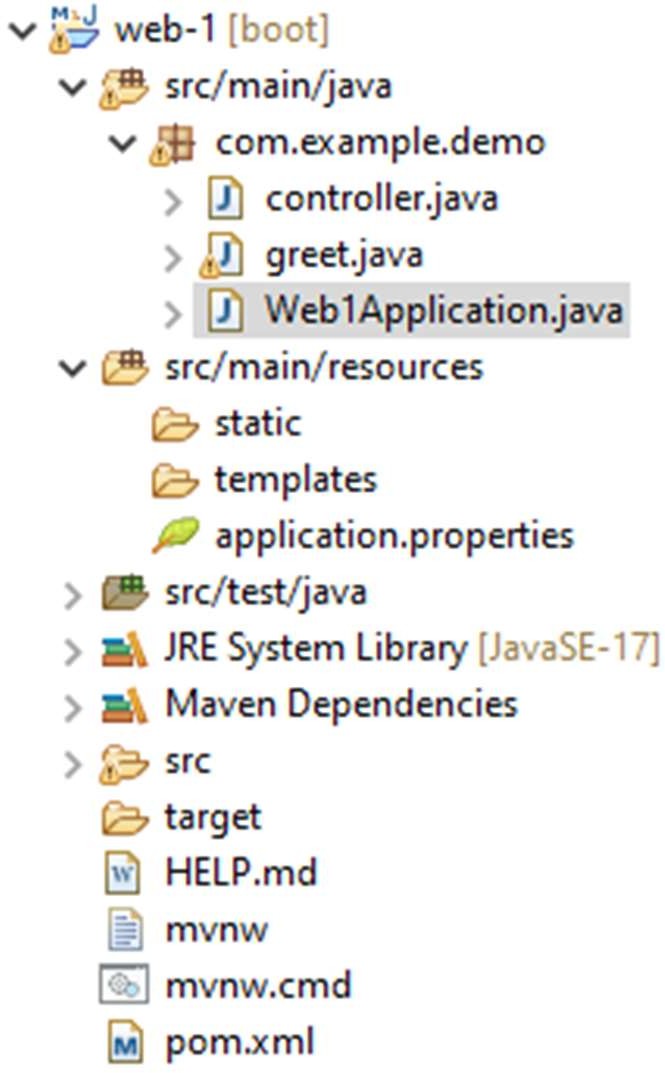
### Practical No.10

**Practical 10.1**

**Aim:** Write a program to create a simple Spring Boot application that prints a message. Code:





greet.java

greet.java

**package** com.example.demo;

**import** org.springframework.web.bind.annotation.GetMapping; **import** org.springframework.web.bind.annotation.RestController; @RestController

**public class** greet{ @GetMapping("/greet") **public** String greet() {

**return** "Hello, chirag";

}

}



Web1Application.java

Web1Application.java package com.example.demo;

import org.springframework.boot.SpringApplication;

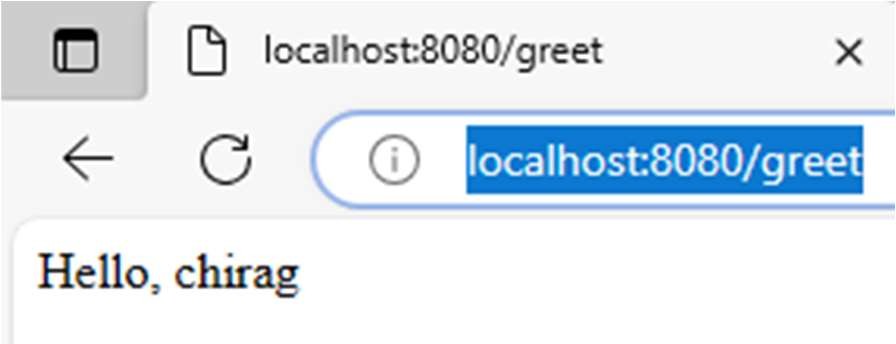
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication public class Web1Application {

public static void main(String[] args) { SpringApplication.run(Web1Application.class, args);

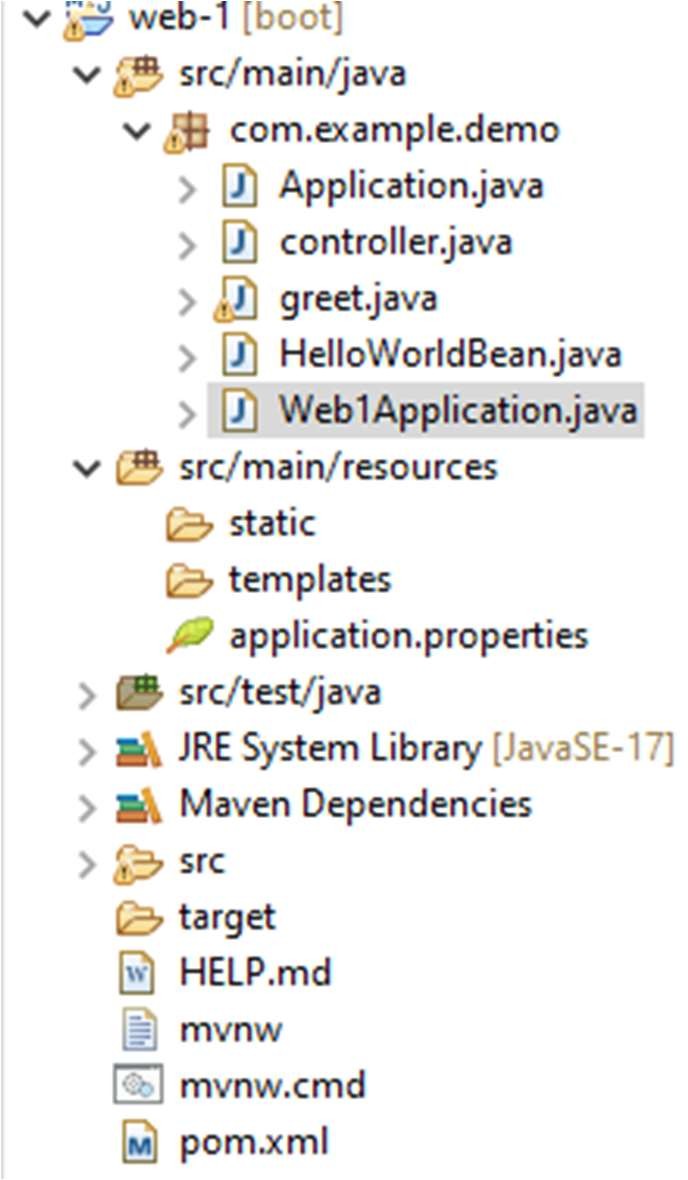
}

}

Output:

### Practical 10.2

**Aim:** Write a program to demonstrate RESTful Web Services with spring boot. Code:



controller.java

Controller.java:

**package** com.example.demo;

**import** org.springframework.web.bind.annotation.GetMapping; **import** org.springframework.web.bind.annotation.RestController; @RestController

**public class** controller { @GetMapping("/hello-world") **public** String helloWorld() {

**return** "Hey Mr";

}

@GetMapping("/hello-world-bean")

**public** HelloWorldBean helloWorldBean() {

**return new** HelloWorldBean("Hey");

}

}



HelloWorldBean.java

HelloWorldBean.java:

**package** com.example.demo;**public class** HelloWorldBean {

**private** String message;

**public** HelloWorldBean(String message) {

**this**.message = message;

}

**public** String getMessage() {

**return** message;

}

**public void** setMessage(String message) {

**this**.message = message;

}

}



Application.java

Application.java:

