Case Study on Ecommerce Application

Submitted By: Vrushali Tekchand Rahangdale

Email Id: vrushutr08@gmail.com

1. **Create following tables in SQL Schema with appropriate class and write the unit test case for the Ecommerce application.**

Schema Design:

1. customers table:

- customer_id (Primary Key)
- name
- email
- password

```
CREATE TABLE customers (
    customer_id INTEGER PRIMARY KEY,
    name varchar(20) ,
    email varchar(20) ,
    password varchar(20)
);
desc customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
name	varchar(20)	YES		NULL	
email	varchar(20)	YES		NULL	
password	varchar(20)	YES		NULL	

2. products table:

- product_id (Primary Key)
- name
- price
- description
- stockQuantity

```
CREATE TABLE products (
    product_id int PRIMARY KEY,
    name varchar(20),
    price int,
    description varchar(50),
    stockQuantity INTEGER
);
desc products;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	NULL	
name	varchar(20)	YES		NULL	
price	int	YES		NULL	
description	varchar(50)	YES		NULL	
stockQuantity	int	YES		NULL	

3. cart table:

- cart_id (Primary Key)
- customer_id (Foreign Key)
- product_id (Foreign Key)
- quantity

```sql
CREATE TABLE cart (
    cart_id int PRIMARY KEY,
    customer_id int,
    product_id int,
    quantity int,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
desc cart;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
cart_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
product_id	int	YES	MUL	NULL	
quantity	int	YES		NULL	

4. orders table:

- order_id (Primary Key)
- customer_id (Foreign Key)
- order_date
- total_price
- shipping_address

```sql
CREATE TABLE orders (
    order_id int PRIMARY KEY,
    customer_id int,
    order_date date,
    total_price int,
    shipping_address varchar(30),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
desc orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
order_date	date	YES		NULL	
total_price	int	YES		NULL	
shipping_address	varchar(30)	YES		NULL	

5. order_items table (to store order details):

- order_item_id (Primary Key)
- order_id (Foreign Key)
- product_id (Foreign Key)

