Lab 12

Part 1

Design and implement a webshop framework with the following list of requirements:

Product catalog
You can add, remove and get products from the product catalog.

ShoppingCart
You can add products to the shoppingcart, you can remove products from the shoppingcart, and you can update the quantity of products in the shoppingcart. Assume that every shoppingCart has an unique id.
You can get the content of the shoppingCart

If you checkout the shoppingcart, an order is created based on the content of the shoppingcart.

Order
You can create an order based on the content of the shoppingCart. Once an order is created, you can add payment information (amount, date, creditcard number, validation date, creditcard type), customer information (name, email, phone), and the shipping address and billing address (street, city, zip) to the order. Every order has an unique orderNumber. The framework only supports credit card payments, but it should be easy to plugin other type of payments. The billing and shipping addresses are domestic (street, city, zip)

You can get the content of the order

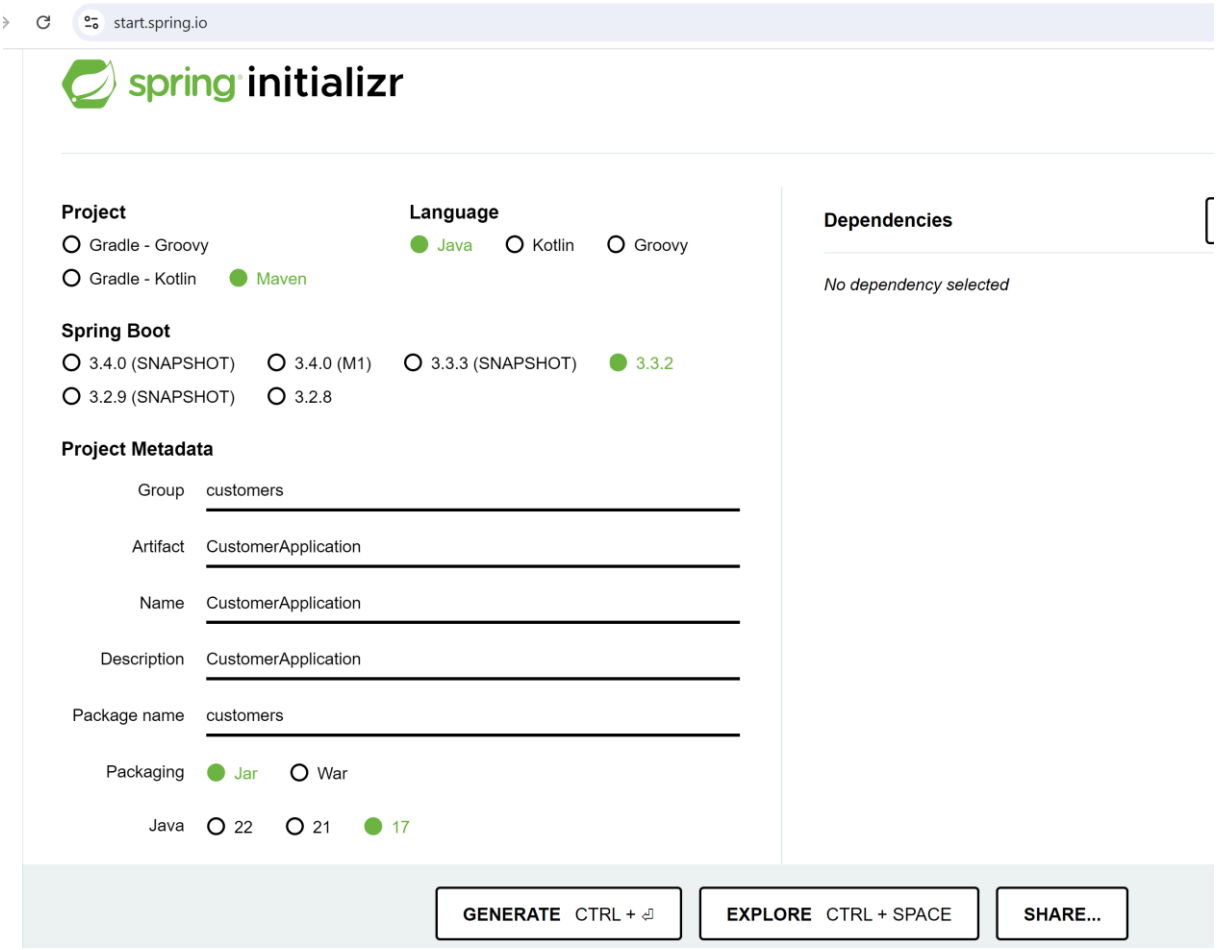
If you place an order, an email is send to the particular customer.