**package** com.example.demo;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

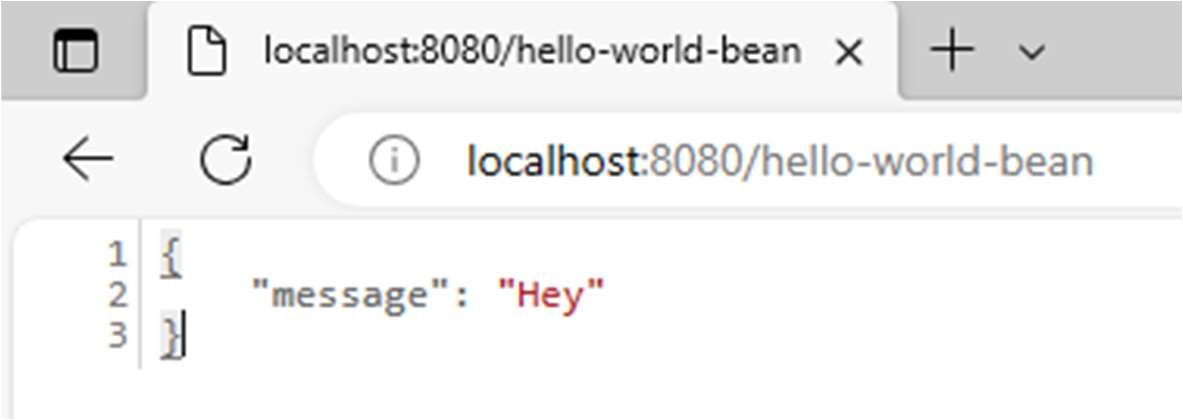
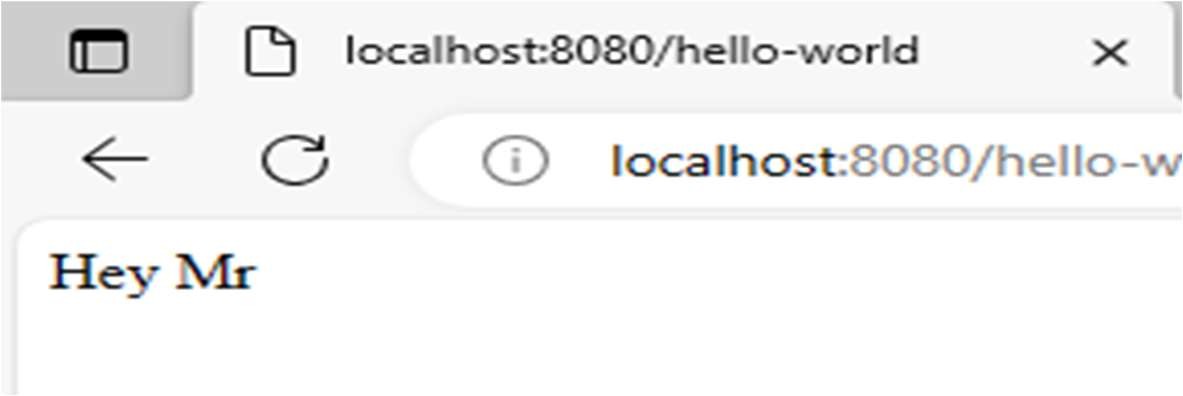
**public class** Application {

**public static void** main(String[] args) { SpringApplication.*run*(Application.**class**, args);

}

}

Output:



### Practical-8

**Practical 8.1**

Aim: Write a program to demonstrate Spring AOP – before advice.

Code:

Application.java:

**package** com.example;

**import** com.example.service.YourService;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.context.ConfigurableApplicationContext; @SpringBootApplication

**public class** YourApplication {

**public static void** main(String[] args) {

ConfigurableApplicationContext context = SpringApplication.*run*(YourApplication.**class**, args);

YourService yourService = context.getBean(YourService.**class**); yourService.yourmethod();

}

}

Aspect.java:

**package** com.example.aspect;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.aspectj.lang.annotation.Before;

**import** org.springframework.stereotype.Component; @Aspect

@Component

**public class** myaspect {

@Before("execution(\* com.example.service.YourService.yourmethod())")

**public void** beforeAdvice() { System.***out***.println("Before advice executed.");

}

}

### YourService.java

**package** com.example.service;

**import** org.springframework.stereotype.Service; @Service

**public class** YourService {

**public void** yourmethod() { System.***out***.println("Your method executed.");

}

}

In pom.xml only change

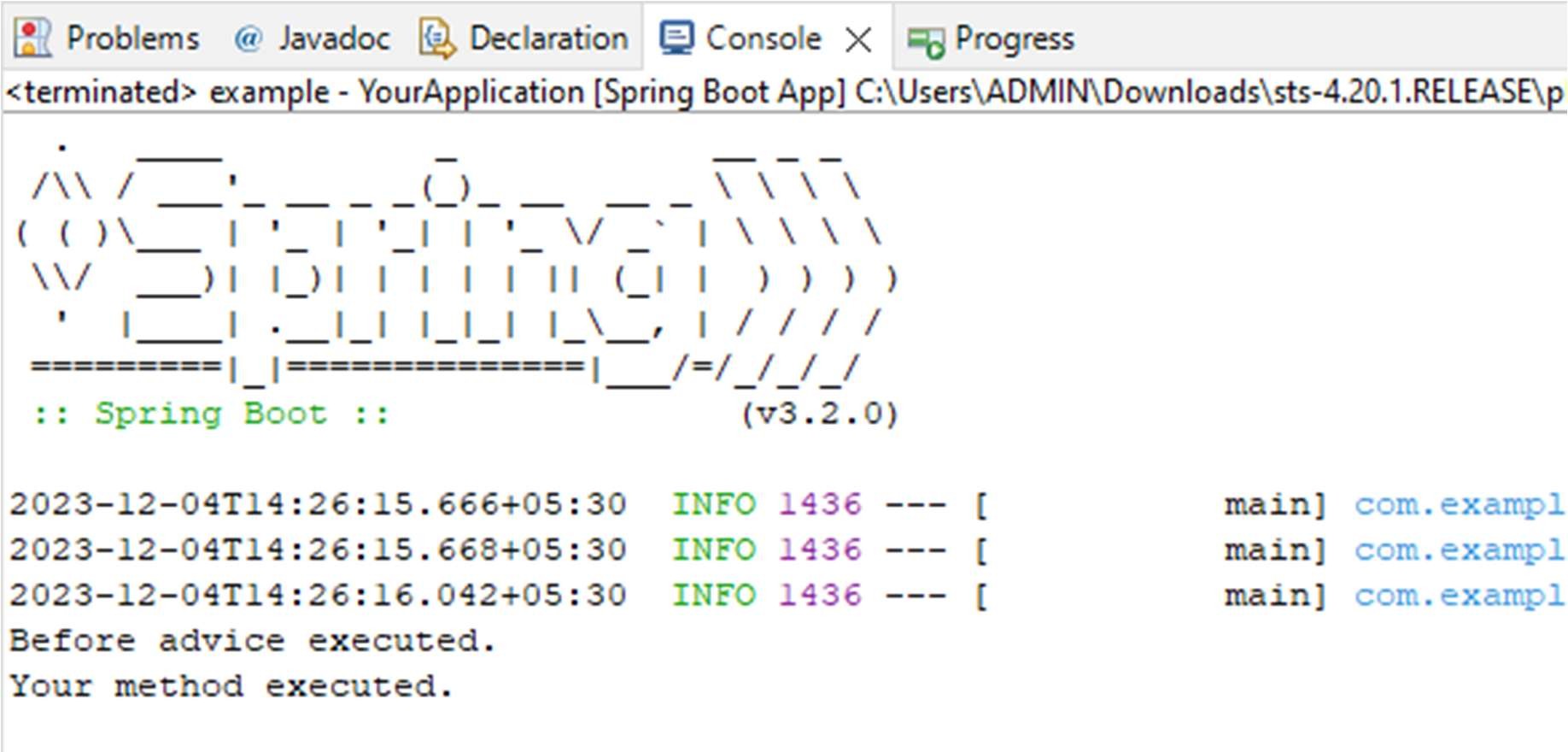
<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.0</version>

</parent> Output:



### Practical 8.2

Aim: Write a program to demonstrate Spring AOP – after advice. Code:

**package** com.example;

**import** com.example.service.YourService;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.context.ConfigurableApplicationContext; @SpringBootApplication

**public class** YourApplication {

**public static void** main(String[] args) {

ConfigurableApplicationContext context = SpringApplication.*run*(YourApplication.**class**, args);

YourService yourService = context.getBean(YourService.**class**); yourService.yourmethod();

}

}

Aspect.java:

**package** com.example.aspect;

**import** org.aspectj.lang.annotation.After;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.springframework.stereotype.Component; @Aspect

@Component

**public class** myaspect {

@After("execution(\* com.example.service.YourService.yourmethod())")

