COGNIZANT

Digital Nurture 4.0

Deep Skilling - Java FSE

WEEK-2 HANDS ON

By Kaviya P

**JUnit\_Spring Test exercises**

Exercise 1: Basic Unit Test for a Service Method

Task: Write a unit test for a service method that adds two numbers.

**CalculatorService.java**

**package** com.javafse.springBootTesting;

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

@Service

**public** **class** CalculatorService {

**public** **int** add(**int** a, **int** b) {

**return** a + b;

}

}

**CalculatorServiceTest.java**

**package** com.javafse.springBootTesting;

**import** **static** org.junit.Assert.*assertEquals*;

**import** org.junit.jupiter.api.Test;

**public** **class** CalculatorServiceTest {

**private** CalculatorService service = **new** CalculatorService();

@Test

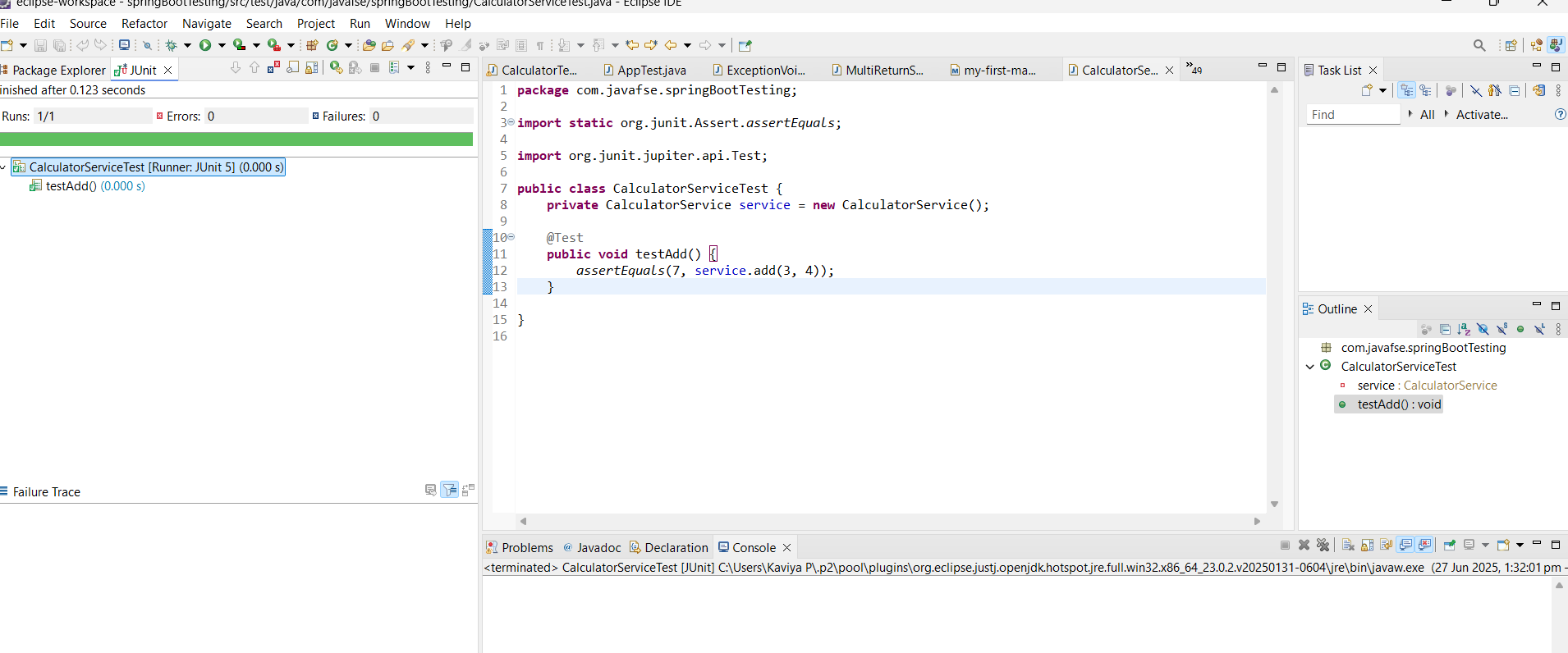
**public** **void** testAdd() {

*assertEquals*(7, service.add(3, 4));

}

}

OUTPUT:



Exercise 2: Mocking a Repository in a Service Test

Task: Test a service that uses a repository to fetch data.

**User.java**

**package** com.javafse.springBootTesting;

**import** jakarta.persistence.\*;

@Entity

**public** **class** User {

@Id

**private** Long id;

**private** String name;

// Getters & Setters

**public** Long getId() { **return** id; }

**public** **void** setId(Long id) { **this**.id = id; }

**public** String getName() { **return** name; }

**public** **void** setName(String name) { **this**.name = name; }

}

**UserRepository.java**

**package** com.javafse.springBootTesting;

**import** org.springframework.data.jpa.repository.JpaRepository;

**public** **interface** UserRepository **extends** JpaRepository<User,Long>{

}

**UserService.java**

**package** com.javafse.springBootTesting;

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

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

@Service

**public** **class** UserService {

@Autowired

**public** UserRepository userRepository;

**public** User getUserById(Long id) {

**return** userRepository.findById(id).orElse(**null**);

}

**public** User saveUser(User user) {

**return** userRepository.save(user);

}

}

**UserServiceTest.java**

**package** com.javafse.springBootTesting;

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** **static** org.mockito.Mockito.\*;

**import** java.util.Optional;

**import** org.junit.jupiter.api.Test;

**import** org.mockito.InjectMocks;

**import** org.mockito.Mock;

**import** org.mockito.MockitoAnnotations;

**public** **class** UserServiceTest {

@Mock

**private** UserRepository userRepository;

@InjectMocks

**private** UserService userService;

**public** UserServiceTest() {

MockitoAnnotations.*openMocks*(**this**);

}

@Test

**public** **void** testGetUserById() {

User user = **new** User();

user.setId(1L);

user.setName("Alice");

*when*(userRepository.findById(1L)).thenReturn(Optional.*of*(user));

User result = userService.getUserById(1L);

