# Case Study: Product-Order Management System (With Mockito Testing)

## Product.java

```
package com.springtest.productordermanagement.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Product {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Long productId;

	private String name;

	private double price;

	private int availableQuantity;

	public Product() {}

	public Product(Long productId, String name, double price, int availableQuantity) {
		this.productId = productId;
		this.name = name;
		this.price = price;
		this.availableQuantity = availableQuantity;
	}

	public Long getProductId() {
		return productId;
	}

	public void setProductId(Long productId) {
		this.productId = productId;
	}

	public String getName() {
		return name;
	}

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

```
	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public int getAvailableQuantity() {
		return availableQuantity;
	}

	public void setAvailableQuantity(int availableQuantity) {
		this.availableQuantity = availableQuantity;
	}




}
```

## Order.java

```
package com.springtest.productordermanagement.entity;

import java.time.LocalDate;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Table;

@Entity
@Table(name="orders")
public class Order {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Long orderId;

	@ManyToOne
	private Product product;

	private LocalDate orderDate;
	private int quantityOrdered;

	public Order() {}
```

```
	public Order(Long orderId, Product product, LocalDate orderDate, int quantityOrdered) {
		this.orderId = orderId;
		this.product = product;
		this.orderDate = orderDate;
		this.quantityOrdered = quantityOrdered;
	}

	public Long getOrderId() {
		return orderId;
	}

	public void setOrderId(Long orderId) {
		this.orderId = orderId;
	}

	public Product getProduct() {
		return product;
	}

	public void setProduct(Product product) {
		this.product = product;
	}

	public LocalDate getOrderDate() {
		return orderDate;
	}

	public void setOrderDate(LocalDate orderDate) {
		this.orderDate = orderDate;
	}

	public int getQuantityOrdered() {
		return quantityOrdered;
	}

	public void setQuantityOrdered(int quantityOrdered) {
		this.quantityOrdered = quantityOrdered;
	}



}
```

## OrderRepository

```
package com.springtest.productordermanagement.repository;

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

import com.springtest.productordermanagement.entity.Order;
```

```
public interface OrderRepository extends JpaRepository<Order,Long> {

}
```

## ProductRepository

```
package com.springtest.productordermanagement.repository;

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

import com.springtest.productordermanagement.entity.Product;

public interface ProductRepository extends JpaRepository<Product,Long> {

}
```

## OrderService.java

```
package com.springtest.productordermanagement.service;

import java.time.LocalDate;
import java.util.List;

import org.springframework.stereotype.Service;

import com.springtest.productordermanagement.entity.Order;
import com.springtest.productordermanagement.entity.Product;
import com.springtest.productordermanagement.repository.OrderRepository;
import com.springtest.productordermanagement.repository.ProductRepository;

@Service
public class OrderService {
	private final OrderRepository orderRepo;
	private final ProductRepository productRepo;

	public OrderService(OrderRepository orderRepository,ProductRepository productRepository)
{
		this.orderRepo=orderRepository;
		this.productRepo=productRepository;
	}

	public Order placeOrder(Long productId, int quantity) {
		Product product=productRepo.findById(productId).orElseThrow();
		if(product.getAvailableQuantity()<quantity) {
			throw new IllegalArgumentException("Insufficient Stock");
		}
		product.setAvailableQuantity(product.getAvailableQuantity());
		productRepo.save(product);

		Order order=new Order();
		order.setProduct(product);
		order.setQuantityOrdered(quantity);
		order.setOrderDate(LocalDate.now());
```

```
		return orderRepo.save(order);
	}

	public List<Order> getAllOrders(){
		return orderRepo.findAll();
	}

}
```

## ProductService.java

```
package com.springtest.productordermanagement.service;

import java.util.List;

import org.springframework.stereotype.Service;

import com.springtest.productordermanagement.entity.Product;
import com.springtest.productordermanagement.repository.ProductRepository;

@Service
public class ProductService {
	private final ProductRepository productRepo;

	public ProductService(ProductRepository productRepository) {
		this.productRepo=productRepository;
	}

	public Product addProduct(Product p) {
		return productRepo.save(p);
	}

	public List<Product> getAllProducts(){
		return productRepo.findAll();
	}

	public void updateStock(Long productId,int qty) {
		Product product=productRepo.findById(productId).orElseThrow();
		product.setAvailableQuantity(qty);
		productRepo.save(product);
	}


