INTRODUCTION

Overview

This **InOrder** program is an e-commerce program that enables three functions of a business.

- Managing products: Manage information about products that can be sold to customers.
- Tracking inventory: Track current inventories of products
- Processing orders: Process orders for products from customers.

Our teamed develop this backend program **with user interface**. We strictly followed the OOD rules and designed our projects properly by separating different modules into different directory. The basic operations in our program includes: DDL for creating the tables, DML for adding entries in the tables, and DQL for making commonly used queries to retrieve product, inventory, and order information from the database.

You will find some brief information in this introduction, such as the **Tools and the Environment**, **the Structure of our project** and **how to Run our project step by step**. For other detailed information, you can go to the documentation folder to find the answer.

Team Members

Our Team consists of four members: Wenxuan Guo, Lei Cao, Suyue Jiang and Yibao Hu.

Tools and Environment

The basic tools and environment we were using:

- Project Building: Java, Apache Derby
- Project Testing: JUnit5
- Project IDE: IntelliJ

Project Structure

There are three main parts in this repository: **Documentation**, **Code** and **Tests.**

- Documentation: Contains 4 parts to describe our project: Projects Functionality, Project design, Project API, Test plan and also the Javadoc of the project.

- Code: Includes all of the code that implements our project
- Tests: Includes all of the tests that can be run to verify the functionality of the project.

You can find a document that describes the purpose of each folder and the files and directories that it contains. Also, we generated the Javadoc of the program and it is located in the documentation folder. You can start reading it from index.html and it can be read offline.

How to Run Our Project?

- **Get IntelliJ IDE:**

We use IntelliJ as our IDE. You can get IntelliJ through: https://www.jetbrains.com/idea/. The community version is free and open to everyone and the unlimited version can be used by verifying your university email address.

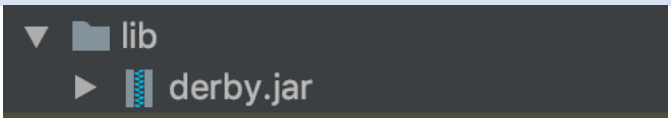
- **Import and set up the project:**

After importing the project into IntelliJ, you may see some errors regarding JUnit5 in the test files, this is because you are not installing Junit5. Click on the red bulb and it will assist you to add JUnit5 to the build path.

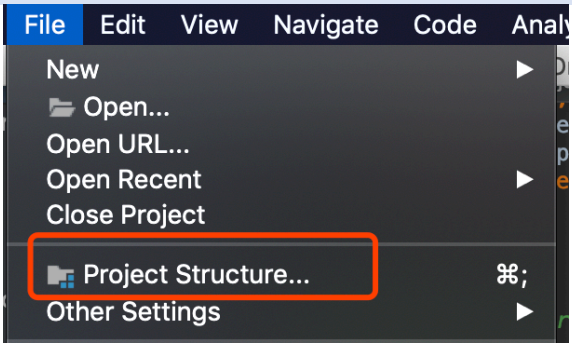
- **Configure derby**

There are many ways to configure derby in IntelliJ. One way we are using is to:

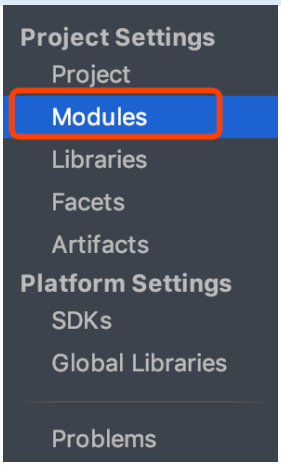
1. Create a new repository named 'lib' into the project
2. Paste "derby.jar" into "lib"