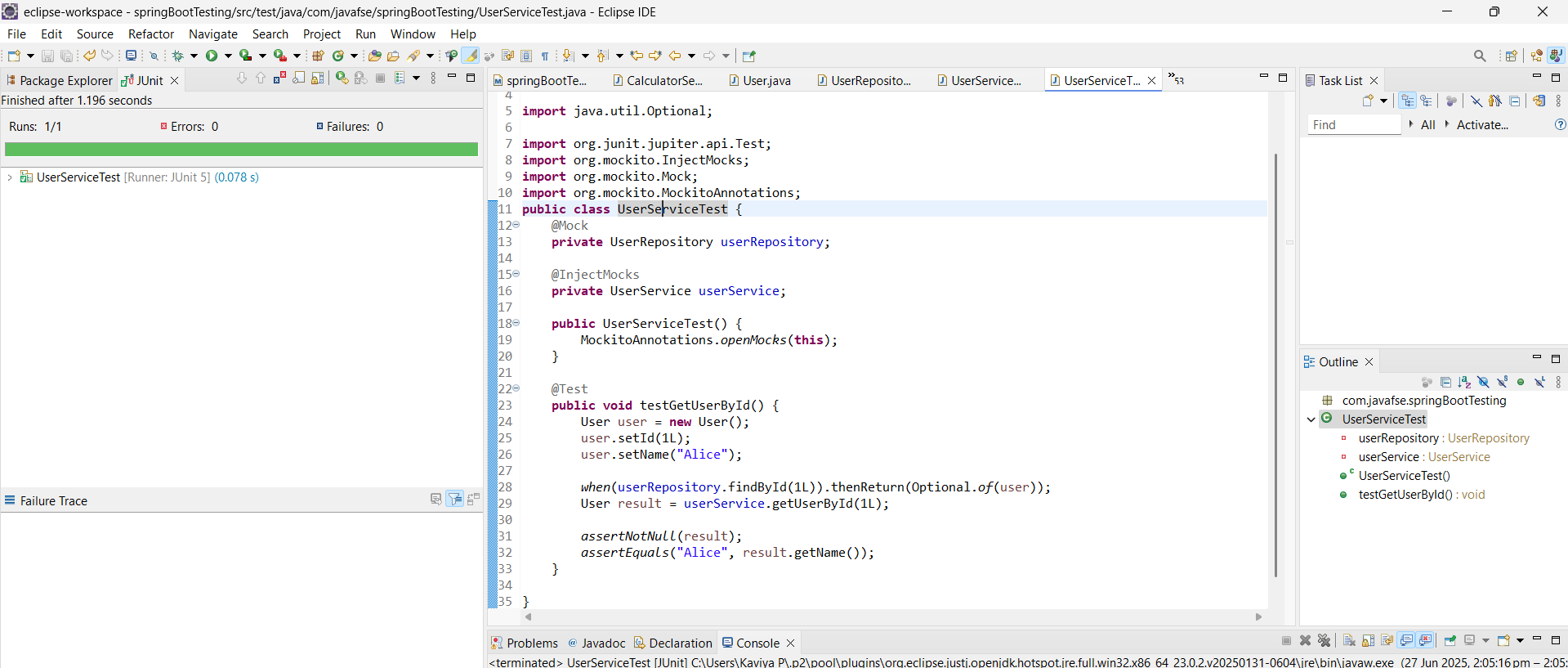
*assertNotNull*(result);

*assertEquals*("Alice", result.getName());

}

}

OUTPUT:



Exercise 3: Testing a REST Controller with MockMvc

Task: Test a controller endpoint that returns a user.

**UserController.java**

**package** com.kaviyaP.SpringBootFirst.controller;

**import** service.UserService;

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

**import** org.springframework.http.ResponseEntity;

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

**import** com.kaviyaP.SpringBootFirst.model.User;

@RestController

@RequestMapping("/users")

**public** **class** UserController {

@Autowired

**private** UserService userService;

@GetMapping("/{id}")

**public** ResponseEntity<User> getUser(@PathVariable Long id) {

**return** ResponseEntity.*ok*(userService.getUserById(id));

}

}

**User.java**

**package** com.kaviyaP.SpringBootFirst.model;

**public** **class** User {

**private** Long id;

**private** String name;

**public** User() {}

**public** User(Long id, String name) {

**this**.id = id;

**this**.name = name;

}

// Getters and Setters

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

}

**UserService.java**

**package** service;

**import** com.kaviyaP.SpringBootFirst.model.User;

**public** **interface** UserService {

User getUserById(Long id);

}

**UserServiceImpl.java**

**package** service;

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

**import** com.kaviyaP.SpringBootFirst.model.User;

@Service

**public** **class** UserServiceImpl **implements** UserService{

@Override

**public** User getUserById(Long id) {

**return** **new** User(id, "John Doe");

}

}

**UserControllerTest.java**

**package** com.kaviyaP.SpringBootFirst;

**import** **static** org.mockito.Mockito.*when*;

**import** org.junit.jupiter.api.Test;

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

**import** org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;

**import** org.springframework.boot.test.mock.mockito.~~MockBean~~;

**import** org.springframework.test.web.servlet.MockMvc;

**import** com.kaviyaP.SpringBootFirst.controller.UserController;

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** service.UserService;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

**import** com.fasterxml.jackson.databind.ObjectMapper;

@WebMvcTest(UserController.**class**)

**public** **class** UserControllerTest {

@Autowired

**private** MockMvc mockMvc;

@~~MockBean~~

**private** UserService userService;

@Test

**public** **void** testGetUser() **throws** Exception {

User mockUser = **new** User(1L, "John Doe");

*when*(userService.getUserById(1L)).thenReturn(mockUser);

mockMvc.perform(*get*("/users/1"))

.andExpect(*status*().isOk())