**public void** beforeAdvice() { System.***out***.println("After advice executed.");

}

}

### YourService.java

**package** com.example.service;

**import** org.springframework.stereotype.Service; @Service

**public class** YourService {

**public void** yourmethod() { System.***out***.println("Your method executed.");

}

}

In pom.xml only change

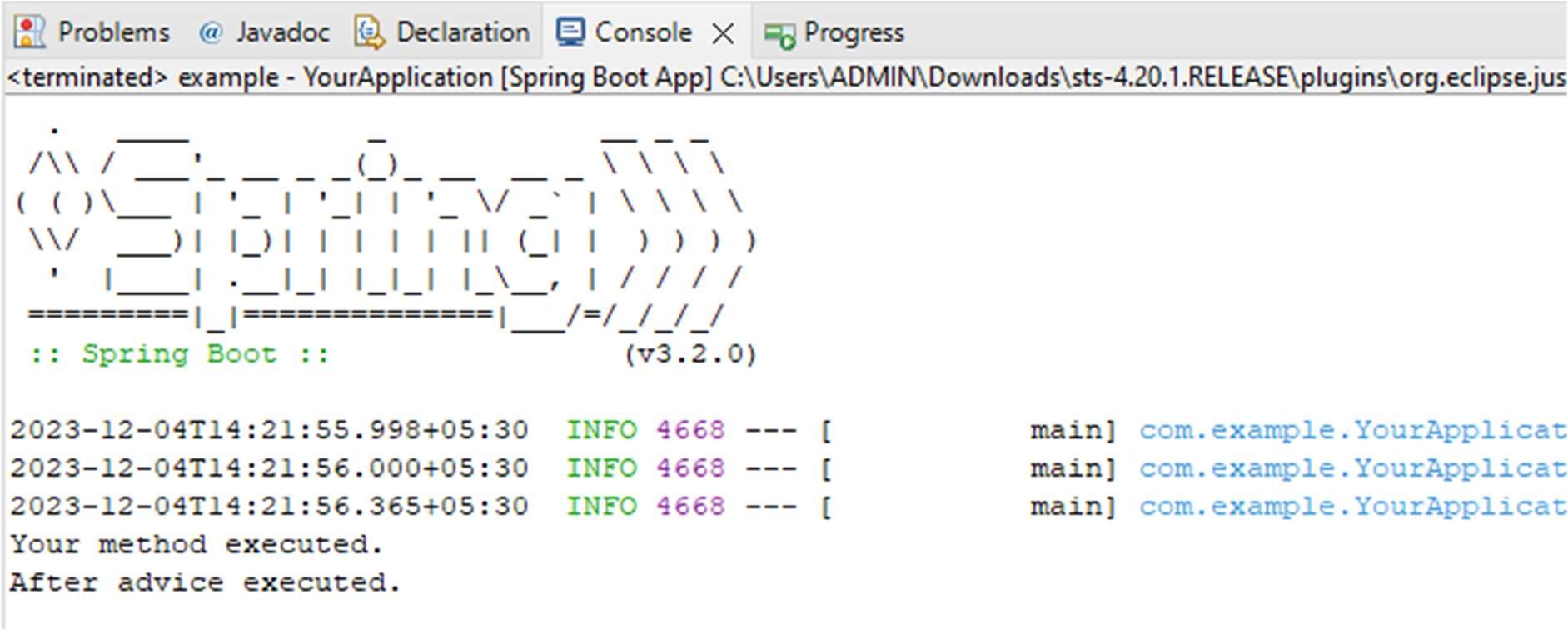
<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.0</version>

</parent> Output:



### Practical 8.3

Aim: Write a program to demonstrate Spring AOP – Around advice. Code:

Application.java:

**package** com.example;

**import** com.example.service.YourService;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.context.ConfigurableApplicationContext; @SpringBootApplication

**public class** YourApplication {

**public static void** main(String[] args) {

ConfigurableApplicationContext context = SpringApplication.*run*(YourApplication.**class**, args);

YourService yourService = context.getBean(YourService.**class**); yourService.yourmethod();

}

}

Aspect.java:

**package** com.example.aspect;

**import** org.aspectj.lang.ProceedingJoinPoint; **import** org.aspectj.lang.annotation.Around; **import** org.aspectj.lang.annotation.Aspect;

**import** org.springframework.stereotype.Component; @Aspect

@Component

**public class** myaspect {

@Around("execution(\* com.example.service.YourService.yourmethod())")

**public void** aspect(ProceedingJoinPoint joinPoint) {

### try {

System.***out***.println("Before Advice"); joinPoint.proceed(); // Proceed with the original method System.***out***.println("After Advice");

} **catch** (Throwable e) {

System.***err***.println("Exception in advice: " + e.getMessage());

}

}

}

### YourService.java

**package** com.example.service;

**import** org.springframework.stereotype.Service; @Service

**public class** YourService {

**public void** yourmethod() { System.***out***.println("Your method executed.");

}

}

In pom.xml only change

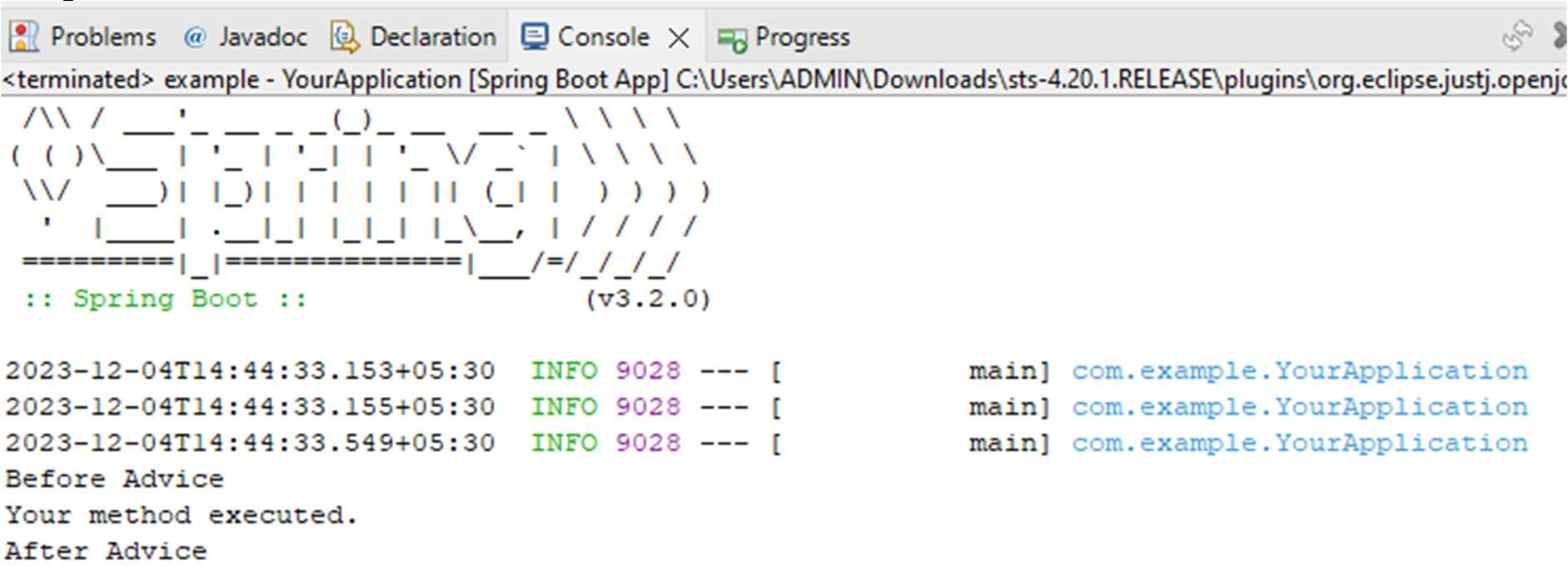
<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.0</version>

</parent> Output:



### Practical 8.4

Aim: Write a program to demonstrate Spring AOP – After Returning advice. Code:

Application.java:

**package** com.example;

**import** com.example.service.YourService;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.context.ConfigurableApplicationContext; @SpringBootApplication

**public class** YourApplication {

**public static void** main(String[] args) {

ConfigurableApplicationContext context = SpringApplication.*run*(YourApplication.**class**, args);

YourService yourService = context.getBean(YourService.**class**); yourService.yourmethod();

}

}

Aspect.java:

**package** com.example.aspect;

**import** org.aspectj.lang.annotation.AfterReturning;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.springframework.stereotype.Component; @Aspect

@Component

**public class** myaspect { @AfterReturning(

pointcut = "execution(\* com.example.service.YourService.yourmethod())", returning = "result"

)

**public void** afterReturningAdvice() {

System.***out***.println("After Returning Advice: Aspect executed after service2()returns");