• quantity

```
150 • CREATE TABLE order_items (
151       order_item_id int PRIMARY KEY,
152       order_id int,
153       product_id int,
154       quantity int,
155       FOREIGN KEY (order_id) REFERENCES orders(order_id),
156       FOREIGN KEY (product_id) REFERENCES products(product_id)
157   );
158 • desc order_items;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
order_item_id	int	NO	PRI	NULL	
order_id	int	YES	MUL	NULL	
product_id	int	YES	MUL	NULL	
quantity	int	YES		NULL	

Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters)

6. Service Provider Interface/Abstract class:

Keep the interfaces and implementation classes in package dao

- Define an OrderProcessorRepository interface/abstract class with methods for adding/removing products to/from the cart and placing orders. The following methods will interactwith database.

1. createProduct()
parameter: Product product
return type: Boolean

2. createCustomer()
parameter: Customer customer
return type: boolean

3. deleteProduct()
parameter: productId
return type: boolean

4. deleteCustomer(customerId)
parameter: customerId
return type: boolean

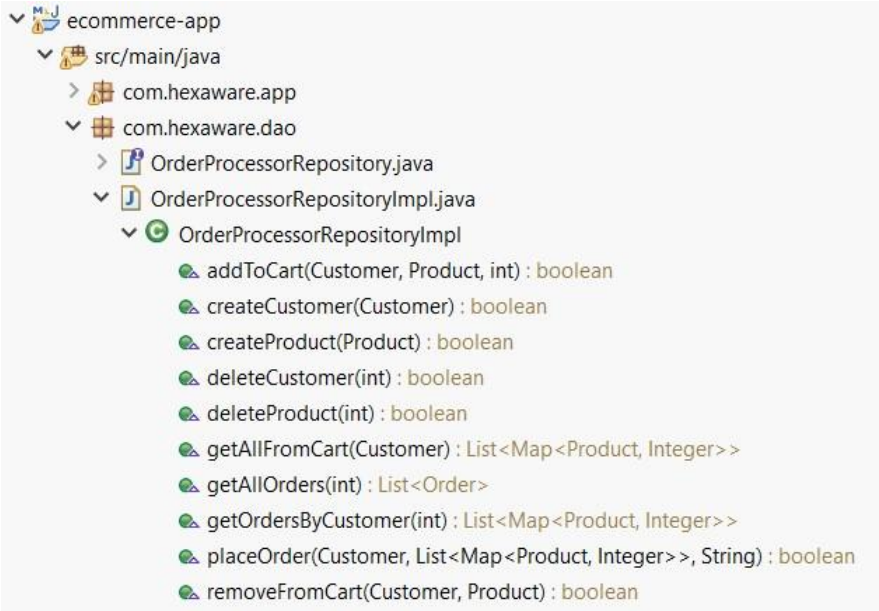
5. addToCart(): insert the product in cart.
parameter: Customer customer, Product product, int quantity