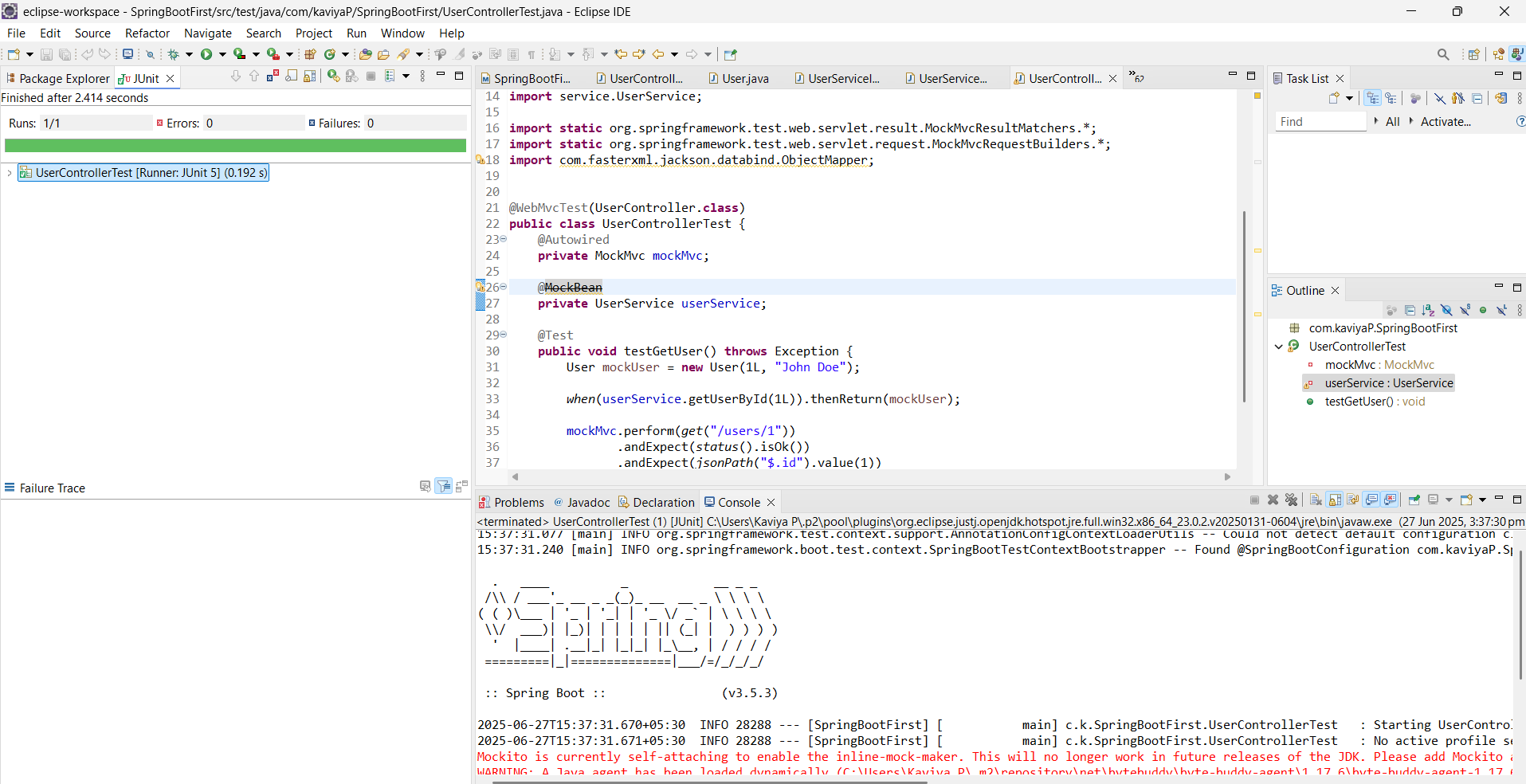
.andExpect(*jsonPath*("$.id").value(1))

.andExpect(*jsonPath*("$.name").value("John Doe"));

}

}

OUTPUT



Exercise 4: Integration Test with Spring Boot Task: Write an integration test that tests the full flow from controller to database. Test: Write code for this.

**UserController.java**

**package** com.kaviyaP.SpringBootFirst.controller;

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

**import** org.springframework.http.ResponseEntity;

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

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.service.UserService;

@RestController

@RequestMapping("/users")

**public** **class** UserController {

@Autowired

**private** UserService userService;

@GetMapping("/{id}")

**public** ResponseEntity<User> getUser(@PathVariable Long id) {

**return** ResponseEntity.*ok*(userService.getUserById(id));

}

}

**User.java**

**package** com.kaviyaP.SpringBootFirst.model;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

@Entity

@Table(name = "users")

**public** **class** User {

@Id

**private** Long id;

**private** String name;

**public** User() {}

**public** User(Long id, String name) {

**this**.id = id;

**this**.name = name;

}

// Getters and Setters

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

}

**UserRepository.java**

**package** com.kaviyaP.SpringBootFirst.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.kaviyaP.SpringBootFirst.model.User;

**public** **interface** UserRepository **extends** JpaRepository<User, Long> {

}

**UserService.java**

**package** com.kaviyaP.SpringBootFirst.service;

**import** com.kaviyaP.SpringBootFirst.model.User;

**public** **interface** UserService {

User getUserById(Long id);

}

**UserServiceImpl.java**

**package** com.kaviyaP.SpringBootFirst.service;

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

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

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.repository.\*;

@Service

**public** **class** UserServiceImpl **implements** UserService{

@Autowired

**private** UserRepository userRepository;

@Override

**public** User getUserById(Long id) {

**return** userRepository.findById(id).orElse(**null**);

}

}

**UserControllerTest.java**

**package** com.kaviyaP.SpringBootFirst;

**import** **static** org.mockito.Mockito.*when*;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Test;

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

**import** org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

**import** org.springframework.boot.test.context.SpringBootTest;

**import** org.springframework.http.MediaType;

**import** org.springframework.test.web.servlet.MockMvc;

**import** **static** org.hamcrest.Matchers.*is*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** com.kaviyaP.SpringBootFirst.controller.UserController;

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.repository.UserRepository;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

**import** com.fasterxml.jackson.databind.ObjectMapper;

@SpringBootTest

@AutoConfigureMockMvc

**public** **class** UserControllerTest {

@Autowired

**private** MockMvc mockMvc;

@Autowired

**private** UserRepository userRepository;

@BeforeEach

**public** **void** setup() {

userRepository.deleteAll();

userRepository.save(**new** User(1L, "John Doe"));

}

@Test

**public** **void** testGetUserFromDatabase() **throws** Exception {

mockMvc.perform(*get*("/users/1").accept(MediaType.***APPLICATION\_JSON***))

.andExpect(*status*().isOk())