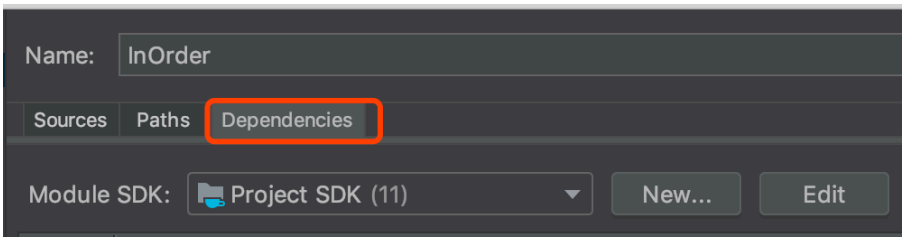


3. Click on File -> Project Structure.

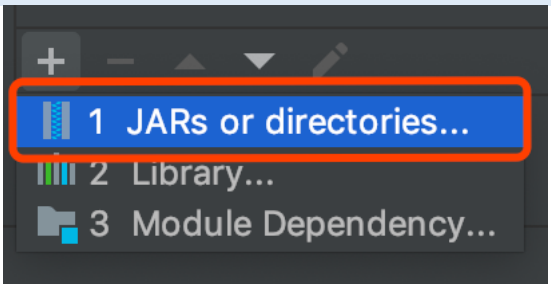


4.Click on Modules->Dependencies

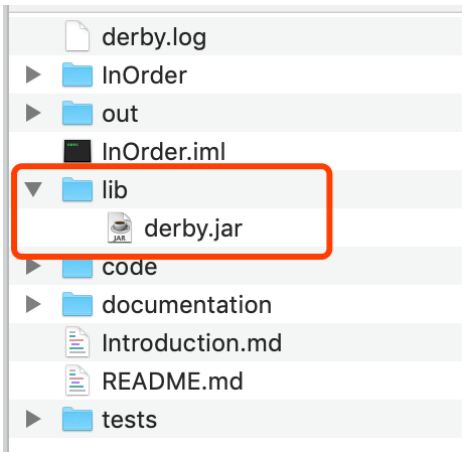




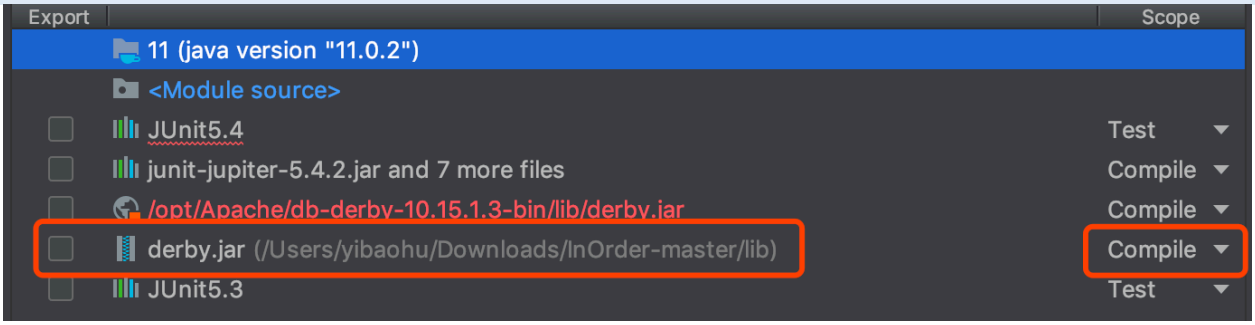
5. Click on the "+" button and choose "JARs or directories…"



6. Choose "derby.jar" which we just put in the "lib" directory



7.You will find "derby.jar" has been added to the list and make sure the Scope is "Compile" and click Ok or apply



- **Run the program**

This is a backend program **with user interface**. Basically, you can do three things in the program. The first one is adding and checking information through the database by using user interface.

By using UI:

First you need to run the InOrder.java located in the "code" folder. Then you will see a menu in the terminal like this:

```
Welcome to the how to invest for dummies!
Main Menu
What do you want to do?
Please enter the index of a command or enter q or quit to quit:
1 Add a new product.
2 Examine all product.
3 Add an inventory record.
4 Examine all inventory records.
5 Add a customer.
6 Examine all customers.
7 Place an order all customers.
8 Examine an order.
```

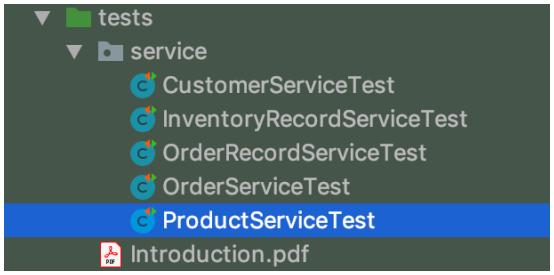
Then you can choose which operation you want to do with the database, for example, if you enter 1, then the terminal will show the next step asking you to enter information of a new product:

Just follow the steps and you will add a new product to the database. You can return to the main menu to do other operations according to your needs.

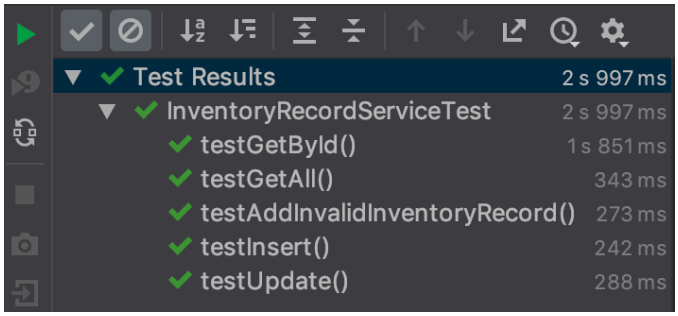
The second thing you can do is to run the tests.

By running tests:

You can go to the "test" folder and find different tests that exam whether our services files work fine. Our test plan is located in the /documentation/Test Plan.pdf, you can find a detailed description of the purpose of each test.



We use Junit5 for our tests, if you hit run, you will see results at the button of your IDE, it will tell you how many tests that have passed.



The last thing you can do is to add data to the database by using the APIs.

By using APIs:

You can call APIs in each services file. You can find detailed description of the APIs is the **/documentation/Project API.pdf** and go from there. For example, you can add a product "apple" into the database by doing this:

1.connect to the database:

```
Connection conn = null;
try {
    InOrderModel driver = new InOrderModel();
    conn = driver.init();
} catch (SQLException e) {
    System.out.println("No Database Connection");
}
```

2.define a new product you want to add to the database:

```
Product p0 = new Product( name: "apple",  description: "apple",  sku: "AB-123456-0N");
```

3.Use the insert function defined in the ProductService.java

```
ProductService.insert(conn, p0);
```

4.Get the product apple by Sku:

```
Product p = ProductService.getProductBySku(conn,  sku: "AB-123456-0N");
System.out.println(p.getDescription());
```