}

}

### YourService.java

**package** com.example.service;

**import** org.springframework.stereotype.Service; @Service

**public class** YourService {

**public void** yourmethod() { System.***out***.println("Your method executed.");

}

}

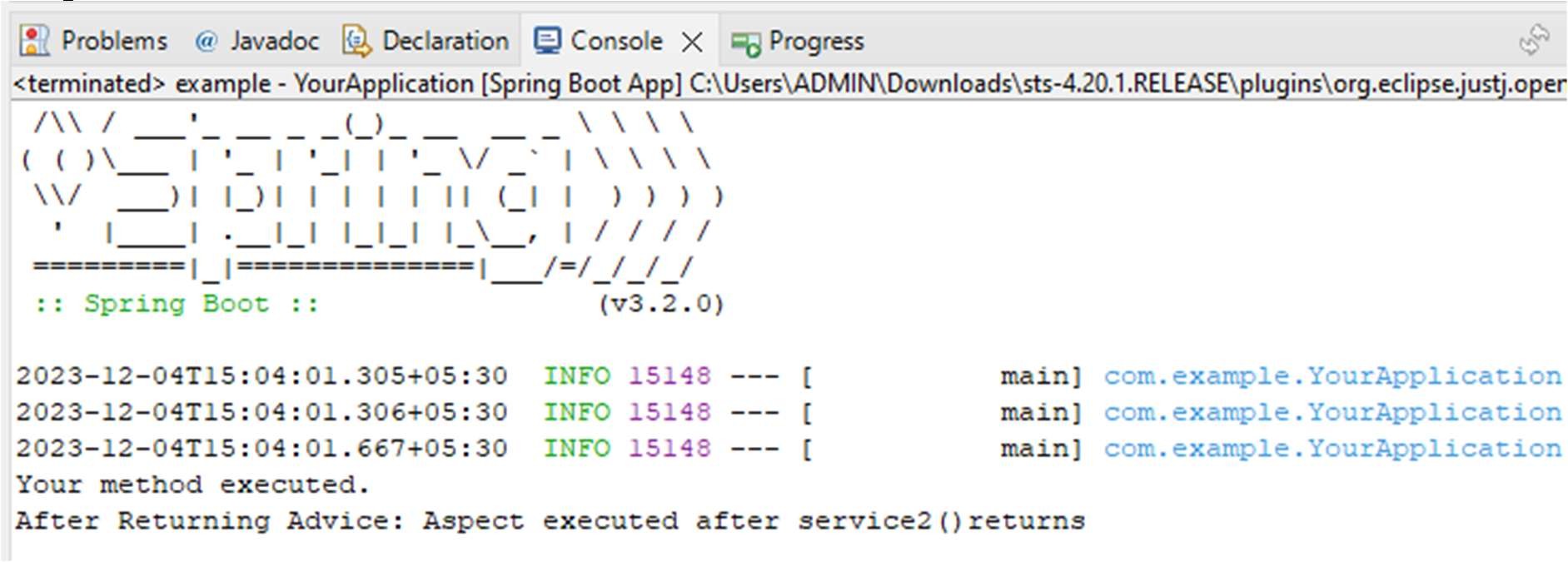
In pom.xml only change

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.0</version>

</parent> Output:

### Practical 8.5

Aim: Write a program to demonstrate Spring AOP – After throwing advice. Code:

Application.java:

**package** com.example;

**import** com.example.service.YourService;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.context.ConfigurableApplicationContext; @SpringBootApplication

**public class** YourApplication {

**public static void** main(String[] args) {

ConfigurableApplicationContext context = SpringApplication.*run*(YourApplication.**class**, args);

YourService yourService = context.getBean(YourService.**class**); yourService.yourmethod();

}

}

Aspect.java:

**package** com.example.aspect;

**import** org.aspectj.lang.annotation.AfterThrowing;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.springframework.stereotype.Component; @Aspect

@Component

**public class** myaspect { @AfterThrowing(

pointcut = "execution(\* com.example.service.YourService.yourmethod())", throwing = "result"

)

**public void** afterReturningAdvice() {

System.***out***.println("After Returning Advice: Aspect executed after service2()returns");

}

}

### YourService.java

**package** com.example.service;

**import** org.springframework.stereotype.Service; @Service

**public class** YourService {

**public void** yourmethod() { System.***out***.println("Your method executed.");

}

}

In pom.xml only change

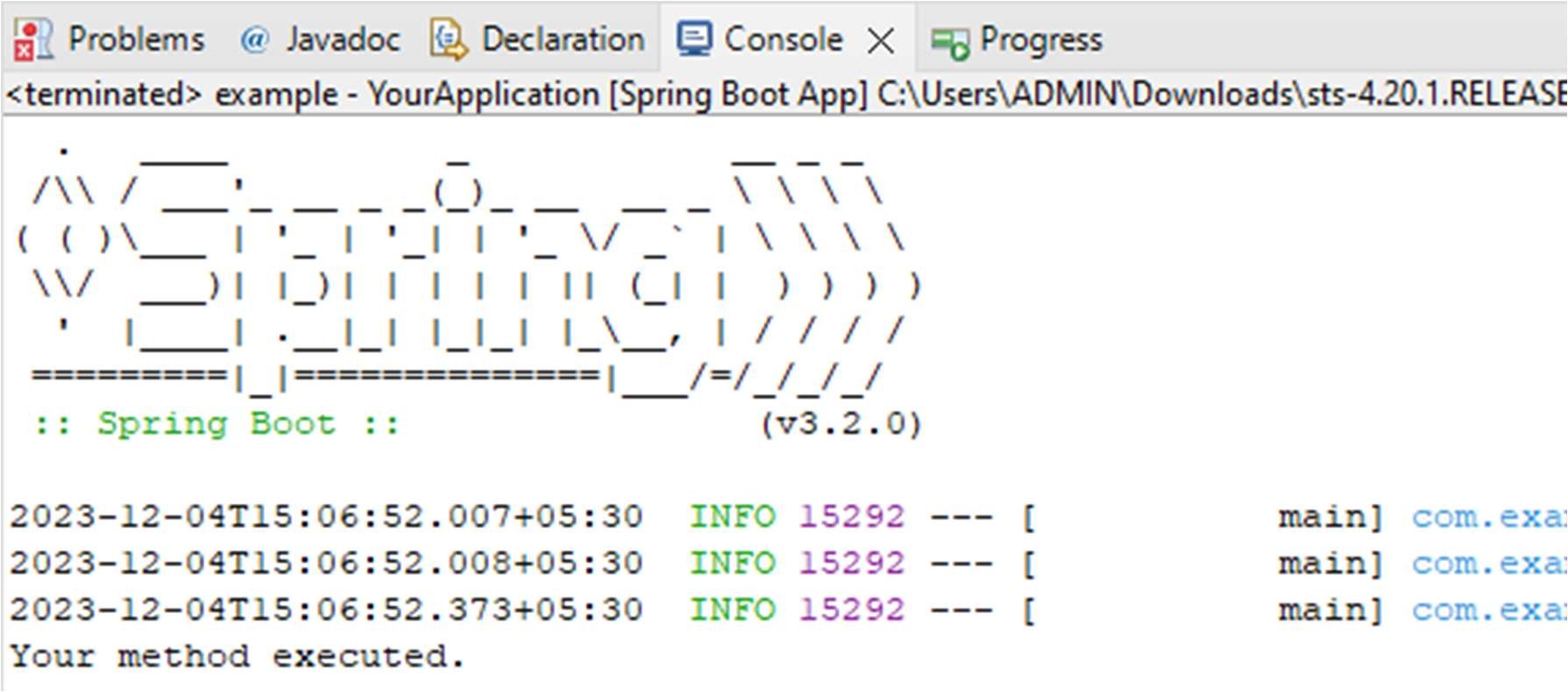
<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.0</version>

</parent> Output:



### Practical 8.6

**Aim:** Write a program to demonstrate Spring AOP – pointcuts.

Code: Application.java:

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.ConfigurableApplicationContext; import com.example.service.serv1;

@SpringBootApplication public class application {

public static void main(String[] args) {

ConfigurableApplicationContext context=SpringApplication.*run*(application.class, args);

serv1 mservice=context.getBean(serv1.class); mservice.method1();

mservice.method2();

}

}

# Aspect.java:

package com.example.aspect;

import org.aspectj.lang.annotation.Aspect; import org.aspectj.lang.annotation.Before; import org.aspectj.lang.annotation.Pointcut;

import org.springframework.stereotype.Component; @Aspect

@Component

public class myaspect {

@Pointcut("execution(\* com.example.service.serv1.method1())") public void pointcutMethod1() {

}

@Pointcut("execution(\* com.example.service.serv1.method2())")

public void pointcutMethod2() {

}

@Before("pointcutMethod1()") public void beforeMethod1() {

System.*out*.println("Before advice for method1");

}

@Before("pointcutMethod2()") public void beforeMethod2() {

System.*out*.println("Before advice for method2");

}

}

# Service.java:

package com.example.service;

import org.springframework.stereotype.Service; @Service

public class serv1 {

public void method1() { System.*out*.println("Executing method1");

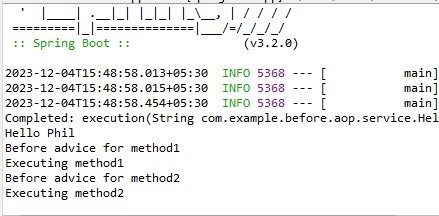
}

public void method2() { System.*out*.println("Executing method2");