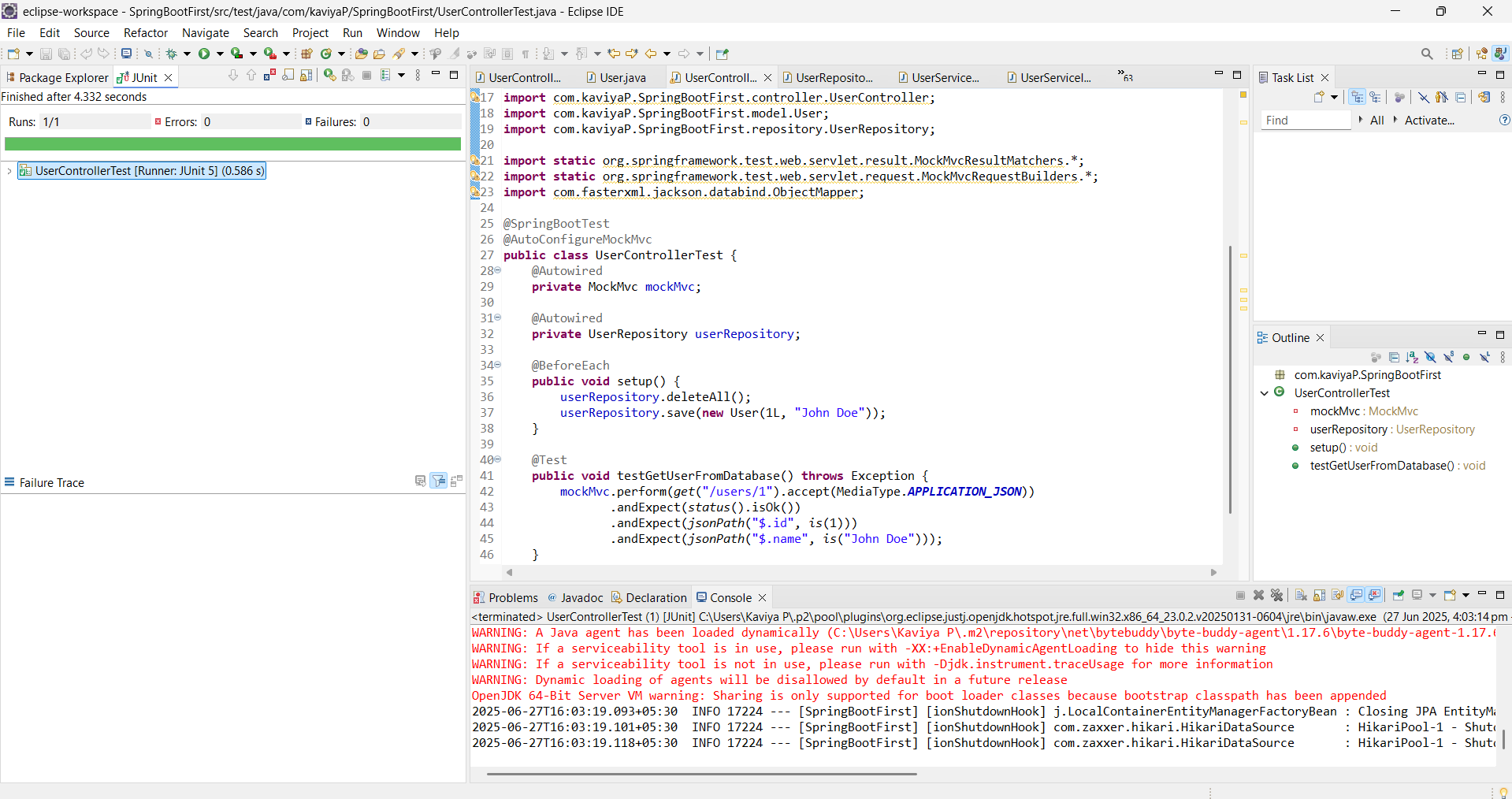
.andExpect(*jsonPath*("$.id", *is*(1)))

.andExpect(*jsonPath*("$.name", *is*("John Doe")));

}

}

OUTPUT



Exercise 5: Test Controller POST Endpoint

Task: Test a POST endpoint that creates a user.

Controller:

@PostMapping

public ResponseEntity<User> createUser(@RequestBody User user) {

return ResponseEntity.ok(userService.saveUser(user));

}

Test:

Write code for this.

**UserController.java**

**package** com.kaviyaP.SpringBootFirst.controller;

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

**import** org.springframework.http.ResponseEntity;

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

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.service.UserService;

@RestController

@RequestMapping("/users")

**public** **class** UserController {

@Autowired

**private** UserService userService;

@PostMapping

**public** ResponseEntity<User> createUser(@RequestBody User user) {

**return** ResponseEntity.*ok*(userService.saveUser(user));

}

}

**User.java**

**package** com.kaviyaP.SpringBootFirst.model;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

@Entity

@Table(name = "users")

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Long id;

**private** String name;

**public** User() {}

**public** User(Long id, String name) {

**this**.id = id;

**this**.name = name;

}

// Getters and Setters

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

}

**UserRepository.java**

**package** com.kaviyaP.SpringBootFirst.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.kaviyaP.SpringBootFirst.model.User;

**public** **interface** UserRepository **extends** JpaRepository<User, Long> {

}

**UserService.java**

**package** com.kaviyaP.SpringBootFirst.service;

**import** com.kaviyaP.SpringBootFirst.model.User;

**public** **interface** UserService {

User saveUser(User user);

}

**UserServiceImpl.java**

**package** com.kaviyaP.SpringBootFirst.service;

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

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

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.repository.\*;

@Service

**public** **class** UserServiceImpl **implements** UserService{

@Autowired

**private** UserRepository userRepository;

@Override

**public** User saveUser(User user) {

**return** userRepository.save(user);

}

}

**UserControllerTest.java**

**package** com.kaviyaP.SpringBootFirst;

**import** **static** org.mockito.Mockito.*when*;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Test;

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

**import** org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

**import** org.springframework.boot.test.context.SpringBootTest;

**import** org.springframework.http.MediaType;

**import** org.springframework.test.web.servlet.MockMvc;

**import** **static** org.hamcrest.Matchers.*is*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** com.kaviyaP.SpringBootFirst.controller.UserController;

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.repository.UserRepository;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

**import** com.fasterxml.jackson.databind.ObjectMapper;

@SpringBootTest

@AutoConfigureMockMvc

**public** **class** UserControllerTest {

@Autowired

**private** MockMvc mockMvc;

@Autowired

**private** UserRepository userRepository;

**private** **final** ObjectMapper objectMapper = **new** ObjectMapper();

@BeforeEach

**public** **void** setup() {

userRepository.deleteAll();

}

@Test

**public** **void** testCreateUser() **throws** Exception {

User user = **new** User();

user.setName("John Doe");

mockMvc.perform(*post*("/users")

.contentType(MediaType.***APPLICATION\_JSON***)

.content(objectMapper.writeValueAsString(user)))

.andExpect(*status*().isOk())