return type: boolean

6. removeFromCart(): delete the product in cart.
parameter: Customer customer, Product product
return type: boolean

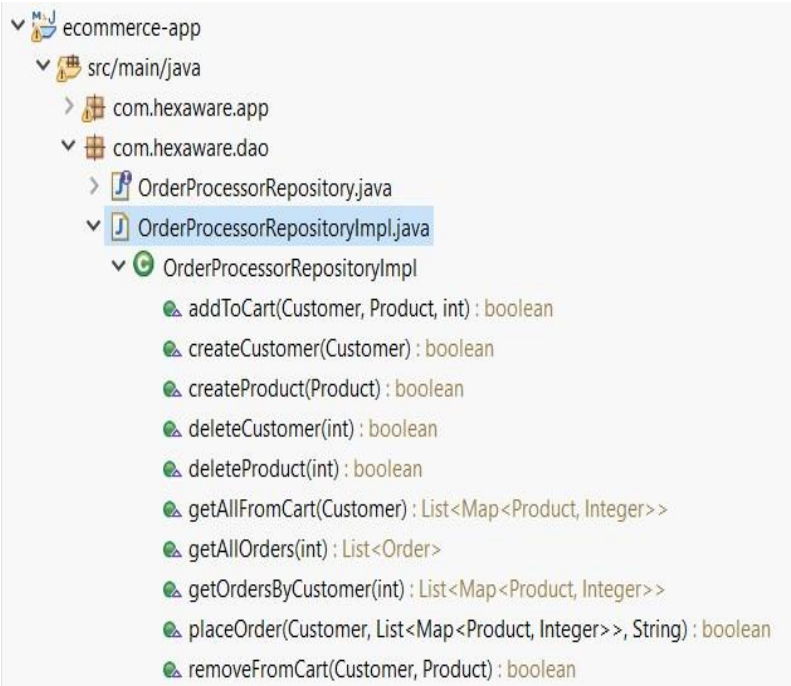
7. getAllFromCart(Customer customer): list the product in cart for a customer.
parameter: Customer customer
return type: list of product

8. placeOrder(Customer customer, List<Map<Product,quantity>>, string shippingAddress): should update order table and orderItems table.
1. parameter: Customer customer, list of product and quantity
2. return type: boolean

9. getOrdersByCustomer()
1. parameter: customerid
2. return type: list of product and quantity



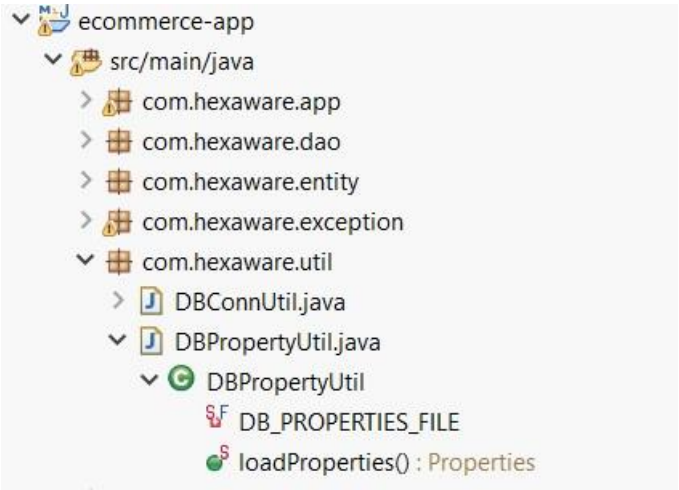
7. Implement the above interface in a class called OrderProcessorRepositoryImpl in package dao



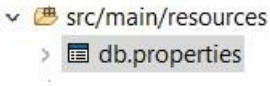
Connect your application to the SQL database:

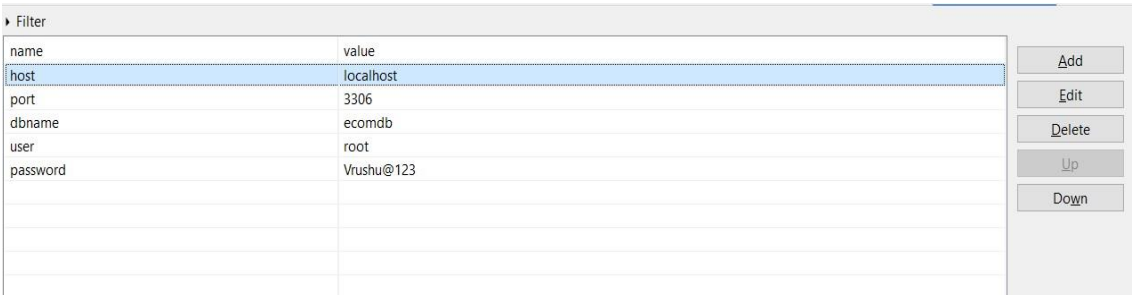
8. Write code to establish a connection to your SQL database.

- **Create a utility class DBConnection in a package util with a static variable connection of Type Connection and a static method getConnection() which returns connection.**
- **Connection properties supplied in the connection string should be read from a property file.**