}

}

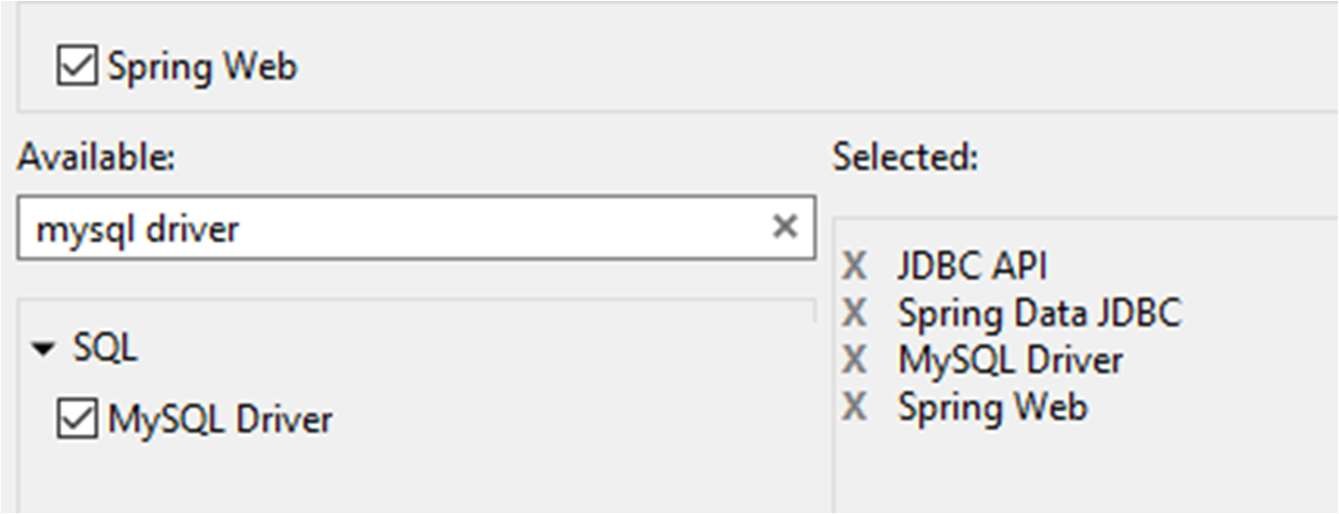
Output:

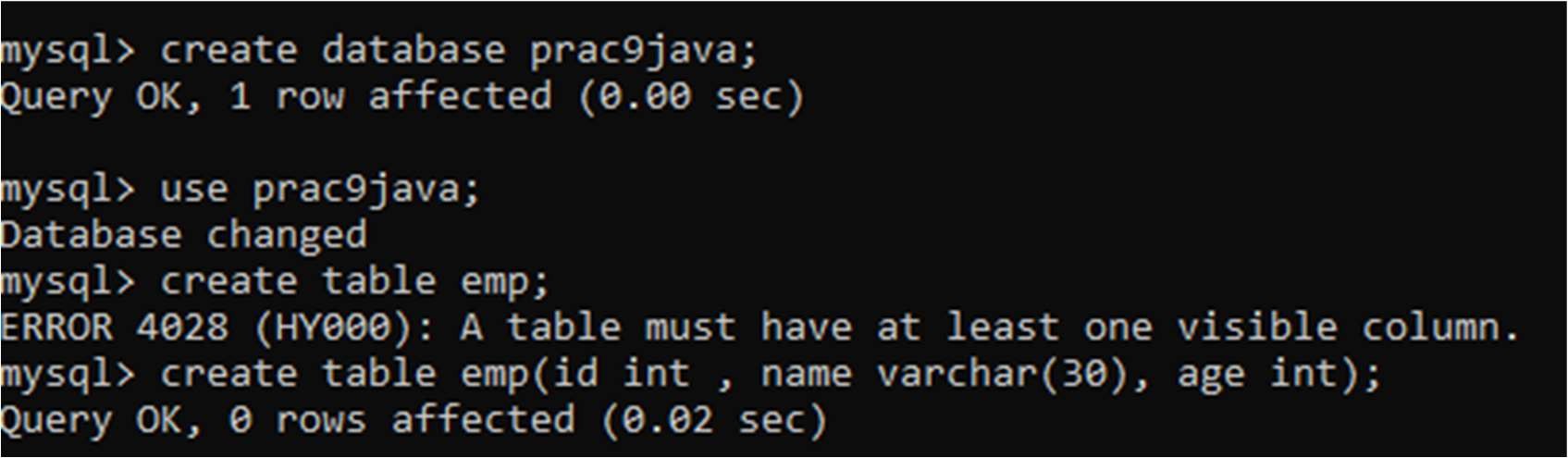


### Practical 9

**Practical 9.1**

**Aim:** Write a program to insert update and delete records from the given table Code:







Appconfig.java

Appconfig.java:

**package** com.example.demo;

**import** javax.sql.DataSource;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.jdbc.datasource.DriverManagerDataSource;

@Configuration

**public class** Appconfig {

@Bean

DataSource dataSource() {

DriverManagerDataSource dataSource = **new** DriverManagerDataSource(); dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver"); dataSource.setUrl("jdbc:mysql://localhost:3306/prac9java"); dataSource.setUsername("root");

dataSource.setPassword("root");

**return** dataSource;

}

@Bean

JdbcTemplate jdbcTemplate(DataSource dataSource) {

**return new** JdbcTemplate(dataSource);

}

}



Operationjdbc.java

Operationjdbc.java

**package** com.example.demo;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.stereotype.Service;

@Service

**public class** Operationjdbc {

@Autowired

**private** JdbcTemplate jdbcTemplate;

**public void** insertRecord(String name, **int** age ,**int** id) {

String sql = "INSERT INTO emp (name, age, id) VALUES (?, ?, ?)"; jdbcTemplate.update(sql, name, age,id);

}

**public void** updateRecord( String name, **int** age ,**int** id) {

String sql = "UPDATE emp SET name = ?, age = ? where id=?"; jdbcTemplate.update(sql, name, age,id);

}

**public void** deleteRecord(**int** id) {

String sql = "DELETE FROM emp WHERE id = ?"; jdbcTemplate.update(sql, id);

}

}



Application.java

Application.java

**package** com.example.demo;

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.Bean;

@SpringBootApplication

**public class** Application {

**public static void** main(String[] args) { SpringApplication.*run*(Application.**class**, args);

}

@Bean

**public** CommandLineRunner demo(Operationjdbc recordService) {

**return** args -> {

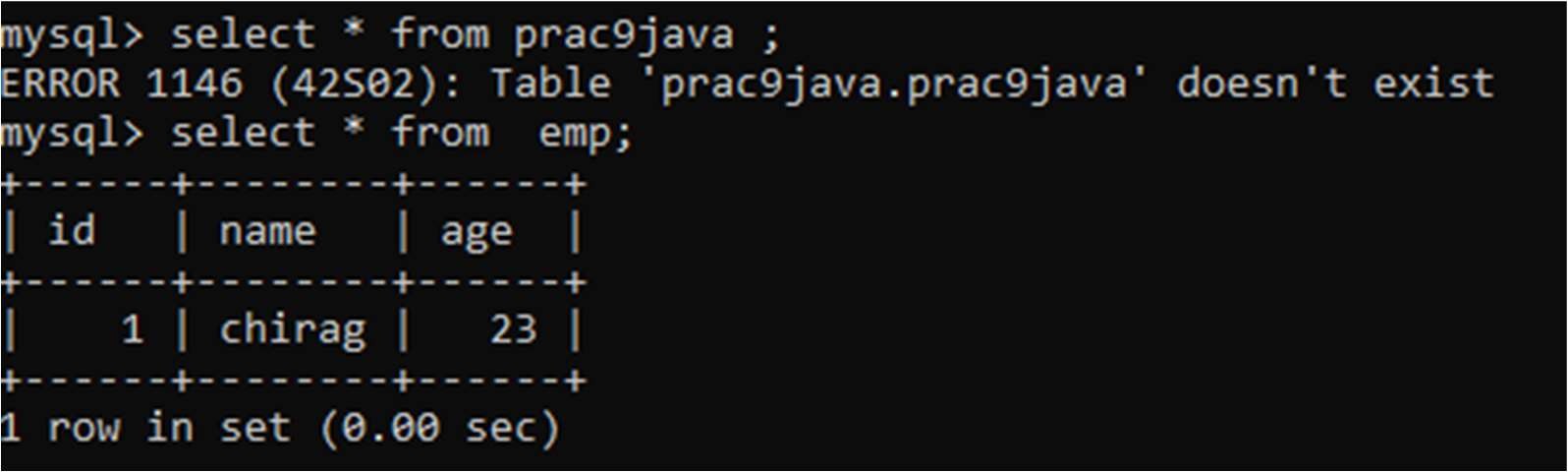
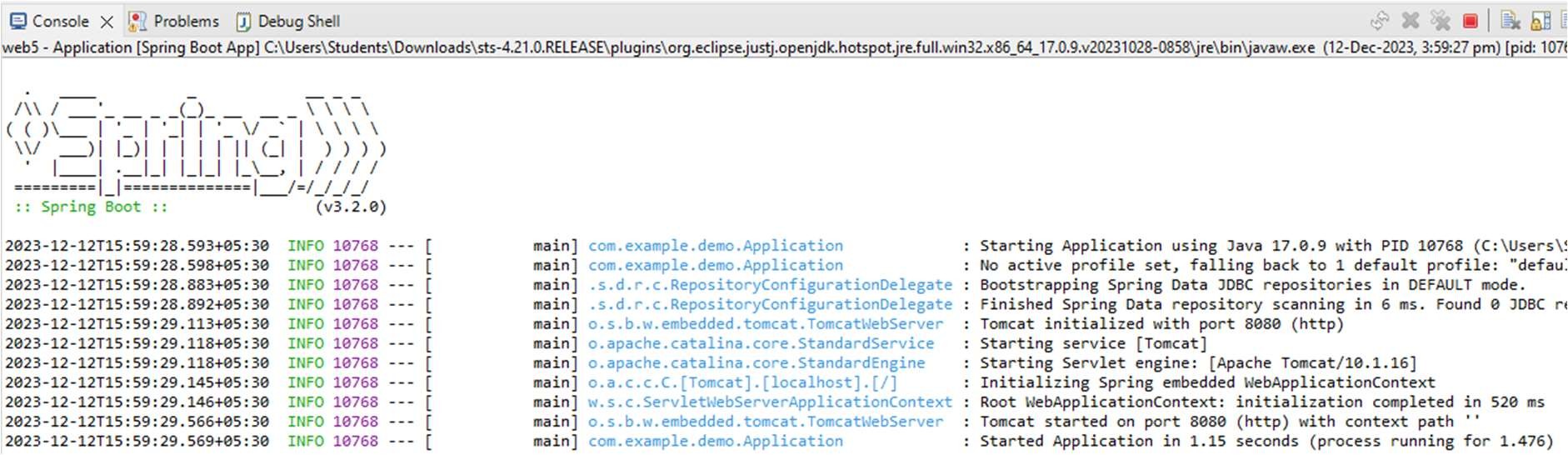
recordService.insertRecord("chirag",23,1);

