JAVA FUNDAMENTALS COURSE

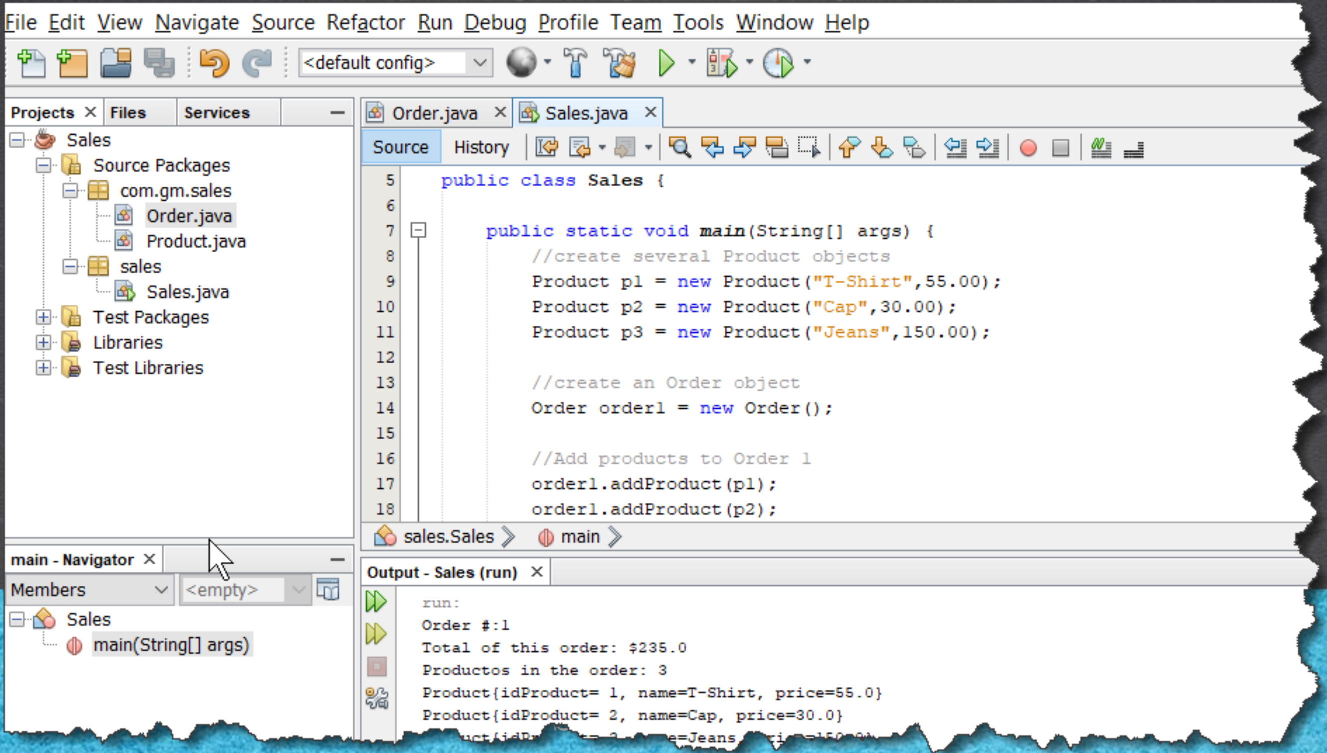
EXERCISE

CLASS DESIGN IN JAVA

Experiencia y Conocimiento para tu vida

EXERCISE OBJECTIVE

With this exercise we will put into practice the concept of Class Design in Java. At the end we will observe the following:

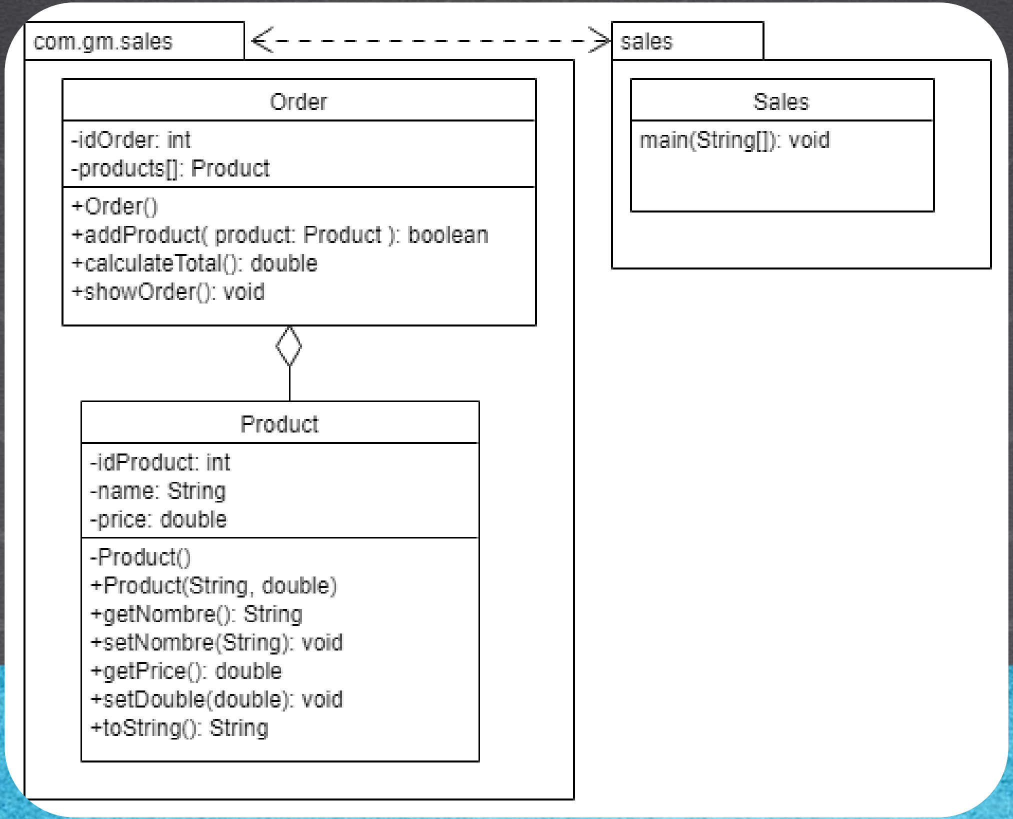


```java
public class Sales {

    public static void main(String[] args) {
        //create several Product objects
        Product p1 = new Product("T-Shirt",55.00);
        Product p2 = new Product("Cap",30.00);
        Product p3 = new Product("Jeans",150.00);

        //create an Order object
        Order order1 = new Order();

        //Add products to Order 1
        order1.addProduct(p1);
        order1.addProduct(p2);
```

```
run:
Order #:1
Total of this order: $235.0
Productos in the order: 3
Product{idProduct= 1, name=T-Shirt, price=55.0}
Product{idProduct= 2, name=Cap, price=30.0}
```