}
```

## OrderController.java

```
package com.springtest.productordermanagement.controller;

import java.util.List;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.springtest.productordermanagement.entity.Order;
import com.springtest.productordermanagement.service.OrderService;

@RestController
@RequestMapping("/api/orders")
public class OrderController {
	private final OrderService orderService;

   public OrderController(OrderService orderService) {
      this.orderService = orderService;
   }

   @PostMapping
   public Order placeOrder(@RequestParam Long productId, @RequestParam int quantity) {
      return orderService.placeOrder(productId, quantity);
   }

   @GetMapping
   public List<Order> getAllOrders() {
      return orderService.getAllOrders();
   }
}
```

## ProductController.java

```
package com.springtest.productordermanagement.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.springtest.productordermanagement.entity.Product;
import com.springtest.productordermanagement.service.ProductService;

@RestController
@RequestMapping("/api/products")
public class ProductController {
	@Autowired
	private final ProductService productService;
	public ProductController(ProductService productService) {
```

```
		this.productService=productService;
	}

	@PostMapping
	public Product addProduct(@RequestBody Product product) {
		return productService.addProduct(product);
	}
	@GetMapping
	public List<Product> getAllProducts(){
		return productService.getAllProducts();

	}
	@PutMapping("/{id}/stock")
	public void updateStock(@PathVariable Long id,@RequestParam int qty) {
		productService.updateStock(id, qty);
	}

}
```

## OrderServiceTest.java

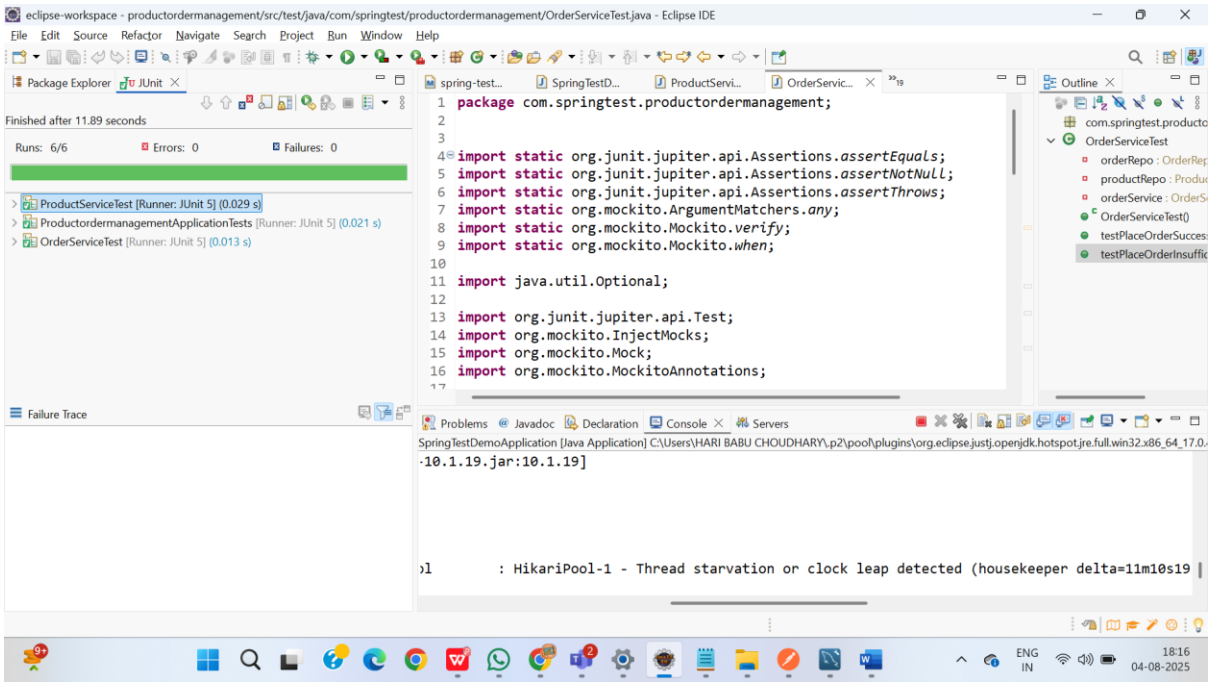
```
package com.springtest.productordermanagement;


import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;
import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.mockito.ArgumentMatchers.any;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.when;

import java.util.Optional;

import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import com.springtest.productordermanagement.entity.Order;
import com.springtest.productordermanagement.entity.Product;
import com.springtest.productordermanagement.repository.OrderRepository;
import com.springtest.productordermanagement.repository.ProductRepository;
import com.springtest.productordermanagement.service.OrderService;

public class OrderServiceTest {

   @Mock
   private OrderRepository orderRepo;