};

}

}

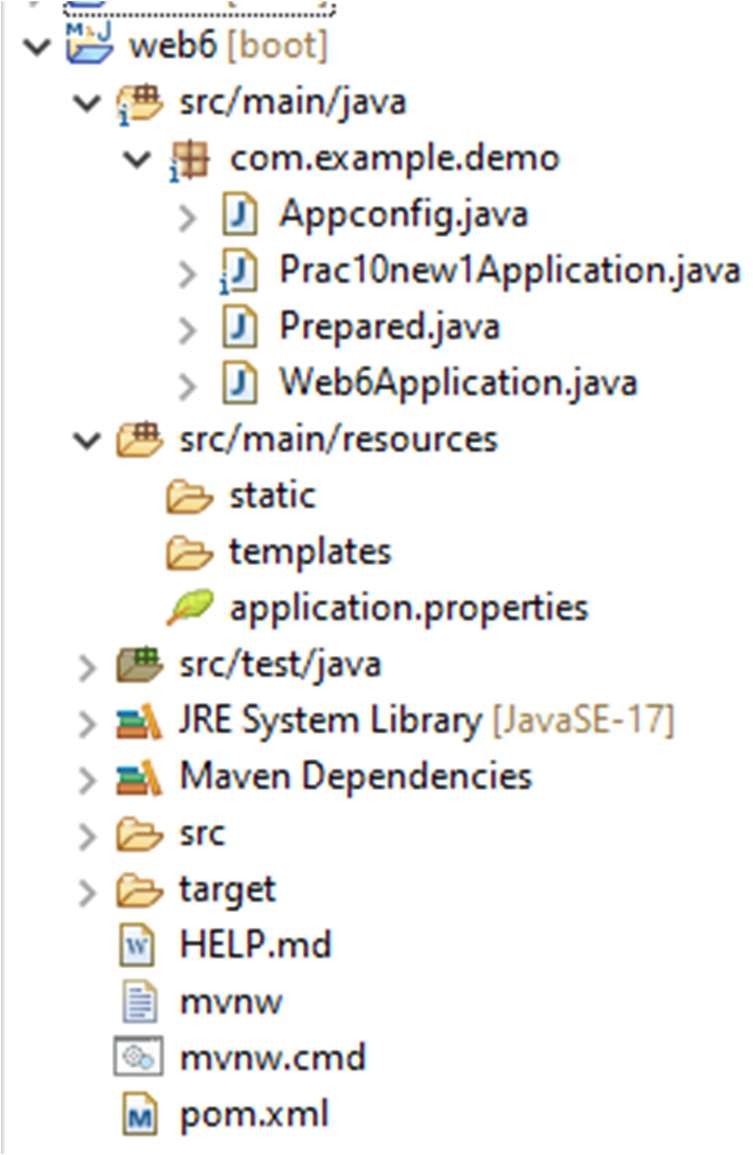
Output:



### Practical 9.2

**Aim:** Write a program to demonstrate Prepared Statement in Spring JDBC Template

Code:



Appconfig.java

Appconfig.java:

package com.example.demo;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration; import javax.sql.DataSource;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

@Configuration

public class Appconfig {

@Bean

DataSource dataSource() {

DriverManagerDataSource dataSource = new DriverManagerDataSource(); dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver"); dataSource.setUrl("jdbc:mysql://localhost:3306/java9");

dataSource.setUsername("root"); dataSource.setPassword("root"); return dataSource;

}

@Bean

JdbcTemplate jdbcTemplate(DataSource dataSource) { return new JdbcTemplate(dataSource);

}

}

Prac10new1Application.java

Prac10new1Applicaton.java:

package com.example.demo;

import org.springframework.boot.SpringApplication; import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class Prac10new1Application {

public static void main(String[] args) { SpringApplication.run(Prac10new1Application.class, args);

}

@Bean

public CommandLineRunner demo(Prepared recordService) { return args -> {

// Insert a record recordService.insertRecord("chirag", 25,1);

};

}

}

Prepared.java

**package** com.example.demo;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.jdbc.core.PreparedStatementCreator;

**import** org.springframework.stereotype.Service;

**import** java.sql.Connection; **import** java.sql.PreparedStatement; **import** java.sql.SQLException; @Service

**public class** Prepared { @Autowired

**private** JdbcTemplate jdbcTemplate;

**public void** insertRecord(String name, **int** age, **int** id) {

**final** String sql = "INSERT INTO emp (name, age, id) VALUES (?, ?, ?)"; jdbcTemplate.update(

**new** PreparedStatementCreator() { @Override

**public** PreparedStatement createPreparedStatement(Connection connection) **throws** SQLException {

PreparedStatement ps = connection.prepareStatement(sql); ps.setString(1, name);

ps.setInt(2, age); ps.setInt(3, id); **return** ps;

}

}

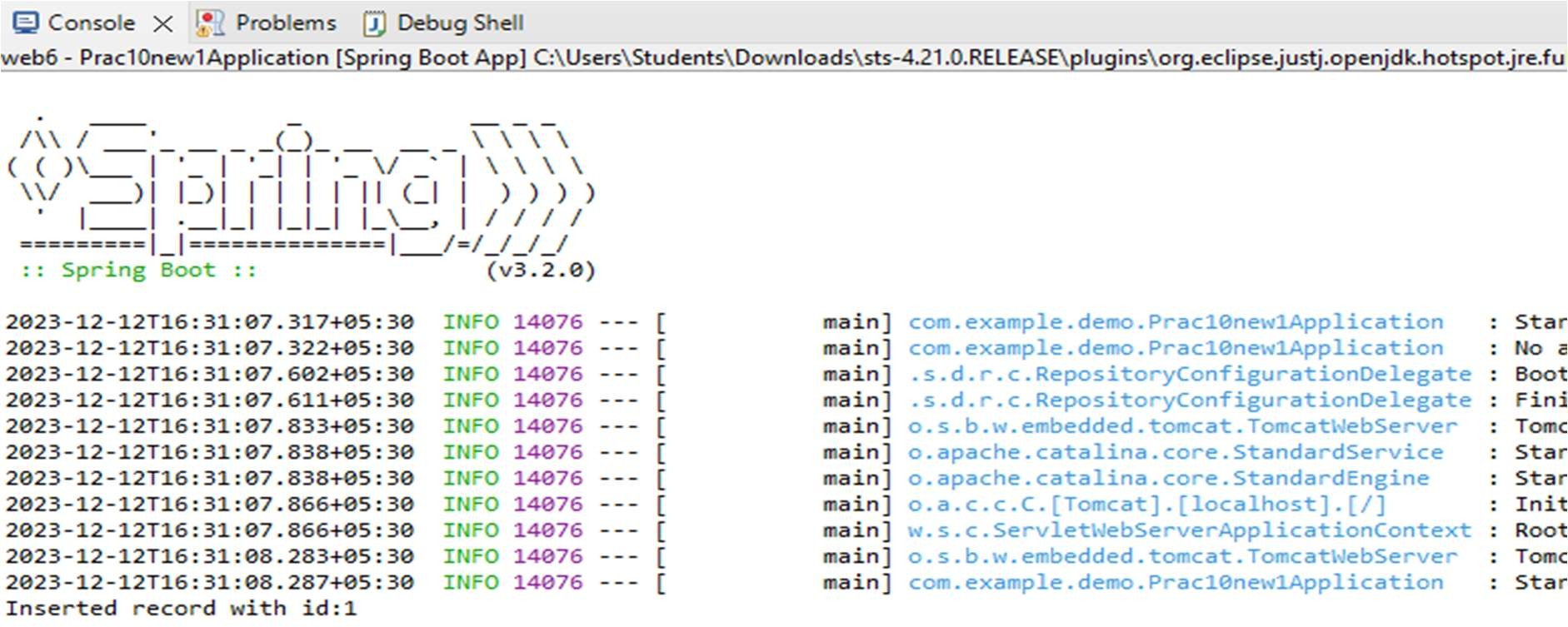
);