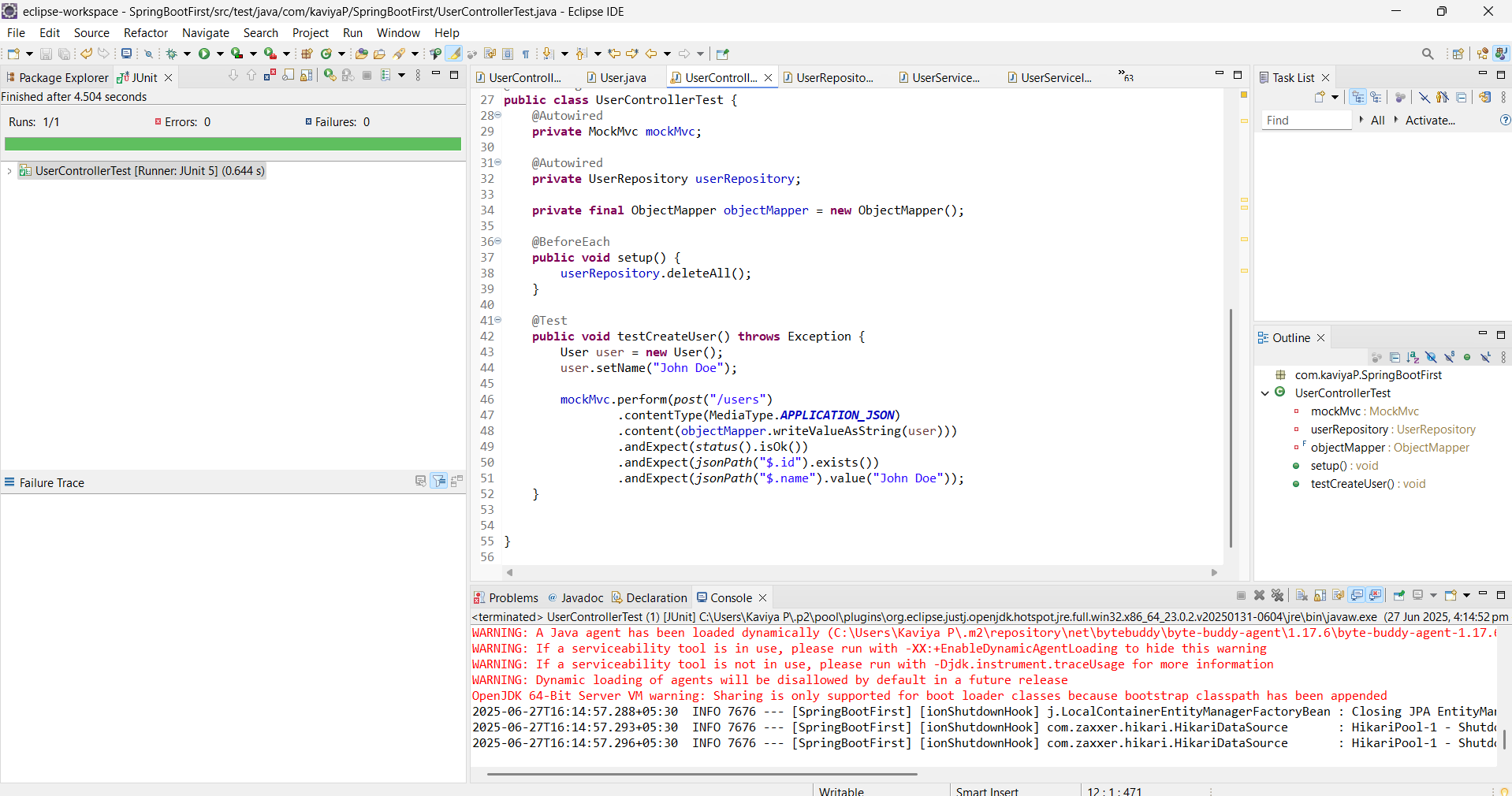
.andExpect(*jsonPath*("$.id").exists())

.andExpect(*jsonPath*("$.name").value("John Doe"));

}

}

OUTPUT



Exercise 6: Test Service Exception Handling

Task: Test how a service handles a missing user.

Test:

Write code for this.

**UserController.java**

**package** com.kaviyaP.SpringBootFirst.controller;

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

**import** org.springframework.http.ResponseEntity;

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

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.service.UserService;

@RestController

@RequestMapping("/users")

**public** **class** UserController {

@Autowired

**private** UserService userService;

@GetMapping("/{id}")

**public** ResponseEntity<User> getUser(@PathVariable Long id) {

**return** ResponseEntity.*ok*(userService.getUserById(id));

}

}

**UserNotFoundException.java**

**package** com.kaviyaP.SpringBootFirst.exception;

**public** **class** UserNotFoundException **extends** RuntimeException{

**public** UserNotFoundException(Long id) {

**super**("User with ID " + id + " not found");

}

}

**User.java**

**package** com.kaviyaP.SpringBootFirst.model;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

@Entity

@Table(name = "users")

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Long id;

**private** String name;

**public** User() {}

**public** User(Long id, String name) {

**this**.id = id;

**this**.name = name;

}

// Getters and setters

**public** Long getId() { **return** id; }

**public** **void** setId(Long id) { **this**.id = id; }

**public** String getName() { **return** name; }

**public** **void** setName(String name) { **this**.name = name; }

}

**UserService.java**

**package** com.kaviyaP.SpringBootFirst.service;

**import** com.kaviyaP.SpringBootFirst.model.User;

**public** **interface** UserService {

User getUserById(Long id);

}

**UserServiceImpl.java**

**package** com.kaviyaP.SpringBootFirst.service;

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

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

**import** com.kaviyaP.SpringBootFirst.exception.UserNotFoundException;

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.repository.\*;

@Service

**public** **class** UserServiceImpl **implements** UserService{

@Autowired

**private** UserRepository userRepository;

**public** **void** setUserRepository(UserRepository userRepository) {

**this**.userRepository = userRepository;

}

@Override

**public** User getUserById(Long id) {

**return** userRepository.findById(id)

.orElseThrow(() -> **new** UserNotFoundException(id));

}

}

**UserControllerTest.java**

**package** com.kaviyaP.SpringBootFirst;

**import** **static** org.mockito.Mockito.*mock*;

**import** **static** org.mockito.Mockito.*when*;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Test;

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

**import** org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

**import** org.springframework.boot.test.context.SpringBootTest;

**import** org.springframework.http.MediaType;

**import** org.springframework.test.web.servlet.MockMvc;

**import** **static** org.hamcrest.Matchers.*is*;

**import** **static** org.junit.jupiter.api.Assertions.*assertEquals*;

**import** **static** org.junit.jupiter.api.Assertions.*assertThrows*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** java.util.Optional;

**import** com.kaviyaP.SpringBootFirst.controller.UserController;

**import** com.kaviyaP.SpringBootFirst.exception.UserNotFoundException;

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.repository.UserRepository;

**import** com.kaviyaP.SpringBootFirst.service.UserServiceImpl;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

**import** com.fasterxml.jackson.databind.ObjectMapper;

**public** **class** UserControllerTest {

**private** **final** UserRepository userRepository = *mock*(UserRepository.**class**);

**private** **final** UserServiceImpl userService = **new** UserServiceImpl();

{

userService.setUserRepository(userRepository); // You'll need a setter or constructor

}

@Test

**public** **void** testUserNotFound\_ThrowsException() {

Long userId = 99L;