All data is stored in the database. The DAO classes just keeps an in-memory list of all the objects (orders, shoppingcarts, products, …)

a. Draw the class diagram of your design of the framework
b. Implement the framework in Java. Package the framework as a jar file
c. Write a simple application that uses the framework. This application should also support PayPal payments. This application also supports international addresses (within a certain country)

Part 2

In the browser go to https://start.spring.io/

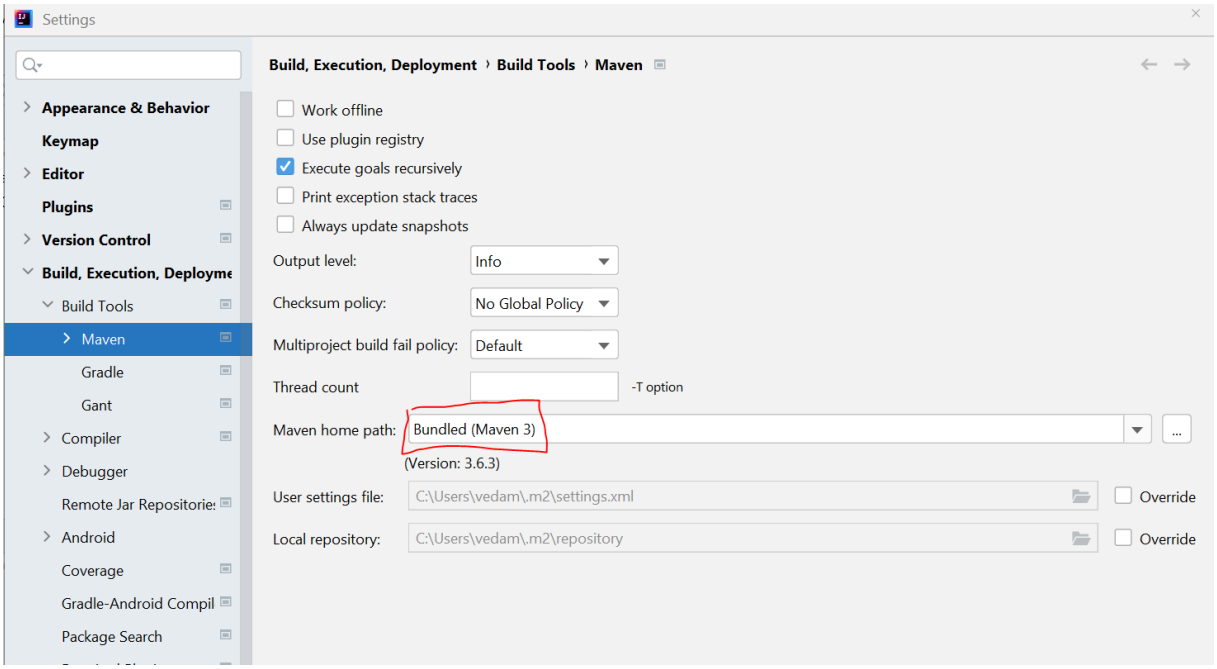


Fill in the fields as shown above. Click the **Generate** button.