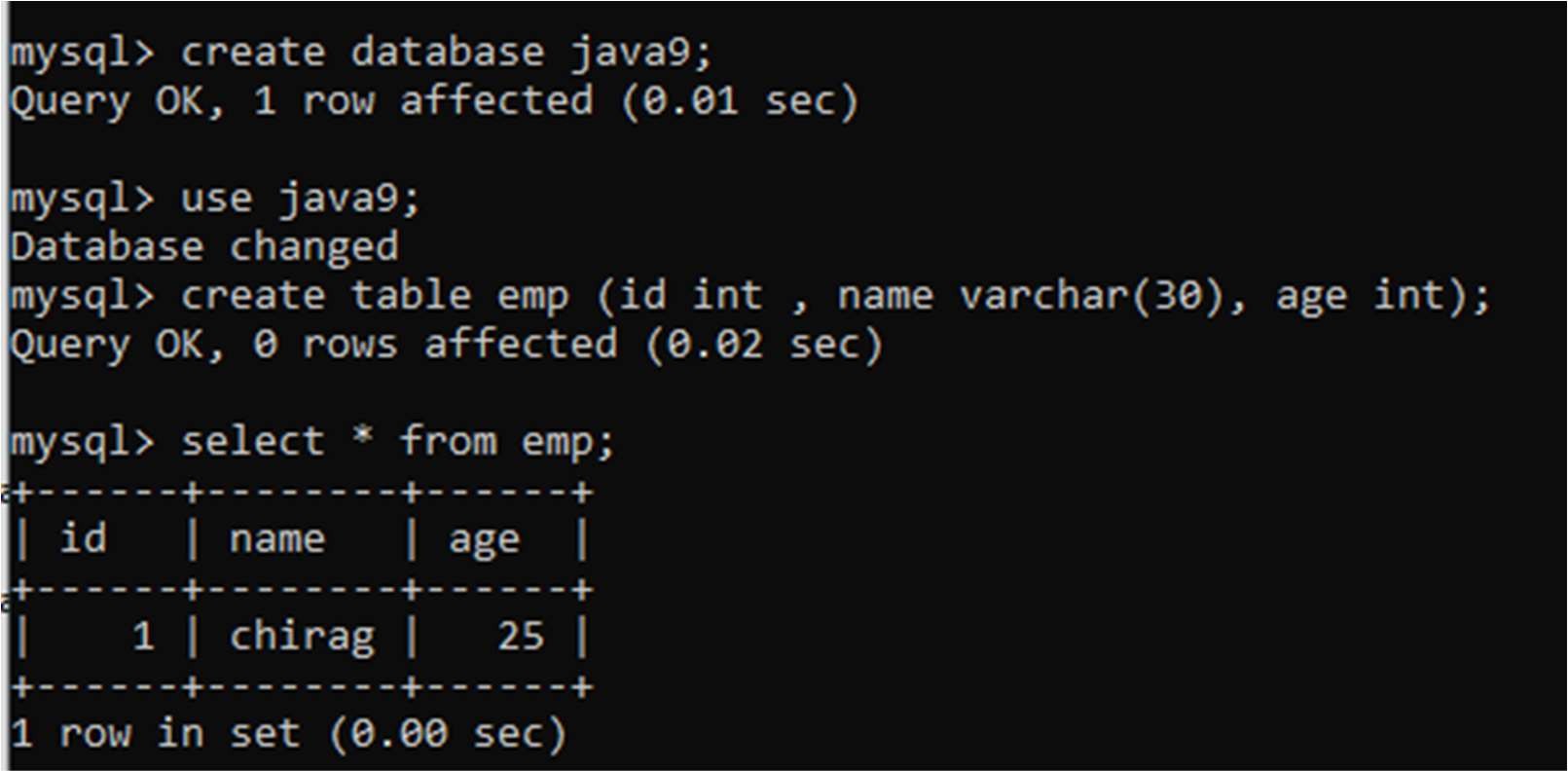
System.***out***.println("Inserted record with id:"+ id);

}

}

Output:

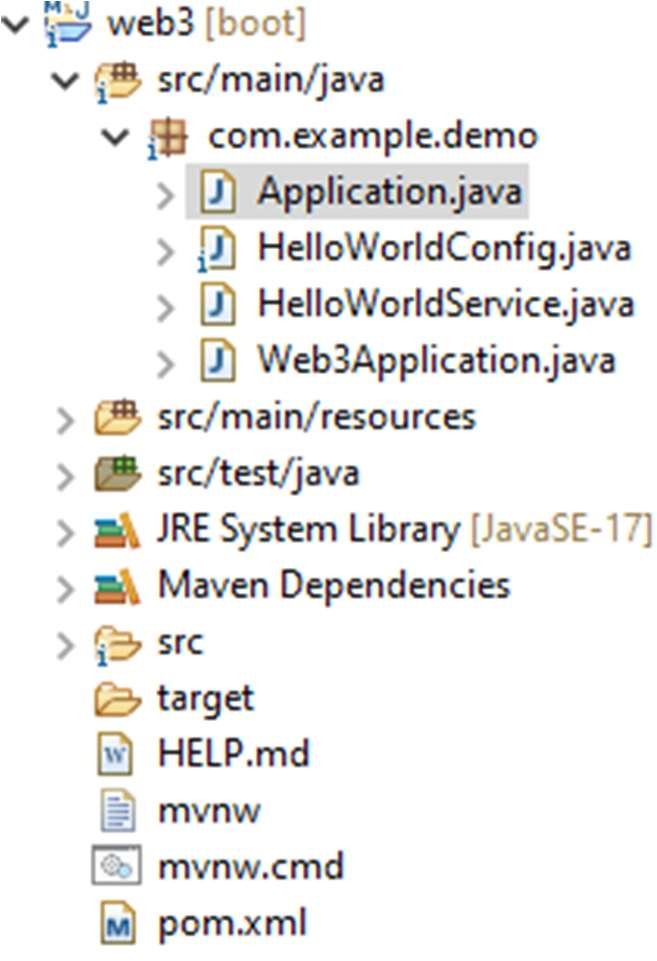




### Practical No.7

**Practical 7.1**

**Aim:** Write a program to print “Hello World” using spring framework. Code:



Application.java

Application.java:

**package** com.example.demo;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

@SpringBootApplication

**public class** Application {

**public static void** main(String[] args) {

**try** (AnnotationConfigApplicationContext context = **new**

AnnotationConfigApplicationContext(HelloWorldConfig.**class**)) {

HelloWorldService helloWorldService = context.getBean(HelloWorldService.**class**); helloWorldService.sayhello();

}

}

}



HelloWorldConfig.java

HelloWorldConfig.java:

**package** com.example.demo;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration; @Configuration

**public class** HelloWorldConfig { @Bean

**public** HelloWorldService helloWorldService() {

**return new** HelloWorldService();

}

}



HelloWorldService.java

HelloWorldService.java:

**package** com.example.demo;

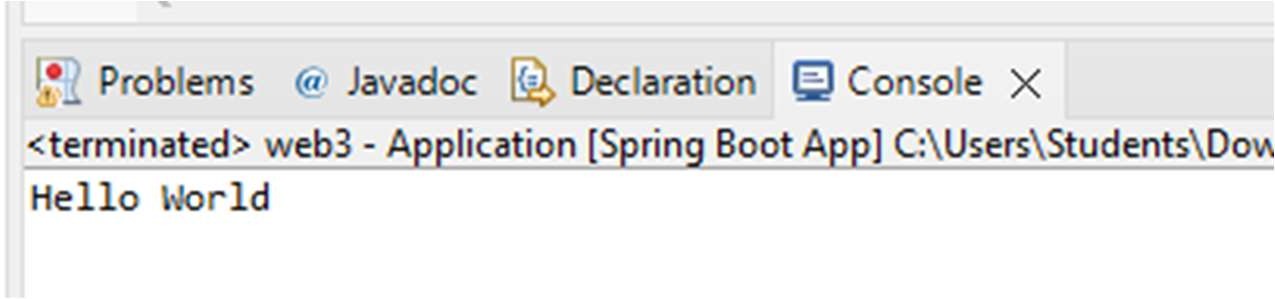
**public class** HelloWorldService {

**public void** sayhello() { System.***out***.println("Hello World");

}

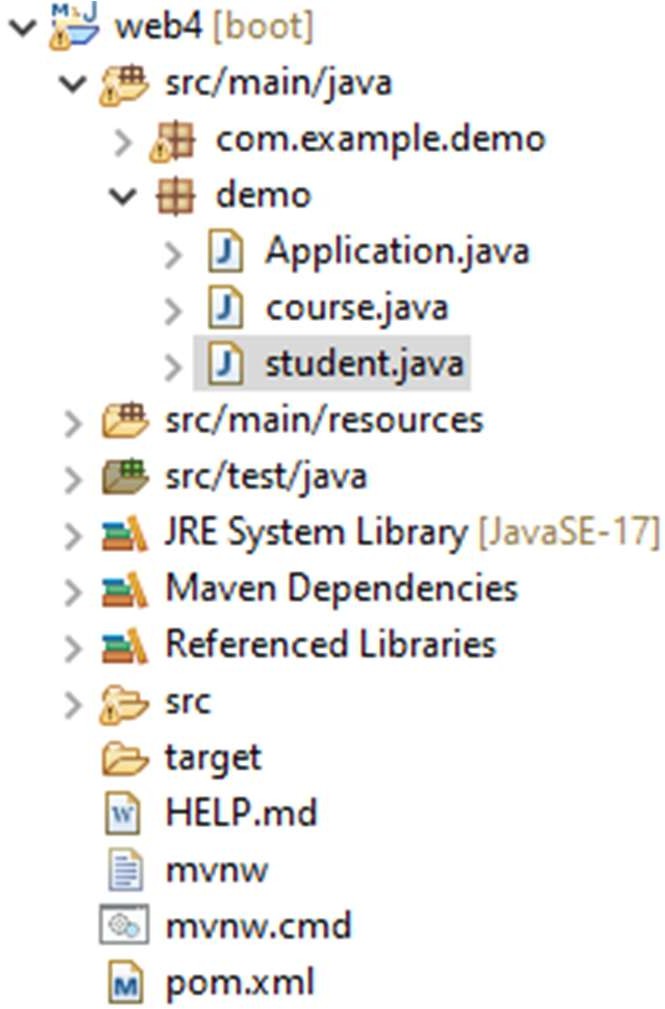
}

Output:



**Practical 7.2**

# **Aim:** Write a program to demonstrate dependency injection via setter method Code:



Application.java

# Application.java:

**package** demo;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.context.ConfigurableApplicationContext; @SpringBootApplication

**public class** Application {

**public static void** main(String[] args) { ConfigurableApplicationContext context=SpringApplication.*run*(Application.**class**, args);

//System.out.print(&quot;hello world&quot;); student s= context.getBean(student.**class**); s.show();

s.compile();

}

}



course.java

# Course.java

**package** demo;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.context.ConfigurableApplicationContext; @SpringBootApplication

**public class** Application {

**public static void** main(String[] args) { ConfigurableApplicationContext context=SpringApplication.*run*(Application.**class**, args);

//System.out.print(&quot;hello world&quot;); student s= context.getBean(student.**class**); s.show();

s.compile();

}

}



student.java

# Student.java

**package** demo;

**import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.beans.factory.annotation.Qualifier; **import** org.springframework.stereotype.Component; @Component

**public class** student { **private int** rollno; **private** String name; **private int** marks; @Autowired @Qualifier("course1") **private** course course; **public** student() { **super**();

System.***out***.println("obj created");

}

**public** course getC() {

**return** course;

}

**public void** setC(course course) {

**this**.course = course;

}

**public int** getRollno() {

**return** rollno;

}

**public void** setRollno(**int** rollno) {

**this**.rollno = rollno;

}

**public** String getName() {

**return** name;

}

**public void** setName(String name) {

**this**.name = name;

}

**public int** getMarks() {

**return** marks;

}

**public void** setMarks(**int** marks) {

**this**.marks = marks;

}

**public void** show() { System.***out***.print("in show");

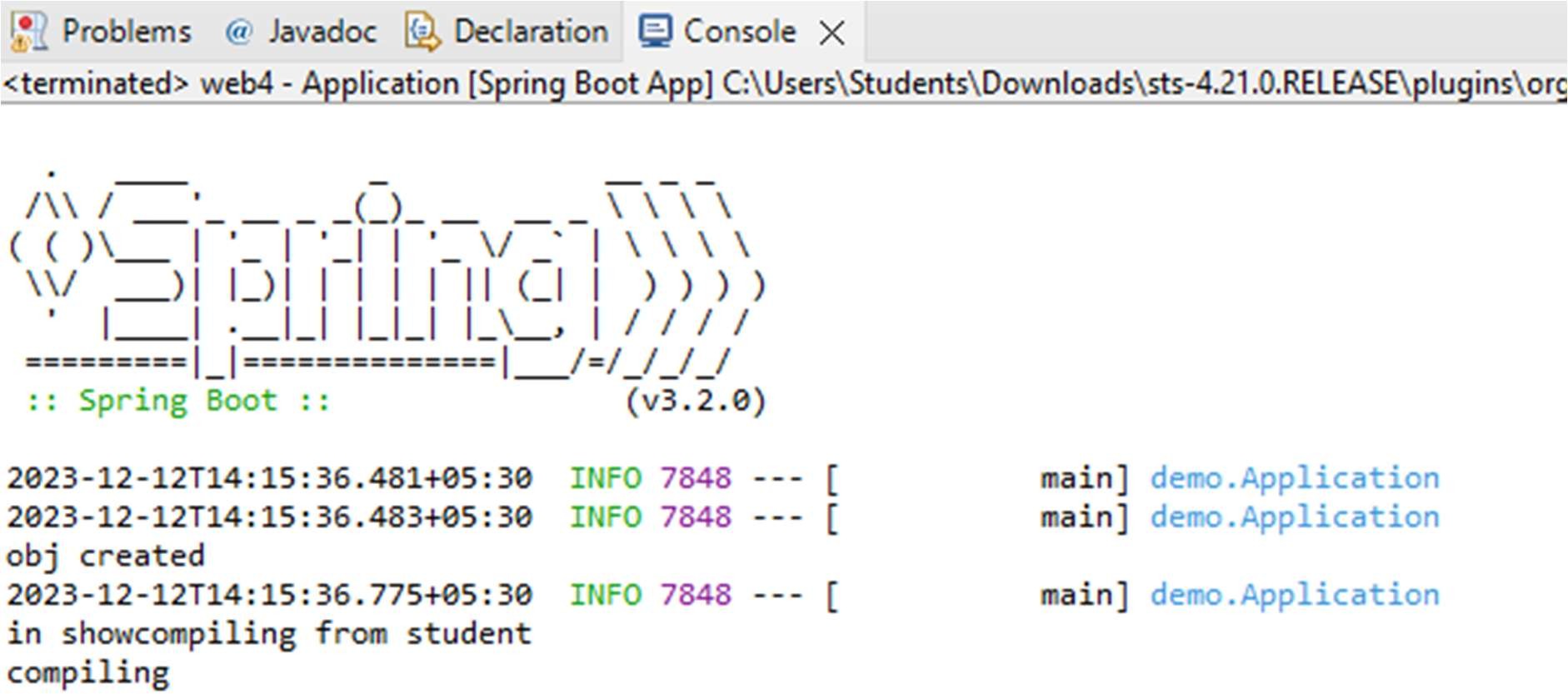
}

**public void** compile() { System.***out***.println("compiling from student"); course.compile();

}

}

## Output:



**Practical 7.3**

# **Aim:** Write a program to demonstrate dependency injection via Constructor Code:



Hello1Application.java

# Hello1Application.java:

**package** com.example.demo;

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

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

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

@SpringBootApplication @ComponentScan(basePackages = "com.example.demo") **public class** Hello1Application {

**public static void** main(String[] args) { SpringApplication.*run*(Hello1Application.**class**, args);

}

@Bean

**public** CommandLineRunner demo(Car car) {

**return** args -> {

// Start the Car

car.setModel("Honda"); Driver dr=**new** Driver(); dr.setName("chirag");

car.setDriver(dr); car.start();

};

}

}

@RestController

**class** HelloController {

**private final** Car car;

**public** HelloController(Car car) {

**this**.car = car;

}

@GetMapping("/hello")

**public** String hello() { car.start();

**return** car.getModel();

}

}



Driver.java

# Driver.java:

**package** com.example.demo;

**import** org.springframework.stereotype.Component; @Component

**public class** Driver { **private** String name; **public** String getName() {

**return** name;

}

**public void** setName(String name) {

**this**.name = name;

}

}



Car.java

# Car.java:

**package** com.example.demo;

**import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Component; @Component

**public class** Car { **private** String model; **private** Driver driver;

**public** String getModel() {

**return** model;

}

**public void** setModel(String model) {

**this**.model = model;

}

**public** Driver getDriver() {

**return** driver;

}

@Autowired

**public void** setDriver(Driver driver) {

**this**.driver = driver;

}

**public void** start() {

System.***out***.println("Car model: " + model); System.***out***.println("Driver: " + driver.getName()); System.***out***.println("Car is starting...");

}

}

## Output:

