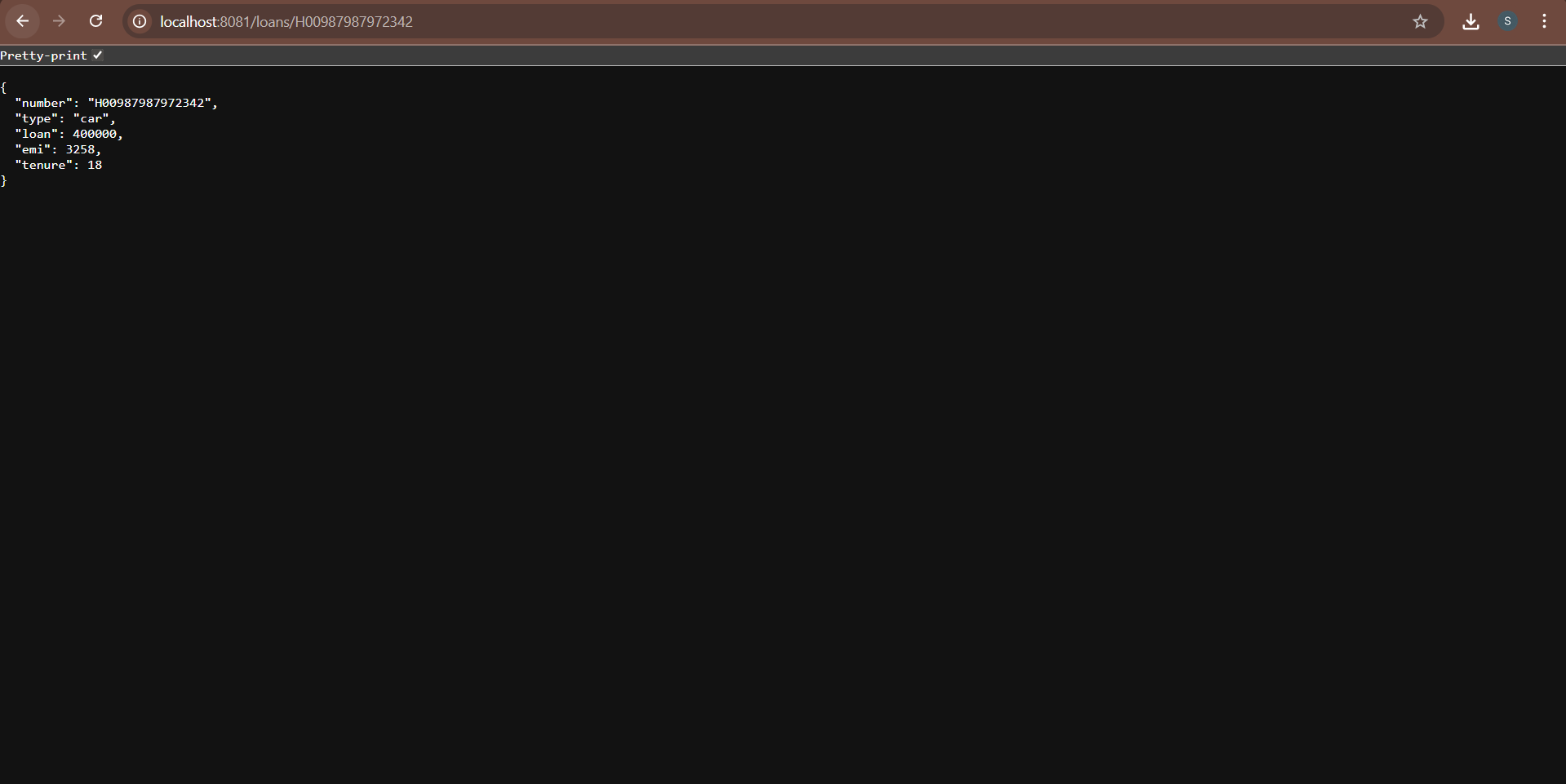
public int getEmi() { return emi; }

public int getTenure() { return tenure; }

}

}

**OUTPUT:**



A computer screen shot of a program

AI-generated content may be incorrect.

**ACCOUNT SERVICES:**

A screenshot of a computer

AI-generated content may be incorrect.