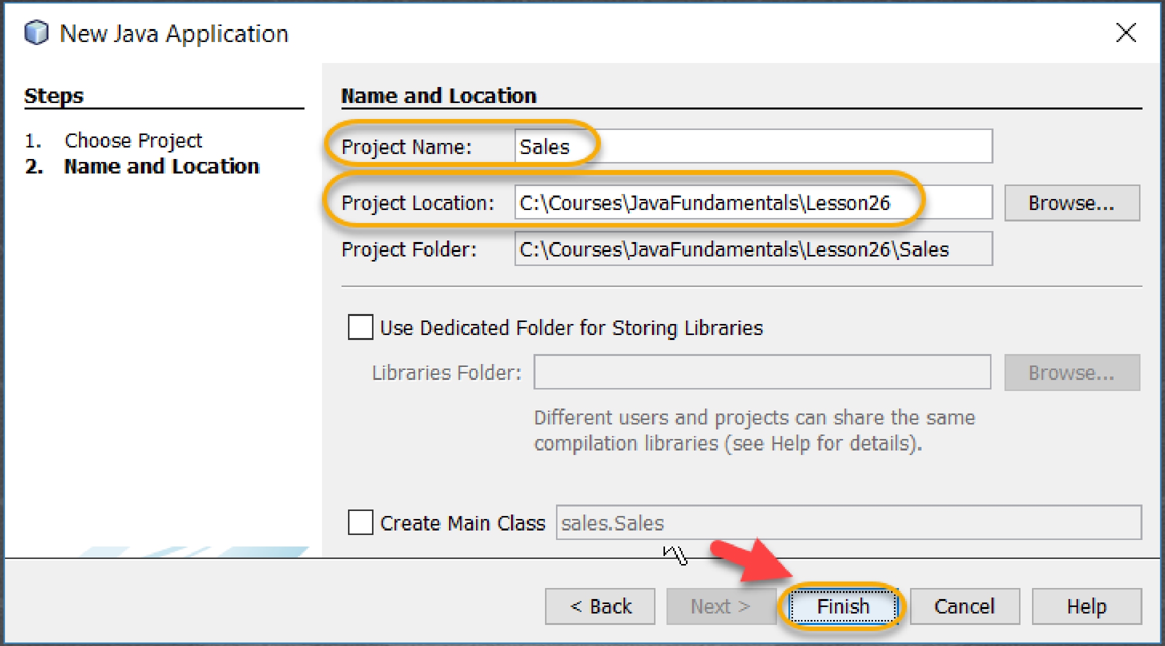
EXERCISE OBJECTIVE

We will create the exercise based on the following diagram:



1. CREATE A PROJECT

Create a new project:



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: Sales

Project Location: C:\Courses\JavaFundamentals\Lesson26

Project Folder: C:\Courses\JavaFundamentals\Lesson26\Sales

☐ Use Dedicated Folder for Storing Libraries

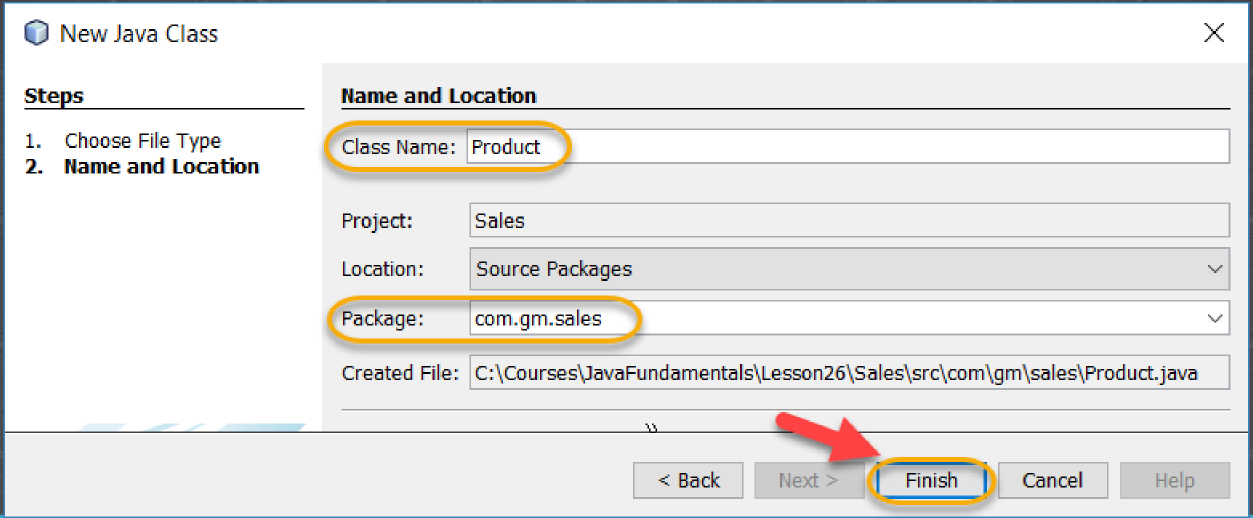
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class sales.Sales

2. CREATE A NEW CLASS

We first create the class that has no dependency with other classes. So, first create the class called Product.