*when*(userRepository.findById(userId)).thenReturn(Optional.*empty*());

UserNotFoundException exception = *assertThrows*(

UserNotFoundException.**class**,

() -> userService.getUserById(userId)

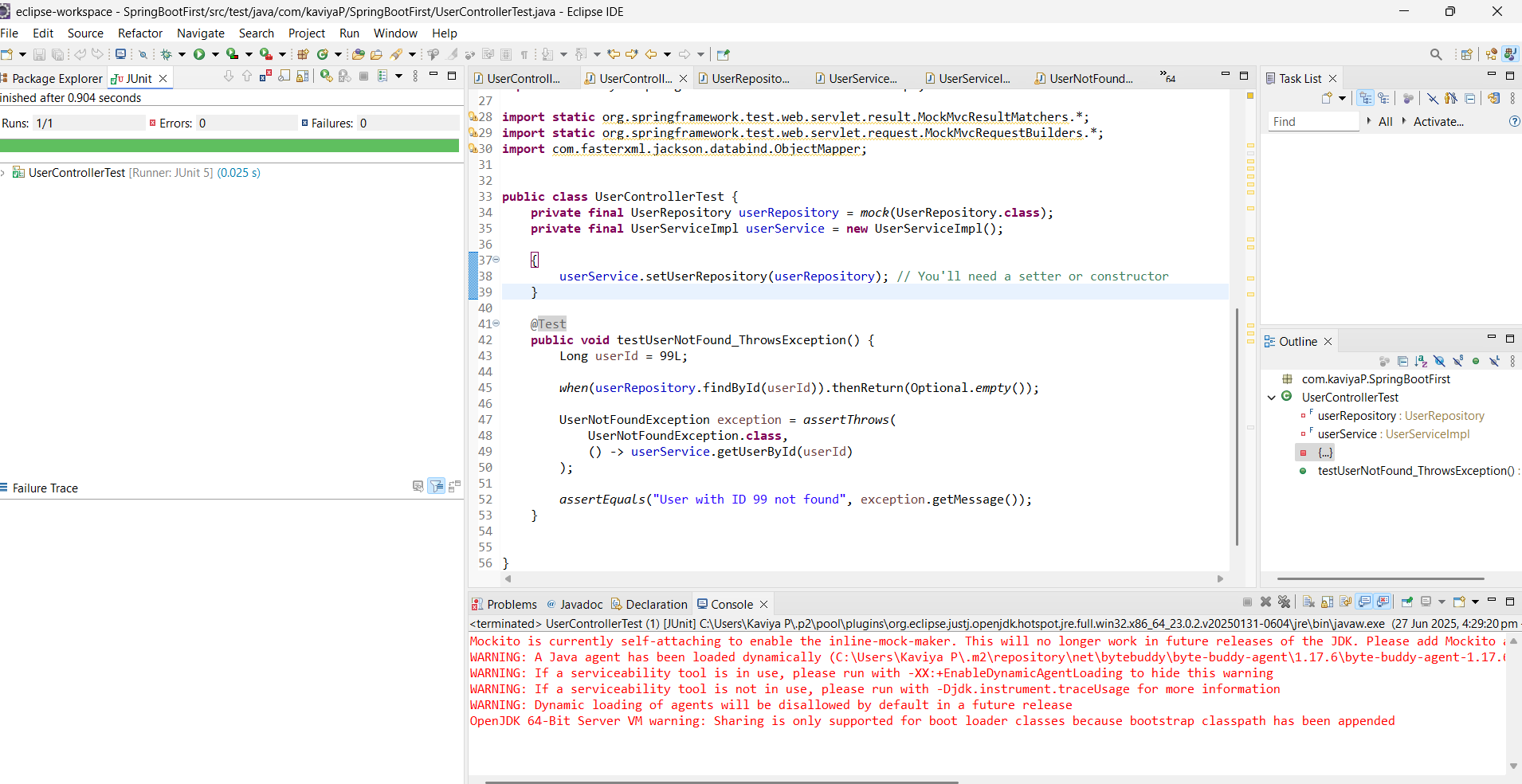
);

*assertEquals*("User with ID 99 not found", exception.getMessage());

}

}

OUTPUT



Exercise 7: Test Custom Repository Query

Task: Add and test a custom query method.

Repository:

public interface UserRepository extends JpaRepository<User, Long> {

List<User> findByName(String name);

}

Test:

Write code for this.

**User.java**

**package** com.kaviyaP.SpringBootFirst.model;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

@Entity

@Table(name = "users")

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Long id;

**private** String name;

**public** User() {}

**public** User(String name) {

**this**.name = name;

}

**public** User(Long id, String name) {

**this**.id = id;

**this**.name = name;

}

// Getters and setters

**public** Long getId() { **return** id; }

**public** **void** setId(Long id) { **this**.id = id; }

**public** String getName() { **return** name; }

**public** **void** setName(String name) { **this**.name = name; }

}

**UserRepository.java**

**package** com.kaviyaP.SpringBootFirst.repository;

**import** java.util.List;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.kaviyaP.SpringBootFirst.model.User;

**public** **interface** UserRepository **extends** JpaRepository<User, Long> {

List<User> findByName(String name);

}

**UserControllerTest.java**

**package** com.kaviyaP.SpringBootFirst;

**import** **static** org.mockito.Mockito.*mock*;

**import** **static** org.mockito.Mockito.*when*;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Test;

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

**import** org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

**import** org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

**import** org.springframework.boot.test.context.SpringBootTest;

**import** org.springframework.http.MediaType;

**import** org.springframework.test.web.servlet.MockMvc;

**import** **static** org.assertj.core.api.Assertions.~~assertThat~~;

**import** **static** org.hamcrest.Matchers.*is*;

**import** **static** org.junit.jupiter.api.Assertions.*assertEquals*;

**import** **static** org.junit.jupiter.api.Assertions.*assertThrows*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** java.util.List;

**import** java.util.Optional;

**import** com.kaviyaP.SpringBootFirst.controller.UserController;

**import** com.kaviyaP.SpringBootFirst.exception.UserNotFoundException;

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.repository.UserRepository;

**import** com.kaviyaP.SpringBootFirst.service.UserServiceImpl;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

**import** com.fasterxml.jackson.databind.ObjectMapper;

@DataJpaTest

**public** **class** UserControllerTest {

@Autowired

**private** UserRepository userRepository;

@Test

**public** **void** testFindByName\_ReturnsMatchingUsers() {

// Arrange

userRepository.save(**new** User("Alice"));

userRepository.save(**new** User("Bob"));

userRepository.save(**new** User("Alice")); // same name

// Act

List<User> users = userRepository.findByName("Alice");