**AccountController.java:**

package com.cognizant.account.controller;

import org.springframework.web.bind.annotation.\*;

*@RestController*

*@RequestMapping*("/accounts")

public class AccountController {

*@GetMapping*("/{number}")

public Account getAccount(*@PathVariable* String number) {

return new Account(number, "savings", 234343);

}

public static class Account {

private String number;

private String type;

private double balance;

public Account(String number, String type, double balance) {

this.number = number;

this.type = type;

this.balance = balance;

}

public String getNumber() { return number; }

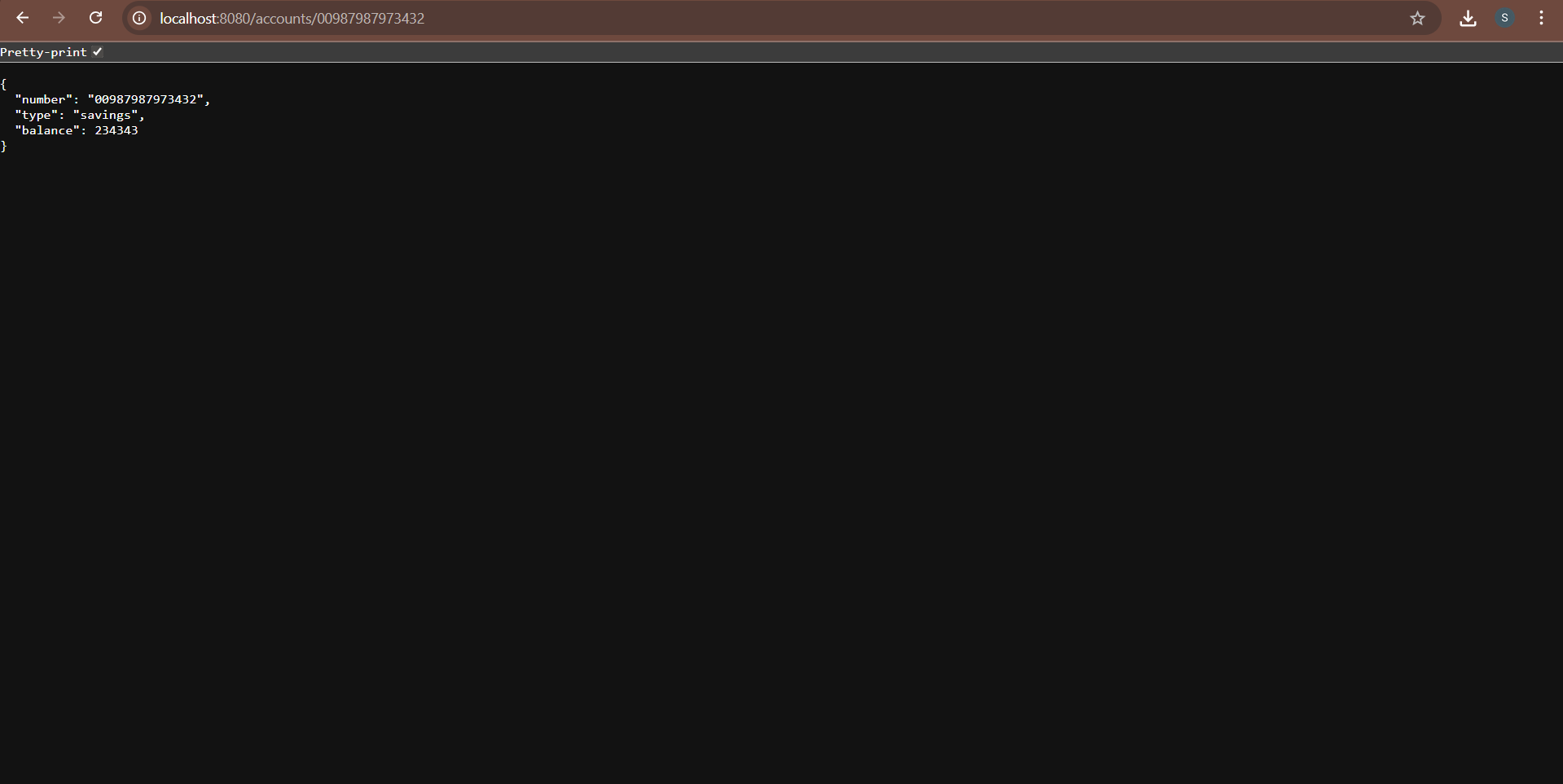
public String getType() { return type; }

public double getBalance() { return balance; }

}

}

**OUTPUT:**



A screenshot of a computer program

AI-generated content may be incorrect.

**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.**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**EurekaDiscoveryServerApplication.java:**

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

public class EurekaDiscoveryServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaDiscoveryServerApplication.class, args);

}

}

application.properties:

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

logging.level.com.netflix.eureka=OFF

logging.level.com.netflix.discovery=OFF

AccountApplication.java:

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class AccountApplication {

public static void main(String[] args) {

SpringApplication.run(AccountApplication.class, args);

}

}

**application.properties:**

server.port=8080

spring.application.name=account-service

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

LoanApplication.java:

@EnableDiscoveryClient

@SpringBootApplication

public class LoanApplication {

public static void main(String[] args) {

SpringApplication.run(LoanApplication.class, args);

}

}

**application.properties;**

server.port=8081

spring.application.name=loan-service

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

GreetController.java:

@RestController

public class GreetController {

@GetMapping("/greet")

public String greet() {

return "Hello World";

}

}

**application.properties:**

server.port=8082

spring.application.name=greet-service

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

**LogFilter.java:**

@Component

public class LogFilter implements GlobalFilter, Ordered {

private static final Logger logger = LoggerFactory.getLogger(LogFilter.class);

@Override

public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {

logger.info("Incoming request: {}", exchange.getRequest().getURI());

return chain.filter(exchange);

}

@Override

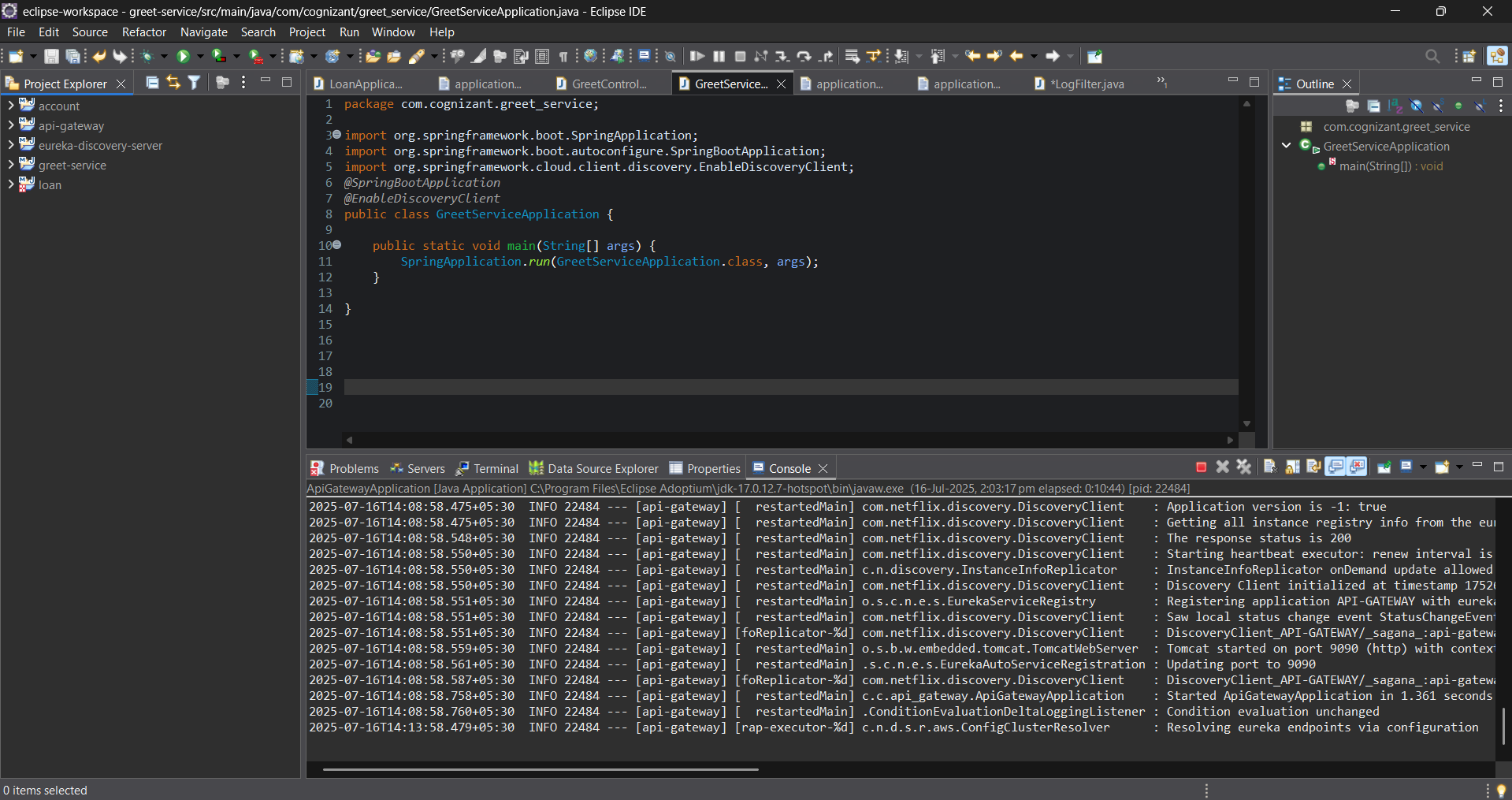
public int getOrder() {

return -1;

}

}

**OUTPUT:**



A screenshot of a computer

AI-generated content may be incorrect.