Class Name: Product

Project: Sales

Location: Source Packages

Package: com.gm.sales

Created File: C:\Courses\JavaFundamentals\Lesson26\Sales\src\com\gm\sales\Product.java

3. MODIFY THE CODE

Product.java:

```java
package com.gm.sales;

public class Product {

    private int idProduct;
    private String name;
    private double price;
    private static int productsCounter;

    //Empty constructor
    private Product() {
        //Assign the idProduct that is unique to every created object
        this.idProduct = ++productsCounter;
    }

    //Overloaded constructor with 2 arguments
    public Product(String name, double price) {
        //Call the private constructor to assign the idProduct value
        this();
        this.name = name;
        this.price = price;
    }
```

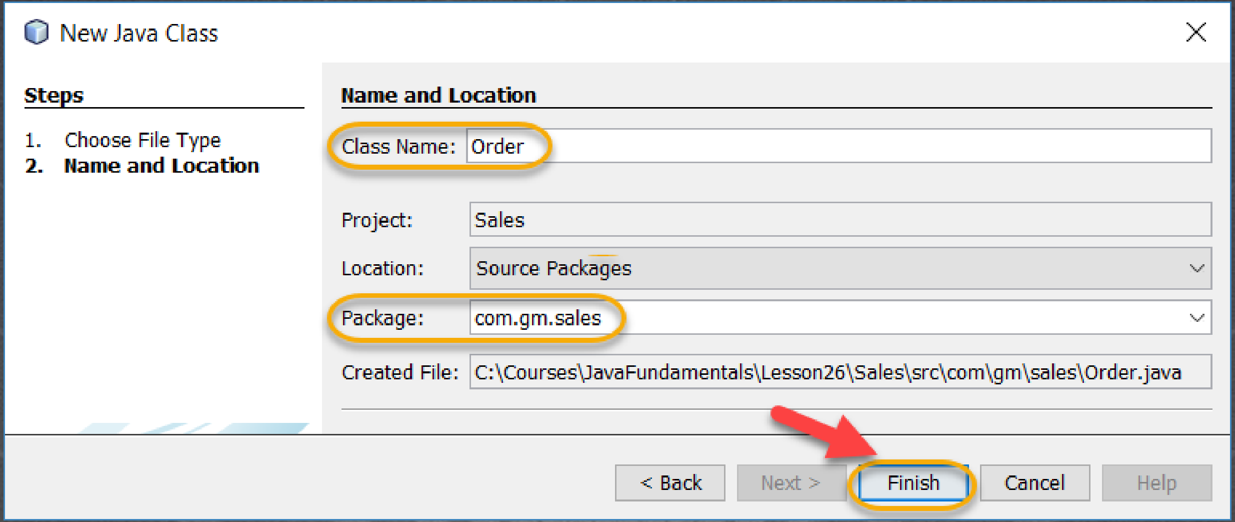
3. MODIFY THE CODE

Product.java:

```java
    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;
    }

    @Override
    public String toString() {
        return "Product{" + "idProduct= " + idProduct + ", name=" + name + ", price=" + price + '}';
    }
}
```

PASO 4. CREATE A CLASS

Create the Order class:



Class Name: Order

Project: Sales

Location: Source Packages

Package: com.gm.sales

Created File: C:\Courses\JavaFundamentals\Lesson26\Sales\src\com\gm\sales\Order.java

5. MODIFY THE CODE

Order.java:

```java
package com.gm.sales;

public class Order {

    private final int idOrder;
    //Declare the products array
    private final Product products[];
    private static int ordersCounter;
    private int productsCounter;
    //define the maximum elements of the array
    private static final int MAX_PRODUCTS = 10;

    public Order() {
        this.idOrder = ++ordersCounter;
        //Instanciate the array of products
        products = new Product[MAX_PRODUCTS];
    }

    public void addProduct(Product product) {
        //Add the new products if it is possible
        if (productsCounter < MAX_PRODUCTS) {
            //Add the new product and increment the productsCounter
            products[productsCounter++] = product;
        }
        else{
            System.out.println("The maximum of products has been exceeded:: " + MAX_PRODUCTS);
        }
    }
```