• **Create a utility class PropertyUtil which contains a static method named getPropertyString() which reads a property file containing connection details like hostname, dbname, username, password, port number and returns a connection string**.

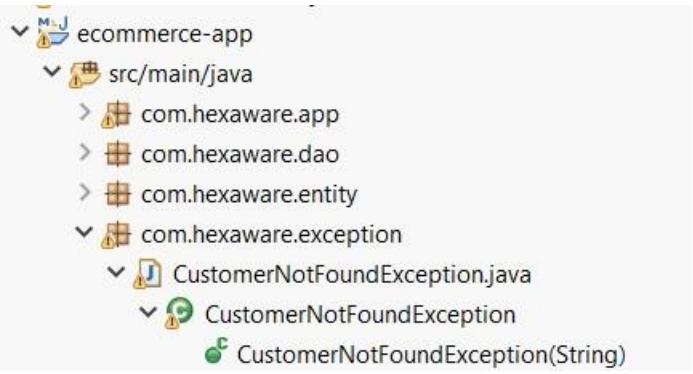




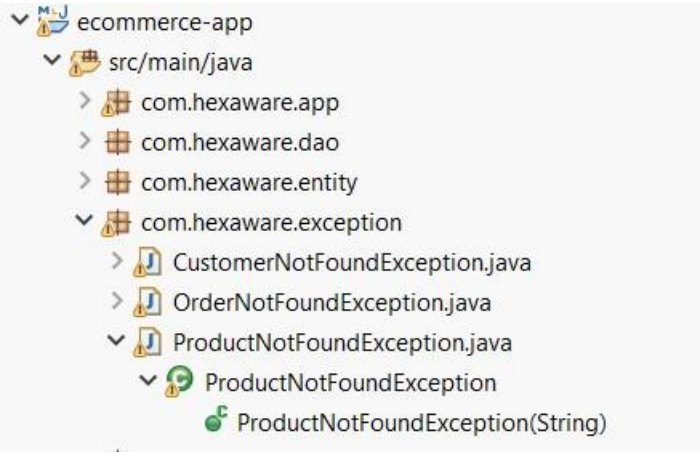
9. Create the exceptions in package myexceptions and create the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

• **CustomerNotFoundException:** throw this exception when user enters an invalid customer id which doesn't exist in db

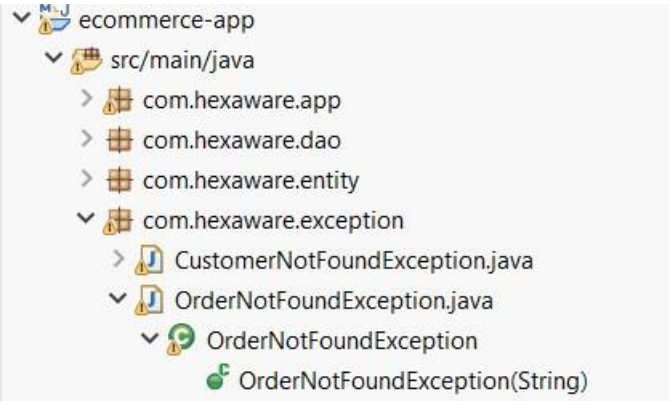
•



ProductNotFoundException: throw this exception when user enters an invalid product id which doesn't exist in db

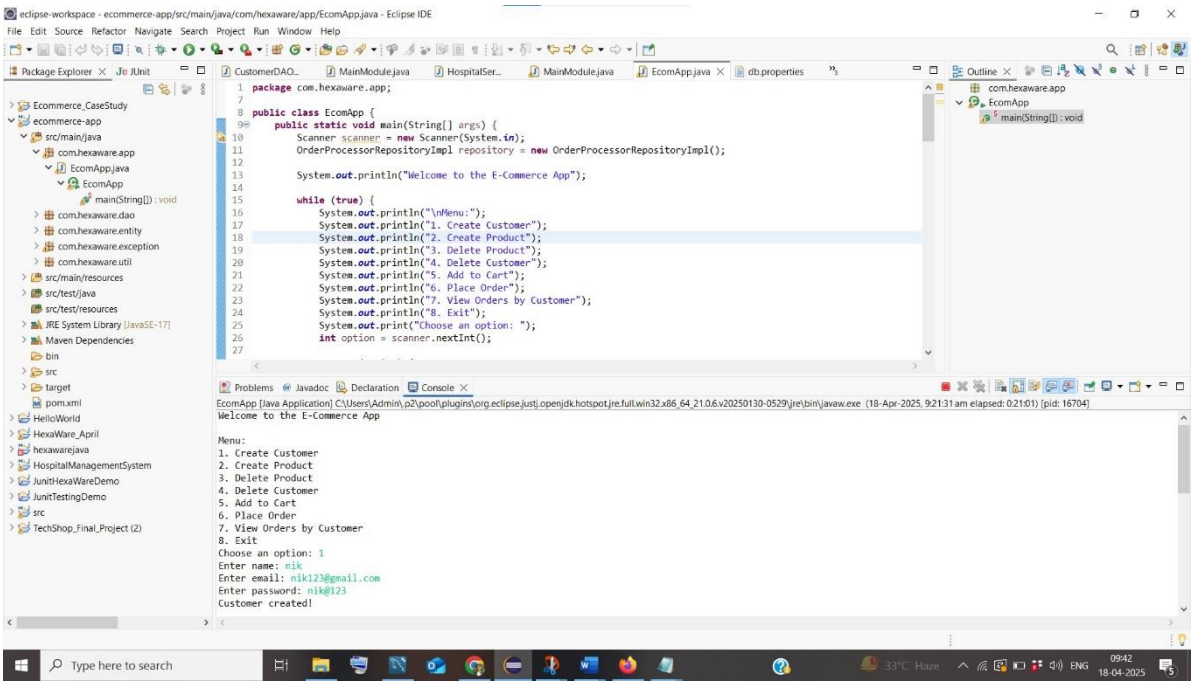


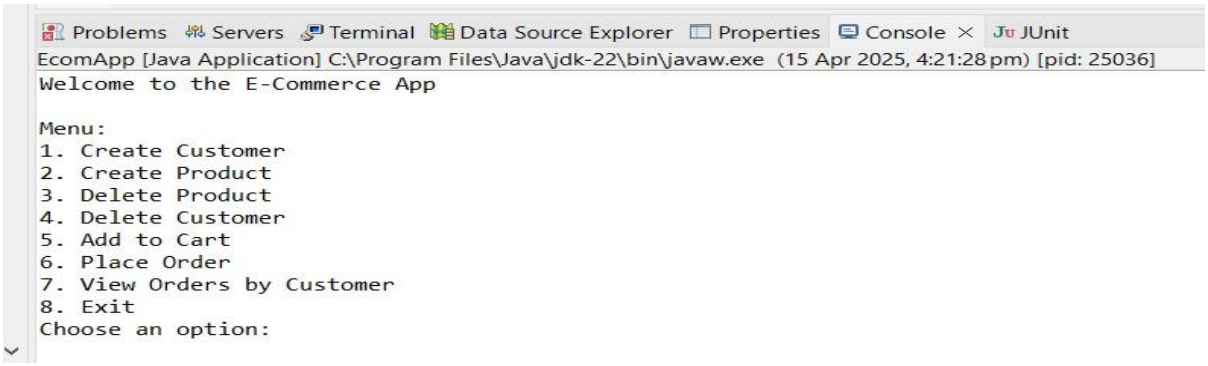
• **OrderNotFoundException:** throw this exception when user enters an invalid order id which doesn't exist in db



10. Create class named EcomApp with main method in app Trigger all the methods in service implementation class by user choose operation from the following menu.

1. **Register Customer.**
2. **Create Product.**
3. **Delete Product.**
4. **Add to cart.**
5. **View cart.**
6. **Place order.**
7. **View Customer Order**





- **Create Customer**