Unzip the content of this file and open the project in IntelliJ

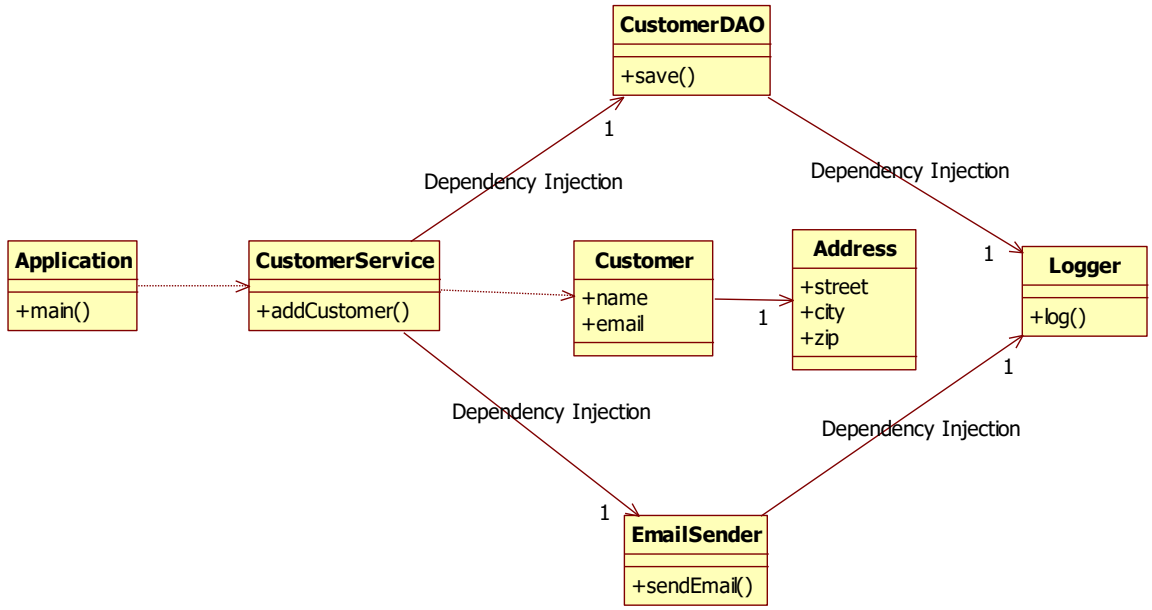
Make sure that the maven configuration in IntelliJ is correct.

In IntelliJ, select **File->Setting**

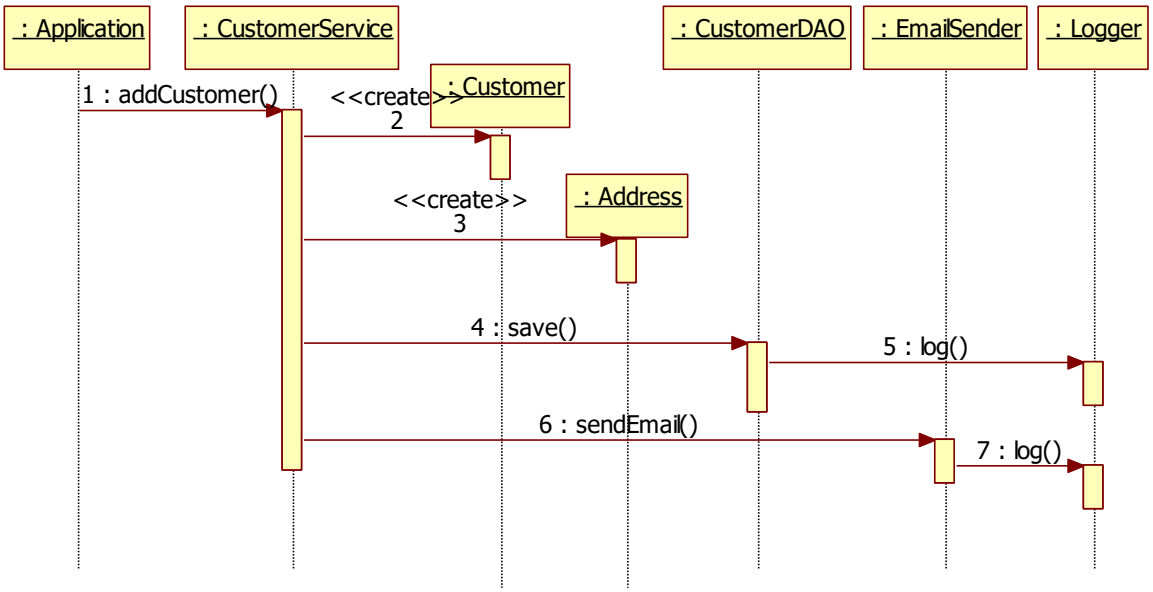


Select **Bundled (Maven 3)** for **Maven home path**

Now implement the following application in Spring boot:



The CustomerService, CustomerDAO, EmailSender and Logger are Spring beans. Wire them together with dependency injection. Use a different type of DI for every relation.



Test if your application works.

Part 3

Modify the application from part 2 as follows:

Write a CustomerDAOMock class that we want to use in our test environment. In our production environment we want to use the existing CustomerDAO class. Use profiles so that we can configure in application.properties which CustomerDAO we want to inject.

Part 4

Modify the application from part 3 as follows:

Whenever we add a new Customer add the following functionality:
If the Customer is from Chicago, print 2 asterisks in the console
If the Customer is from Fairfield, print 3 asterisks in the console
If the Customer is from Iowa City, print 4 asterisks in the console
If the Customer is not from any of the cities above, print 10 asterisks in the console

Make sure that this behavior can be easily extended with more cities, and that it also can be changed easily.