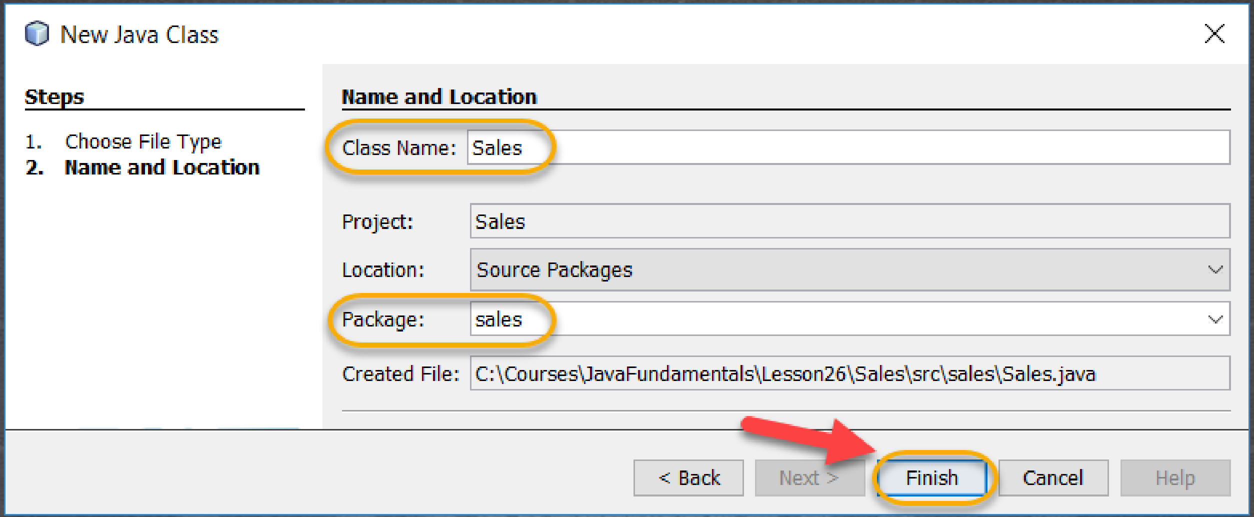
5. MODIFY THE CODE

Order.java:

```
    public double calculateTotal() {
        double total = 0;
        for (int i = 0; i < productsCounter; i++) {
            total += products[i].getPrice();
        }
        return total;
    }

    public void showOrder() {
        System.out.println("Order #:" + idOrder);
        System.out.println("Total of the order $" + calculateTotal());
        System.out.println("Productos in the order:" + productsCounter);
        for (int i = 0; i < productsCounter; i++) {
            System.out.println(products[i]);
        }
    }
}
```

PASO 6. CREATE A CLASS

Create the Sales class:



Class Name: Sales

Project: Sales

Location: Source Packages

Package: sales

Created File: C:\Courses\JavaFundamentals\Lesson26\Sales\src\sales\Sales.java

PASO 7. MODIFY THE CODE

Sales.java:

```java
import com.gm.sales.*;//Import all the classes of this package

public class Sales {

    public static void main(String[] args) {
        //create several Product objects
        Product p1 = new Product("T-Shirt",55.00);
        Product p2 = new Product("Cap",30.00);
        Product p3 = new Product("Jeans",150.00);

        //create an Order object
        Order order1 = new Order();

        //Add products to Order 1
        order1.addProduct(p1);
        order1.addProduct(p2);
        order1.addProduct(p3);

        //Print the first order
        order1.showOrder();
```