```
Menu:
1. Create Customer
2. Create Product
3. Delete Product
4. Delete Customer
5. Add to Cart
6. Place Order
7. View Orders by Customer
8. Exit
Choose an option: 1
Enter name: nik
Enter email: nik123@gmail.com
Enter password: nik@123
Customer created!
```

Result Grid | Filter Rows: | Edit: | Expo

customer_id	name	email	password
1	John Doe	john@example.com	password123
2	vrushu	vrushali@gmail.com	vrush@13
3	nik	nik123@gmail.com	nik@123
NULL	NULL	NULL	NULL

- **Create Product**

```
Menu:
1. Create Customer
2. Create Product
3. Delete Product
4. Delete Customer
5. Add to Cart
6. Place Order
7. View Orders by Customer
8. Exit
Choose an option: 2
Product Name: EarBuds
Price: 2000
Description: Latest Model
Stock: 15
Product created!
```

Result Grid | Filter Rows: | Edit:

product_id	name	price	description	stock
1	Chair	149.99	Ergonomic office chair	20
2	Laptop	950000	Lenovo	25
3	EarBuds	2000	Latest Model	15
NULL	NULL	NULL	NULL	NULL

- **Delete Product**

```
Menu:
1. Create Customer
2. Create Product
3. Delete Product
4. Delete Customer
5. Add to Cart
6. Place Order
7. View Orders by Customer
8. Exit
Choose an option: 3
Enter Product ID to delete: 1
Product deleted!
```

Result Grid | Filter Rows: | Edit:

product_id	name	price	description	stock
2	Laptop	950000	Lenovo	25
3	EarBuds	2000	Latest Model	15
NULL	NULL	NULL	NULL	NULL

4. Delete Customer

```
Menu:
1. Create Customer
2. Create Product
3. Delete Product
4. Delete Customer
5. Add to Cart
6. Place Order
7. View Orders by Customer
8. Exit
Choose an option: 4
Enter Customer ID to delete: 1
Customer deleted!
```

Result Grid | Filter Rows: | Edit:

customer_id	name	email	password
2	vrushu	vrushali@gmail.com	vrush@13
3	nik	nik123@gmail.com	nik@123
NULL	NULL	NULL	NULL

5. Add to Cart

```
Menu:
1. Create Customer
2. Create Product
3. Delete Product
4. Delete Customer
5. Add to Cart
6. Place Order
7. View Orders by Customer
8. Exit
Choose an option: 5
Enter Customer ID: 3
Enter Product ID: 2
Enter Quantity: 5
Added to cart!
```

Result Grid | Filter Rows: | Edit:

cart_id	customer_id	product_id	quantity
1	3	2	5
NULL	NULL	NULL	NULL

6. Place Order

