Day 12 Java Assignment

Product-Order Management System (With Mockito Testing)

pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.2.4</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.wipro</groupId>
	<artifactId>Assignment1_Day13_ProductOrderManagementSystem</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>Assignment1_Day13_ProductOrderManagementSystem</name>
	<description>Spring Boot Testing</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
```

```xml
		<url/>
	</scm>
	<properties>
		<java.version>17</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

    <dependency>
<groupId>com.mysql</groupId>
<artifactId>mysql-connector-j</artifactId>
<scope>runtime</scope>
    </dependency>


		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
		</dependency>
		<dependency>
      <groupId>org.mockito</groupId>
     <artifactId>mockito-core</artifactId>
```

```xml
        <scope>test</scope>
      </dependency>
            <dependency>
                <groupId>com.h2database</groupId>
                <artifactId>h2</artifactId>
                <scope>runtime</scope>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
            </dependency>
        </dependencies>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
            </plugins>
        </build>

</project>
```

<u>OrderController.java:</u>

```java
package com.example.springtest.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.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.example.springtest.entity.Order;
import com.example.springtest.service.OrderService;

@RestController
@RequestMapping("/api/orders")
public class OrderController {
        @Autowired
  private 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();
  }
}
```

<u>ProductController:</u>

```
package com.example.springtest.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.example.springtest.entity.Product;
import com.example.springtest.service.ProductService;

@RestController
@RequestMapping("/api/products")
public class ProductController {
        @Autowired
          private 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 Product updateStock(@PathVariable Long id, @RequestParam int quantity)
{
             return productService.updateStock(id, quantity);
          }
}
```

<u>Order.java:</u>

```
package com.example.springtest.entity;
import java.time.LocalDateTime;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Table;


@Entity
@Table(name="orders")
public class Order {
	@Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long orderId;
   @ManyToOne
   @JoinColumn(name = "product_id")
   private Product product;
   private LocalDateTime orderDate;
   @Column(name="quantityOrdered")
   private int quantityOrdered;
   public Order() {
		// TODO Auto-generated constructor stub
	}
	public Order(Long orderId, Product product, LocalDateTime 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 LocalDateTime getOrderDate() {
		return orderDate;
	}
	public void setOrderDate(LocalDateTime orderDate) {
		this.orderDate = orderDate;
	}
	public int getQuantityOrdered() {
		return quantityOrdered;
	}
	public void setQuantityOrdered(int quantityOrdered) {
		this.quantityOrdered = quantityOrdered;
	}
}
```

Product.java:

```
package com.example.springtest.entity;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="products")
public class Product {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long productId;
	@Column(name="name")
	private String name;
	@Column(name="price")
	private double price;
	@Column(name="availableQuantity")
	private int availableQuantity;
	public Product() {
		// TODO Auto-generated constructor stub
	}
	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;
	}
}
```

Repository:

OrderRepository.java:

```
package com.example.springtest.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.springtest.entity.Order;
public interface OrderRepository extends JpaRepository<Order, Long>{
}
```

ProductRepository.java:

```
package com.example.springtest.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.springtest.entity.Product;
public interface ProductRepository extends JpaRepository<Product, Long>{
}
```

Service

OrderService.java:

```
package com.example.springtest.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.springtest.entity.Order;
import com.example.springtest.entity.Product;
import com.example.springtest.repository.OrderRepository;
import com.example.springtest.repository.ProductRepository;

@Service
public class OrderService {
        @Autowired
  private OrderRepository orderRepository;
  @Autowired
  private ProductRepository productRepository;
  public Order placeOrder(Long productId, int quantity) {
    Product product = productRepository.findById(productId)
        .orElseThrow(() -> new RuntimeException("Product not found"));
    if (product.getAvailableQuantity() < quantity) {
      throw new RuntimeException("Insufficient stock");
    }
```

```
        product.setAvailableQuantity(product.getAvailableQuantity() - quantity);
        productRepository.save(product);
        Order order = new Order();
        return orderRepository.save(order);
    }
    public List<Order> getAllOrders() {
        return orderRepository.findAll();
    }
}
```

ProductService.java:

```
package com.example.springtest.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.springtest.entity.Product;
import com.example.springtest.repository.ProductRepository;

@Service
public class ProductService {
            @Autowired
    private ProductRepository productRepository;
    public Product addProduct(Product product) {
        return productRepository.save(product);
    }
    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }
    public Product updateStock(Long productId, int quantity) {
        Product product = productRepository.findById(productId)
                .orElseThrow(() -> new RuntimeException("Product not found"));
        product.setAvailableQuantity(quantity);
        return productRepository.save(product);
    }
}
```

ProductOrderManagementSystemApplication.java:

```
package com.example.springtest;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class ProductOrderManagementSystemApplication {
        public static void main(String[] args) {
                SpringApplication.run(ProductOrderManagementSystemApplication.class,
args);
        }
}
```

application.properties:

```
spring.application.name=ProductOrderManagementSystem
spring.datasource.url=jdbc:mysql://localhost:3306/product_order_db
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=password@123
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
server.port=8080
spring.h2.console.enabled=false
```

ProductOrderManagementSystemApplicationTests.java:

```
package com.example.springtest;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
@SpringBootTest
class ProductOrderManagementSystemApplicationTests {
        @Test
        void contextLoads() {
        }
}
```

OrderServiceTest.java:

```
package com.example.springtest;
import static org.assertj.core.api.Assertions.*;
import static org.mockito.Mockito.*;
import java.time.LocalDateTime;
import java.util.Arrays;
import java.util.List;
import java.util.Optional;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import com.example.springtest.entity.Order;
import com.example.springtest.entity.Product;
import com.example.springtest.repository.OrderRepository;
import com.example.springtest.repository.ProductRepository;
import com.example.springtest.service.OrderService;
```

```
@ExtendWith(MockitoExtension.class)
public class OrderServiceTest {
   @Mock
   private OrderRepository orderRepository;
   @Mock
   private ProductRepository productRepository;
   @InjectMocks
   private OrderService orderService;
   @Test
   public void testPlaceOrder_Success() {
      Product iceCream = new Product(1L, "Ice Cream", 20, 82);
      Order expectedOrder = new Order();
      expectedOrder.setOrderId(1L);
      expectedOrder.setProduct(ice Cream);
      expectedOrder.setQuantityOrdered(10);
      expectedOrder.setOrderDate(LocalDateTime.now());
      when(productRepository.findById(1L)).thenReturn(Optional.of(iceCream));
      when(orderRepository.save(any(Order.class))).thenReturn(expectedOrder);
      Order result = orderService.placeOrder(1L, 10);
      assertThat(result).isNotNull();
      assertThat(result.getQuantityOrdered()).isEqualTo(10);
      assertThat(result.getProduct().getName()).isEqualTo("Ice Cream ");
   }

   @Test
   public void testPlaceOrder_InsufficientStock() {
      Product goodDay = new Product(2L, "GoodDay", 5, 155);
      when(productRepository.findById(2L)).thenReturn(Optional.of(goodDay));
      assertThatThrownBy(() -> orderService.placeOrder(2L, 200))
         .isInstanceOf(RuntimeException.class)
```

```
            .hasMessageContaining("Insufficient stock");
        verify(orderRepository, never()).save(any());
    }

    @Test
    public void testGetAllOrders() {
        Product iceCream = new Product(1L, "Ice Cream", 20, 82);
        Product goodDay= new Product(2L, "GoodDay", 5, 155);
        Order order1 = new Order();
        order1.setOrderId(1L);
        order1.setProduct(iceCream);
        order1.setQuantityOrdered(2);

        Order order2 = new Order();
        order2.setOrderId(2L);
        order2.setProduct(goodDay);
        order2.setQuantityOrdered(10);
        when(orderRepository.findAll()).thenReturn(Arrays.asList(order1, order2));
        List<Order> result = orderService.getAllOrders();
        assertThat(result)
            .hasSize(2)
            .extracting(Order::getProduct)
            .extracting(Product::getName)
            .containsExactly("Ice Cream", "GoodDay");
    }
}
```

<u>ProductServiceTest.java:</u>

```
package com.example.springtest;
```

```
import static org.assertj.core.api.Assertions.*;
import static org.mockito.ArgumentMatchers.*;
import static org.mockito.Mockito.*;
import java.util.Arrays;
import java.util.List;
import java.util.Optional;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import com.example.springtest.entity.Product;
import com.example.springtest.repository.ProductRepository;
import com.example.springtest.service.ProductService;

@ExtendWith(MockitoExtension.class)
public class ProductServiceTest {
    @Mock
    private ProductRepository productRepository;
    @InjectMocks
    private ProductService productService;
    @Test
    public void testAddProduct() {
        Product basumatiRice = new Product(1L, "Basumati Rice", 155, 160);
        when(productRepository.save(any(Product.class))).thenReturn(basumatiRice);
        Product result = productService.addProduct(basumatiRice);
        assertThat(result)
            .isNotNull()
            .extracting(Product::getName, Product::getPrice)
            .containsExactly("basumatiRice ", 155.0);
        verify(productRepository).save(basumatiRice);
    }

    @Test
    public void testGetAllProducts() {
        Product coffee = new Product(1L, "Coffee", 100, 85);
        Product spicies = new Product(2L, "Spicies", 55, 150);
        when(productRepository.findAll()).thenReturn(Arrays.asList(coffee,spicies));
        List<Product> result = productService.getAllProducts();
        assertThat(result)
            .hasSize(2)
            .extracting(Product::getName)
            .containsExactly("Coffee ", " Spicies ");
    }

    @Test
    public void testUpdateStock() {
        Product goodDay = new Product(1L, "GoodDay", 5, 155);
        Product updatedProduct = new Product(1L, "GoodDay", 5, 100);
        when(productRepository.findById(1L)).thenReturn(Optional.of(parleGBiscuits));
        when(productRepository.save(any(Product.class))).thenReturn(updatedProduct);
        Product result = productService.updateStock(1L, 100);
        assertThat(result)
            .isNotNull()
            .extracting(Product::getAvailableQuantity)
            .isEqualTo(100);
        verify(productRepository).findById(1L);
        verify(productRepository).save(argThat(p -> p.getAvailableQuantity() == 100));
    }
```

```
    @Test
    public void testUpdateStock_ProductNotFound() {
        when(productRepository.findById(99L)).thenReturn(Optional.empty());
        assertThatThrownBy(() -> productService.updateStock(99L, 100))
            .isInstanceOf(RuntimeException.class)
            .hasMessageContaining("Product not found");
        verify(productRepository, never()).save(any());
    }
}
```