   @Mock
   private ProductRepository productRepo;
```

```
    @InjectMocks
    private OrderService orderService;

    public OrderServiceTest() {
        MockitoAnnotations.openMocks(this);
    }

    @Test
    public void testPlaceOrderSuccess() {
        Product product = new Product();
        product.setProductId(1L);
        product.setAvailableQuantity(10);

        when(productRepo.findById(1L)).thenReturn(Optional.of(product));
        when(orderRepo.save(any(Order.class))).thenAnswer(i -> i.getArgument(0));

        Order order = orderService.placeOrder(1L, 5);

        assertNotNull(order);
        assertEquals(5, order.getQuantityOrdered());
        verify(productRepo).save(product);
    }

    @Test
    public void testPlaceOrderInsufficientStock() {
        Product product = new Product();
        product.setProductId(1L);
        product.setAvailableQuantity(2);

        when(productRepo.findById(1L)).thenReturn(Optional.of(product));

        assertThrows(IllegalArgumentException.class, () -> {
            orderService.placeOrder(1L, 5);
        });
    }
}
```

## ProductServiceTest.java

```
package com.springtest.productordermanagement;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.when;

import java.util.Arrays;
import java.util.Optional;

import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
```

```
import com.springtest.productordermanagement.entity.Product;
import com.springtest.productordermanagement.repository.ProductRepository;
import com.springtest.productordermanagement.service.ProductService;

public class ProductServiceTest {

	@Mock
	private ProductRepository productRepo;

	@InjectMocks
	private ProductService productService;

	public ProductServiceTest() {
		MockitoAnnotations.openMocks(this);
	  }

	  @Test
	  public void testAddProduct() {
	    Product product = new Product();
	    when(productRepo.save(product)).thenReturn(product);
	    assertEquals(product, productService.addProduct(product));
	  }

	  @Test
	  public void testGetAllProducts() {
	    when(productRepo.findAll()).thenReturn(Arrays.asList(new Product(), new Product()));
	    assertEquals(2, productService.getAllProducts().size());
	  }

	  @Test
	  public void testUpdateStock() {
	    Product product = new Product();
	    product.setAvailableQuantity(10);
	    when(productRepo.findById(1L)).thenReturn(Optional.of(product));

	    productService.updateStock(1L, 5);

	    verify(productRepo).save(product);
	    assertEquals(5, product.getAvailableQuantity());
	  }
	}
```


eclipse-workspace - productordermanagement/src/test/java/com/springtest/productordermanagement/OrderServiceTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
Finished after 11.89 seconds
Runs: 6/6
Errors: 0
Failures: 0
ProductServiceTest [Runner: JUnit 5] (0.029 s)
ProductordermanagementApplicationTests [Runner: JUnit 5] (0.021 s)
OrderServiceTest [Runner: JUnit 5] (0.013 s)
Failure Trace
spring-test...
SpringTestD...
ProductServi...
OrderServic...
1 package com.springtest.productordermanagement;
4 import static org.junit.jupiter.api.Assertions.assertEquals;
5 import static org.junit.jupiter.api.Assertions.assertNotNull;
6 import static org.junit.jupiter.api.Assertions.assertThrows;
7 import static org.mockito.ArgumentMatchers.any;
8 import static org.mockito.Mockito.verify;
9 import static org.mockito.Mockito.when;
11 import java.util.Optional;
13 import org.junit.jupiter.api.Test;
14 import org.mockito.InjectMocks;
15 import org.mockito.Mock;
16 import org.mockito.MockitoAnnotations;
Outline
com.springtest.productor
OrderServiceTest
orderRepo : OrderRep
productRepo : Produc
orderService : OrderS
OrderServiceTest()
testPlaceOrderSucces
testPlaceOrderInsuffic
Problems
Javadoc
Declaration
Console
Servers
SpringTestDemoApplication [Java Application] C:\Users\HARI BABU CHOUDHARY\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.
·10.1.19.jar:10.1.19]
)l : HikariPool-1 - Thread starvation or clock leap detected (housekeeper delta=11m10s19
ENG
IN
18:16
04-08-2025