```
Menu:
1. Create Customer
2. Create Product
3. Delete Product
4. Delete Customer
5. Add to Cart
6. Place Order
7. View Orders by Customer
8. Exit
Choose an option: 6
Enter Customer ID: 2
Enter Shipping Address: Mumbai
Order placed successfully!
```

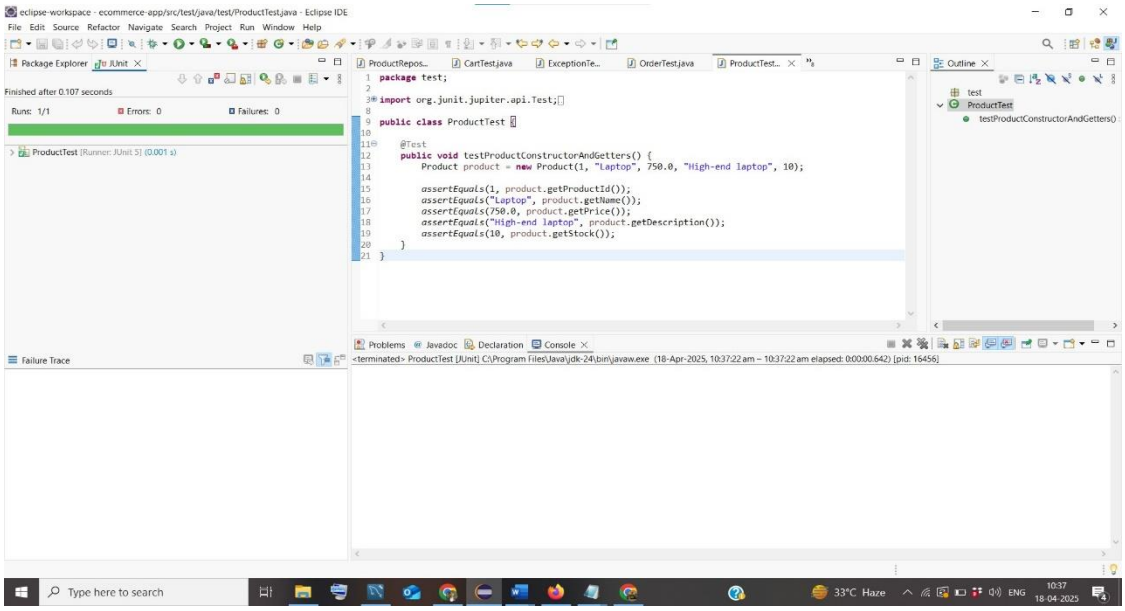
Result Grid | Filter Rows: | Edit: | Export/In

order_id	customer_id	order_date	total_price	shipping_address
1	2	2025-04-18	0	Mumbai
NULL	NULL	NULL	NULL	NULL

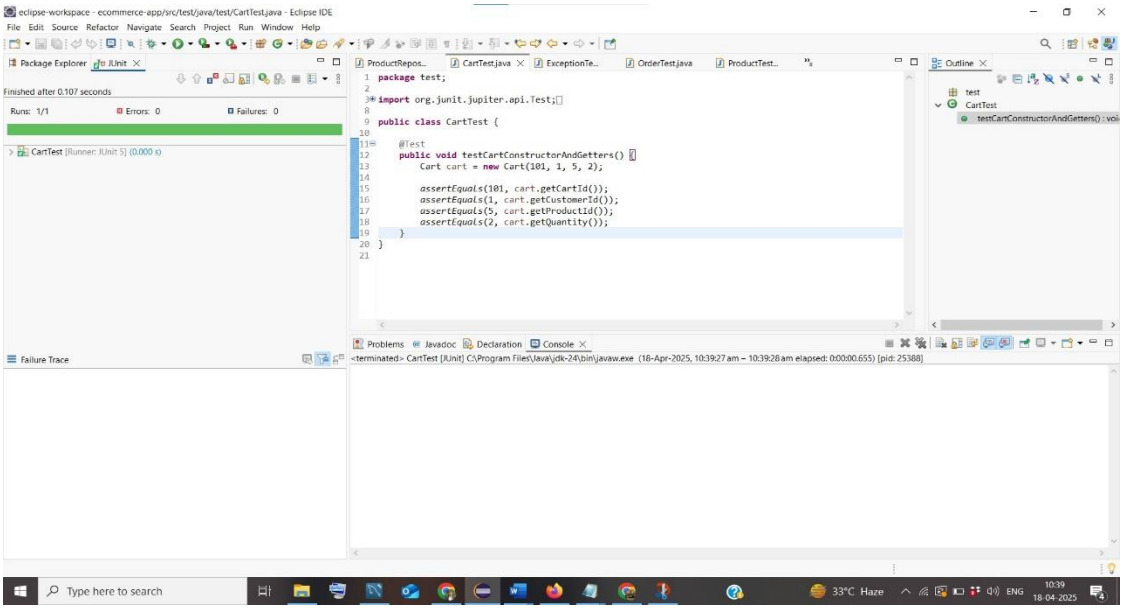
Unit Testing

