

# **Lab6 - Question 2**

**Jarir Bookstore is a well-known Saudi retail store that offers books and electronics and office supplies In this work**

**I will apply validation to 3 models :**

**1-Product**

**2-Customer**

**4-Order**

# Product Model

```
@NotBlank(message = "product name cannot be empty");
@Size(min = 2 , message = "name must be more than 2");
@Pattern(regexp = ""^[\p{L}0-9 ]+$" , message = "name just letters and digit
s ")
private String name;
```

**@NotBlank** Used to ensure the product name is not empty because a product must have a display name and **@Pattren** Used to allow only letters, numbers, and spaces to keep the name meaningful and readable.

```
@NotBlank(message = "id should be 2 number or more")
@Pattern(regexp = "^[0-9]{2,}$", message = "id number must be 2 digits or m
ore ")
private String productID;
```

**@NotBlank** Used to ensure the product ID is not empty because it is important for identifying the product, and **@Pattern** was used to allow only letters, numbers, and hyphens so the ID stays clean and easy to reference.

```
@Min(value = 1 , message = "price must equal 1 or greater than ")
@NotNull(message = "price is required")
private double price;
```

**@Min** used to ensure the price is equal to or greater than 1 because a product cannot have zero or negative price.

| @NotNull used to ensure the price field is required and cannot be missing.

```
@NotBlank(message = "Category cannot be empty")
@Pattern(regexp = "^[\\p{L} ]+$", message = "category must contain letters only")
private String category;
```

| @NotBlank used to ensure the category is not empty because every product must belong to a category.  
| @Pattern used to ensure the category contains letters only to keep categories organized and standardized.

```
@NotNull(message = "Stock quantity is required")
@PositiveOrZero(message = "stock must be zero or greater ")
private int stock;
```

| @NotNull used to ensure the stock field is required and cannot be missing  
| @@PositiveOrZero because number Must be 0 or greater than

## Customer model

```
@NotBlank(message = "Customer ID cannot be empty")
@Pattern(regexp = "^\\d{10}$", message = "Customer ID must be 10 digits (National ID)")
private String customerId;
```

| @NotBlank used ensure the Customer not empty because it's very important  
| @Pattren used ensure the Customer take just numbers and 10 digits

```
@NotBlank(message = "Full name cannot be empty")
@Size(min = 2 , message = "Full name must be 2 letter or more")
```

```
@Pattern(regexp = "^[\\p{L} ]+$", message = "Full name must contain letters  
and spaces only")  
private String fullName;
```

| @NotBlank used ensure the name cannot be empty because it's very important

| @Pattren used ensure the Customer take just letters and space only

```
@NotNull(message = "Phone number cannot be empty")  
@Pattern(regexp = "^05\\d{8}$", message = "Phone number must start with 0  
5 and be 10 digits")  
private String phone;
```

| @NotNull used ensure the name cannot be empty because it's very important

| @Pattren used ensure the Phone number take just numbers with start 05 and  
be 10 digits

```
@NotEmpty(message = "Email cannot be empty")  
@Email(message = "Invalid email format")  
private String email;
```

| @NotEmpty used ensure the email cannot be empty because it's very important

| @Email used ensure the email Must be has @

```
@NotEmpty(message = "Address cannot be empty")  
private String address;
```

| @NotEmpty used ensure the email cannot be empty because it's very important

| @Email used ensure the email Must be has @

```
@NotEmpty(message = "Address cannot be empty")  
private String address;
```

| @NotEmpty used ensure the address cannot by empty it's required

## Order model

```
@NotBlank(message = "Order ID cannot be empty")
@Pattern(regexp = "^\\d+$", message = "Order ID must contain digits only")
private String orderId;
```

| @Pattren used ensure the order id must contain digits only

| @NotEmpty used ensure the order id cannot by empty

```
@NotBlank(message = "Customer ID cannot be empty")
@Pattern(regexp = "^\\d{10}$", message = "Customer ID must be 10 digits (National ID)")
private String customerId;
```

| @Pattren used ensure the Customer take just numbers and 10 digits

| @NotEmpty used ensure the Customer not empty because it's very important

```
@NotBlank(message = "id should be 2 number or more")
@Pattern(regexp = "^[0-9]{2,}$", message = "id number must be 2 digits or more ")
private String productID;
```

| @Pattern was used to allow only letters, numbers, and hyphens so the ID stays clean and easy to reference.

| @NotBlank Used to ensure the product ID is not empty because it is important for identifying the product

```
@NotNull(message = "Quantity is required")
@Min(value = 1, message = "Quantity must be at least 1")
private int quantity;
```

| @NotNull quantity is required

| @Min used ensure the quantity Must be at least 1

```
@NotNull(message = "Total price is required")
@Positive(message = "Total price must be greater than 0")
private double totalPrice;
```