// Assert

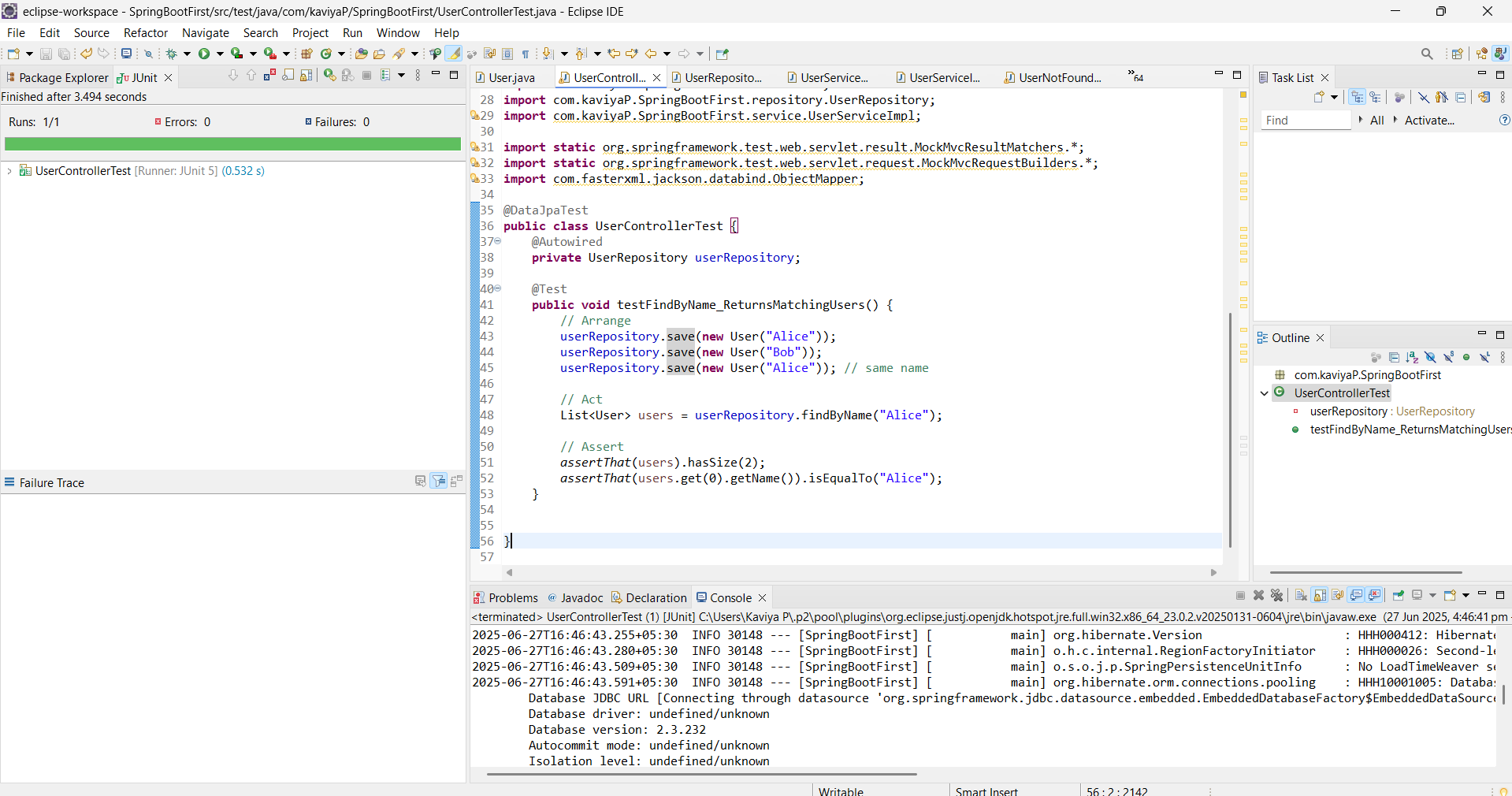
*assertThat*(users).hasSize(2);

*assertThat*(users.get(0).getName()).isEqualTo("Alice");

}

}

OUTPUT



Exercise 8: Test Controller Exception Handling

Task: Add and test a @ControllerAdvice for handling exceptions.

Exception Handler:

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(NoSuchElementException.class)

public ResponseEntity<String> handleNotFound(NoSuchElementException ex) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body("User not found");

}

}

Test:

Write code for this.

**UserController.java**

**package** com.kaviyaP.SpringBootFirst.controller;

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

**import** org.springframework.http.ResponseEntity;

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

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.service.UserService;

@RestController

@RequestMapping("/users")

**public** **class** UserController {

@Autowired

**private** UserService userService;

@GetMapping("/{id}")

**public** ResponseEntity<User> getUser(@PathVariable Long id) {

**return** ResponseEntity.*ok*(userService.getUserById(id));

}

}

**GlobalExceptionHandler.java**

**package** com.kaviyaP.SpringBootFirst.exception;

**import** java.util.NoSuchElementException;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

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

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

@ControllerAdvice

**public** **class** GlobalExceptionHandler {

@ExceptionHandler(NoSuchElementException.**class**)

**public** ResponseEntity<String> handleNotFound(NoSuchElementException ex) {

**return** ResponseEntity.*status*(HttpStatus.***NOT\_FOUND***).body("User not found");

}

}

**User.java**

**package** com.kaviyaP.SpringBootFirst.model;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

@Entity

@Table(name = "users")

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Long id;

**private** String name;

**public** User() {}

**public** User(String name) { **this**.name = name; }

**public** Long getId() { **return** id; }

**public** **void** setId(Long id) { **this**.id = id; }

**public** String getName() { **return** name; }

**public** **void** setName(String name) { **this**.name = name; }

}

**UserRepository.java**

**package** com.kaviyaP.SpringBootFirst.repository;

**import** java.util.List;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.kaviyaP.SpringBootFirst.model.User;

**public** **interface** UserRepository **extends** JpaRepository<User, Long> {

}

**UserService.java**

**package** com.kaviyaP.SpringBootFirst.service;

**import** com.kaviyaP.SpringBootFirst.model.User;

**public** **interface** UserService {

User getUserById(Long id);

}

**UserServiceImpl.java**

**package** com.kaviyaP.SpringBootFirst.service;

**import** java.util.NoSuchElementException;

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

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

**import** com.kaviyaP.SpringBootFirst.exception.UserNotFoundException;

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.repository.\*;