- **Create Unit test cases for Ecommerce System are essential to ensure the correctness and reliability of your system. Following questions to guide the creation of Unit test cases:**

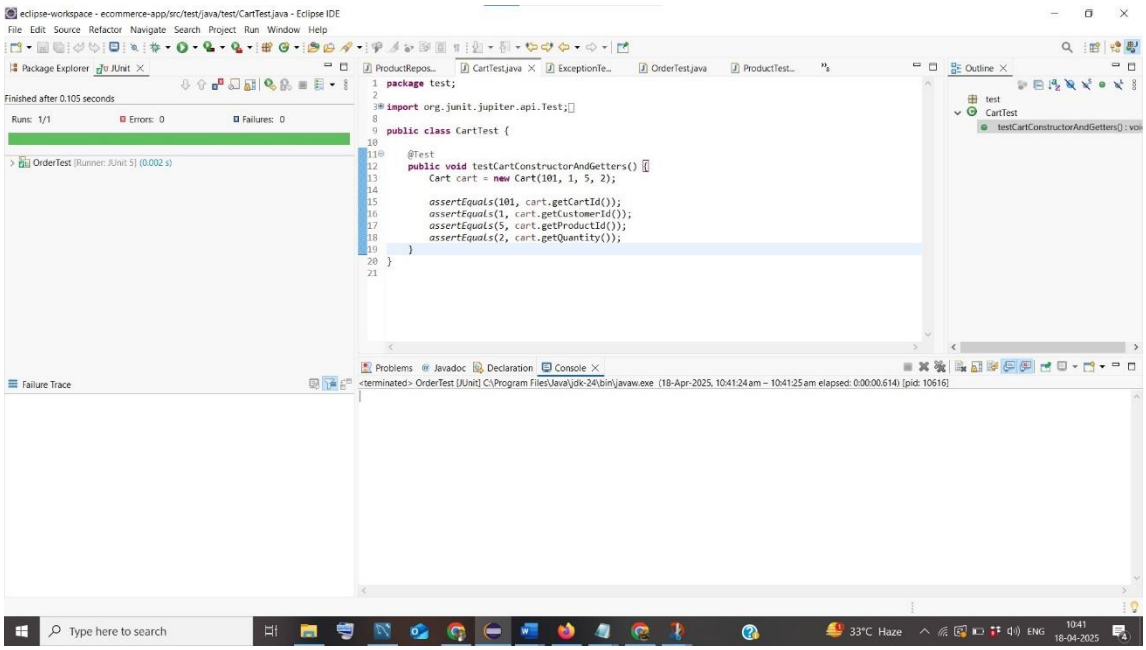
- **Write test case to test Product created successfully or not.**



- **Write test case to test product is added to cart successfully or not.**



- **Write test case to test product is ordered successfully or not.**



- **Write test case to test exception is thrown correctly or not when customer id or product id not found in database.**