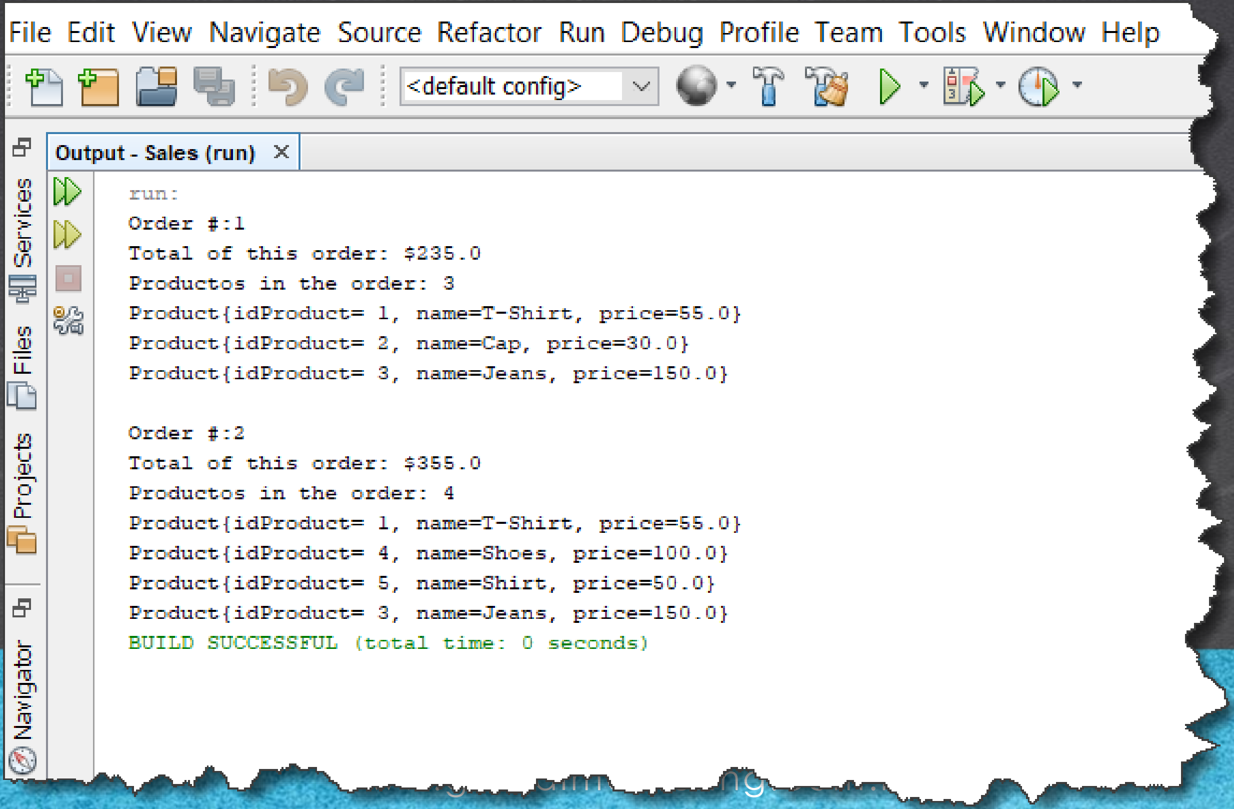
PASO 7. MODIFY THE CODE

Sales.java:

```
        //Create a second order
        Order order2 = new Order();

        //Create new products
        Product p4 = new Product("Shoes", 100);
        Product p5 = new Product("Shirt", 50);

        //Add products to order 2
        order2.addProduct(p1);
        order2.addProduct(p4);
        order2.addProduct(p5);
        order2.addProduct(p3);

        //Print the order 2
        System.out.println("");
        order2.showOrder();
    }
}
```

8. EXECUTE THE PROJECT

The result is as follows:


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - Sales (run)
Services
Files
Projects
Navigator

```
run:
Order #:1
Total of this order: $235.0
Productos in the order: 3
Product{idProduct= 1, name=T-Shirt, price=55.0}
Product{idProduct= 2, name=Cap, price=30.0}
Product{idProduct= 3, name=Jeans, price=150.0}

Order #:2
Total of this order: $355.0
Productos in the order: 4
Product{idProduct= 1, name=T-Shirt, price=55.0}
Product{idProduct= 4, name=Shoes, price=100.0}
Product{idProduct= 5, name=Shirt, price=50.0}
Product{idProduct= 3, name=Jeans, price=150.0}
BUILD SUCCESSFUL (total time: 0 seconds)
```

EXERCISE CONCLUSION

- With this exercise we have put into practice the concept of class design in Java, and how to lean on the relationship between them in order to start creating increasingly complex systems.

- This topic is just the beginning of what we will study in later courses, but the important thing is to have the bases of Java to continue advancing with firm steps in the learning of this fabulous programming language.

- With this lesson we conclude the Java Fundamentals Course and invite you to follow your Java specialization and study the following course: Programming with Java. We wait for you. 🙂

ONLINE COURSE

JAVA FUNDAMENTALS

By: Eng. Ubaldo Acosta

Global Mentoring

Experiencia y Conocimiento para tu vida