@Service

**public** **class** UserServiceImpl **implements** UserService{

@Autowired

**private** UserRepository userRepository;

@Override

**public** User getUserById(Long id) {

**return** userRepository.findById(id).orElseThrow(() -> **new** NoSuchElementException("User not found"));

}

}

**UserControllerTest.java**

**package** com.kaviyaP.SpringBootFirst;

**import** **static** org.mockito.Mockito.*mock*;

**import** **static** org.mockito.Mockito.*when*;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Test;

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

**import** org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

**import** org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

**import** org.springframework.boot.test.context.SpringBootTest;

**import** org.springframework.http.MediaType;

**import** org.springframework.test.web.servlet.MockMvc;

**import** **static** org.assertj.core.api.Assertions.~~assertThat~~;

**import** **static** org.hamcrest.Matchers.*is*;

**import** **static** org.junit.jupiter.api.Assertions.*assertEquals*;

**import** **static** org.junit.jupiter.api.Assertions.*assertThrows*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** java.util.List;

**import** java.util.Optional;

**import** com.kaviyaP.SpringBootFirst.controller.UserController;

**import** com.kaviyaP.SpringBootFirst.exception.UserNotFoundException;

**import** com.kaviyaP.SpringBootFirst.model.User;

**import** com.kaviyaP.SpringBootFirst.repository.UserRepository;

**import** com.kaviyaP.SpringBootFirst.service.UserServiceImpl;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

**import** com.fasterxml.jackson.databind.ObjectMapper;

@SpringBootTest

@AutoConfigureMockMvc

**public** **class** UserControllerTest {

@Autowired

**private** MockMvc mockMvc;

@Test

**public** **void** testUserNotFoundExceptionHandled() **throws** Exception {

mockMvc.perform(*get*("/users/999"))

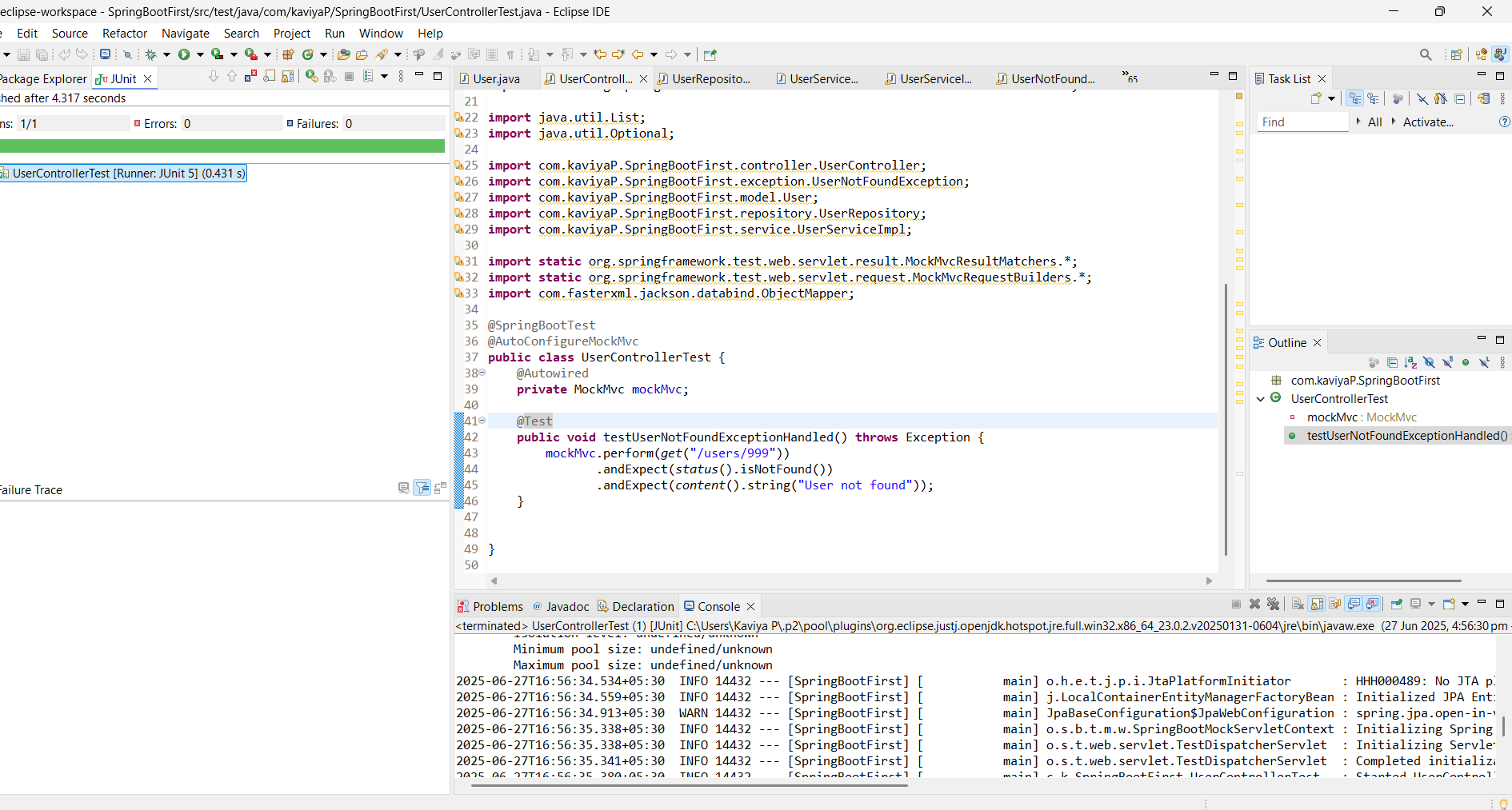
.andExpect(*status*().isNotFound())

.andExpect(*content*().string("User not found"));

}

}

OUTPUT



Exercise 9: Parameterized Test with JUnit Task: Use @ParameterizedTest to test multiple inputs. Test: Write code for this.

**MathService.java**

**package** com.kaviyaP.SpringBootFirst.service;

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

@Service

**public** **class** MathService {

**public** **boolean** isEven(**int** number) {

**return** number % 2 == 0;

}

}

**MathService.java**

**package** com.kaviyaP.SpringBootFirst;

**import** **static** org.junit.jupiter.api.Assertions.*assertEquals*;

**import** org.junit.jupiter.params.ParameterizedTest;

**import** org.junit.jupiter.params.provider.CsvSource;

**import** com.kaviyaP.SpringBootFirst.service.MathService;

**public** **class** MathServiceTest {

**private** **final** MathService mathService = **new** MathService();

@ParameterizedTest

@CsvSource({

"2, true",

"3, false",

"4, true",

"5, false",

"10, true",

"11, false"

})

**void** testIsEven(**int** input, **boolean** expected) {

**boolean** result = mathService.isEven(input);

*assertEquals*(expected, result, "Check if number is even");

}

}

OUPUT

