Inventory Management System (IMS) Project

BY CELINA BASA

Introduction

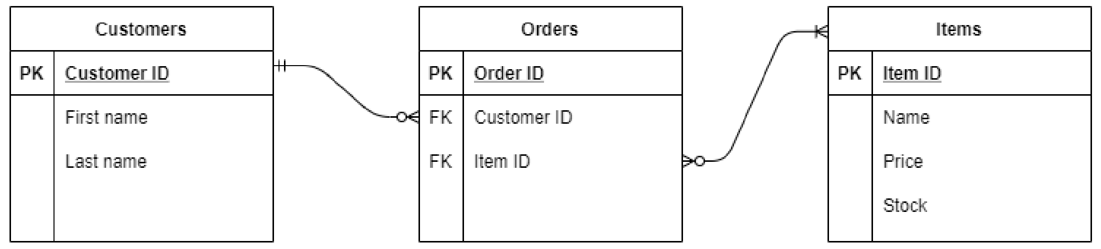
Who am I?

- QA Trainee Consultant

My approach to the project:

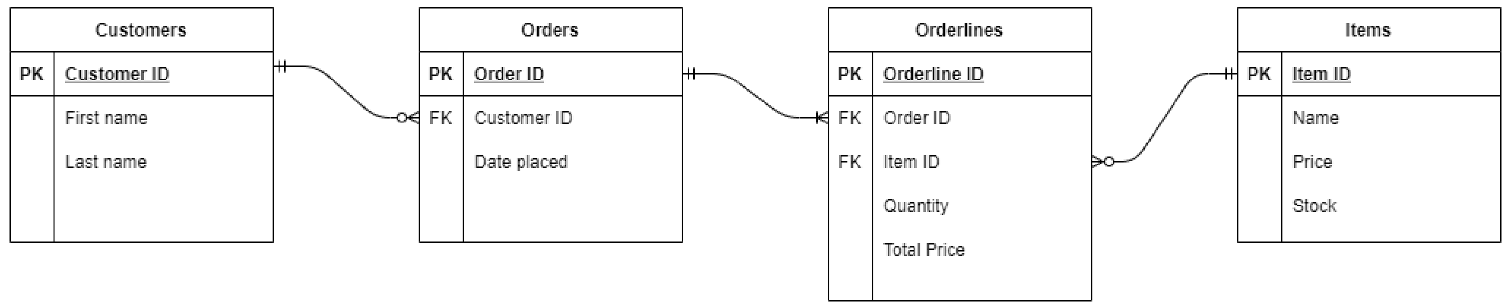
- Read through the specification
- Planned tasks in order of priority
- Made adjustments to code after getting errors

First draft of ERD:



Since there is a many to many relationship between the orders and items table and MySQL does not support this, an intermediary table, orderlines, will be used to handle this relationship.

Final ERD:



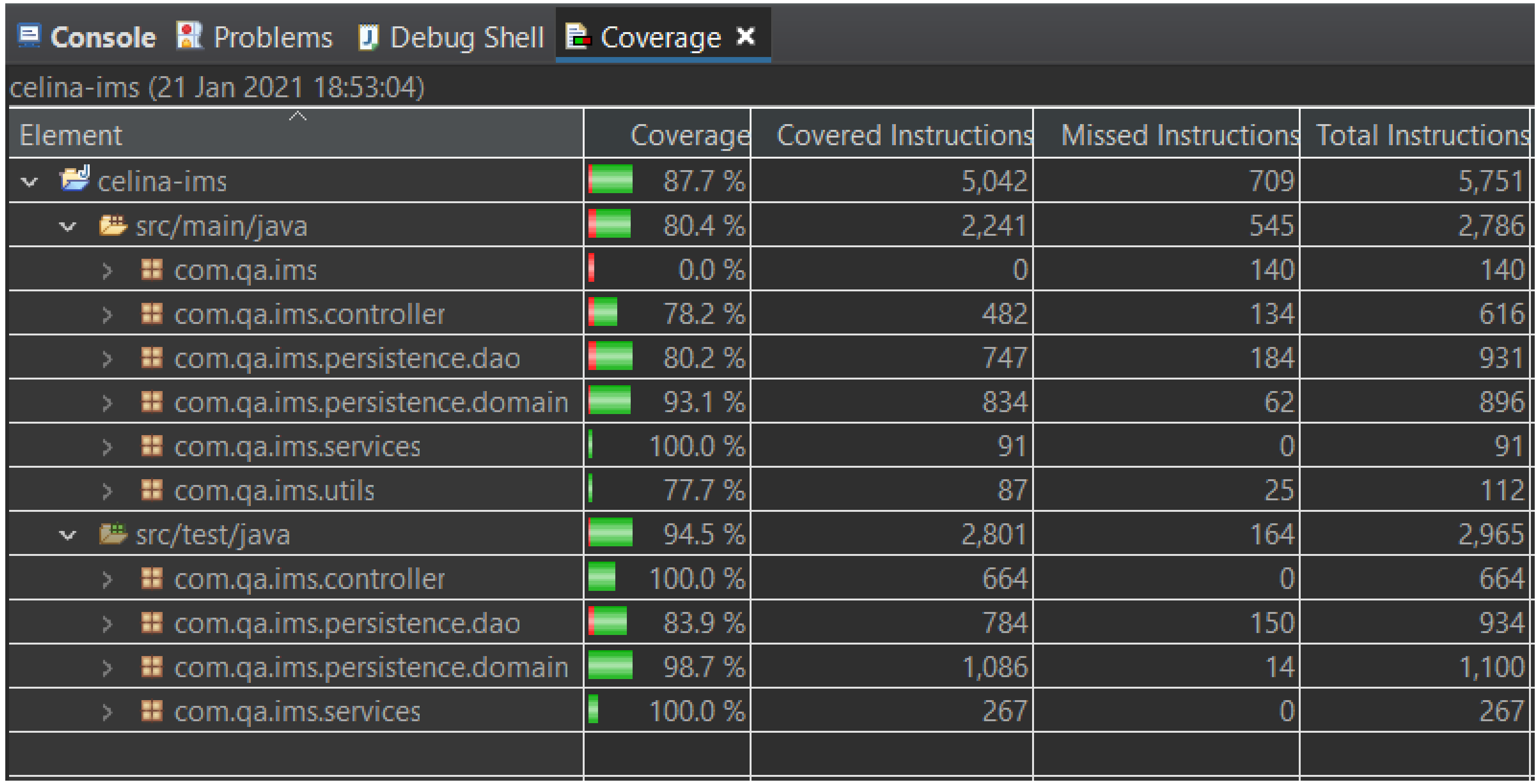
Consultant Journey

What technologies did you learn for this project?

- Git
- MySQL
- Google Cloud Platform
- Java
- Maven
- Jira

Testing

What was tested?



celina-ims (21 Jan 2021 18:53:04)

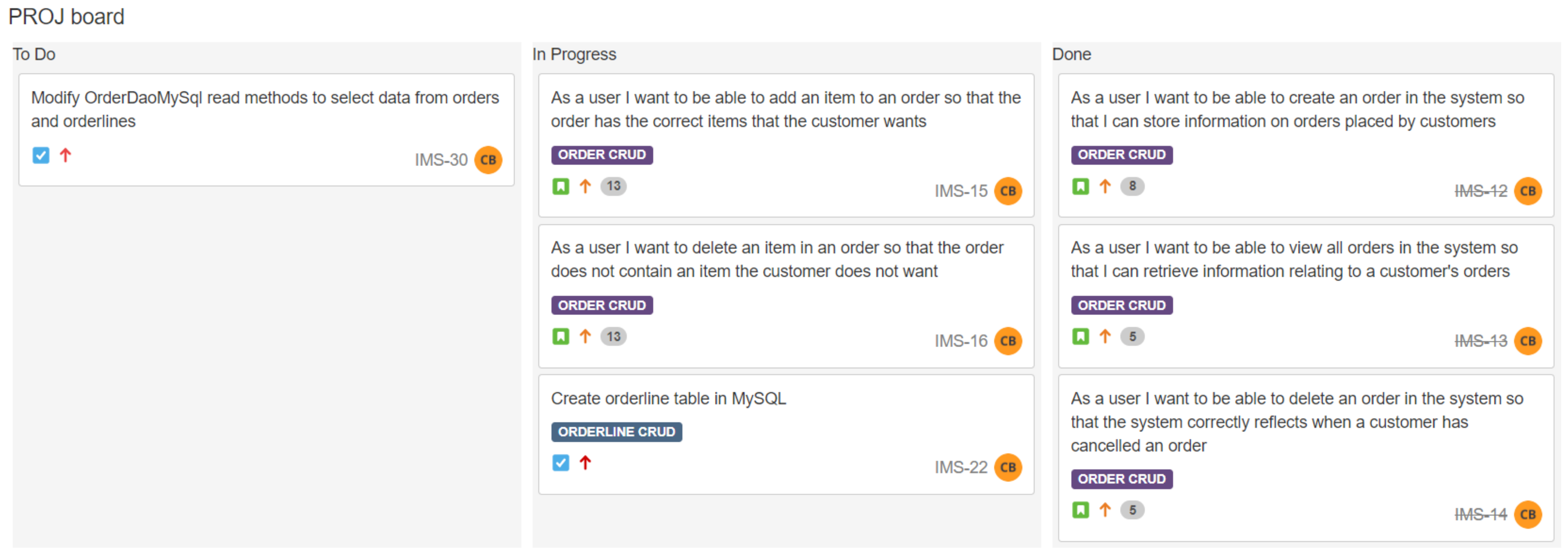
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
celina-ims	87.7 %	5,042	709	5,751
src/main/java	80.4 %	2,241	545	2,786
com.qa.ims	0.0 %	0	140	140
com.qa.ims.controller	78.2 %	482	134	616
com.qa.ims.persistence.dao	80.2 %	747	184	931
com.qa.ims.persistence.domain	93.1 %	834	62	896
com.qa.ims.services	100.0 %	91	0	91
com.qa.ims.utils	77.7 %	87	25	112
src/test/java	94.5 %	2,801	164	2,965
com.qa.ims.controller	100.0 %	664	0	664
com.qa.ims.persistence.dao	83.9 %	784	150	934
com.qa.ims.persistence.domain	98.7 %	1,086	14	1,100
com.qa.ims.services	100.0 %	267	0	267

Demonstration: Example user stories

User Story	Story points
As a user I want to be able to view all orders in the system so that I can retrieve information relating to a customer's orders	8
As a user I want to be able to create an order in the system so that I can store information on orders placed by customers	13
As a user I want to be able to add an item to an order so that the order has the correct items that the customer wants	13
As a user I want to be able to calculate a cost for an order so that I can see how much the customer has paid	8

Sprint review

First week:



PROJ board

To Do

Modify OrderDaoMySql read methods to select data from orders and orderlines
IMS-30 CB

In Progress

As a user I want to be able to add an item to an order so that the order has the correct items that the customer wants
ORDER CRUD
13 IMS-15 CB

As a user I want to delete an item in an order so that the order does not contain an item the customer does not want
ORDER CRUD
13 IMS-16 CB

Create orderline table in MySQL
ORDERLINE CRUD
IMS-22 CB

Done

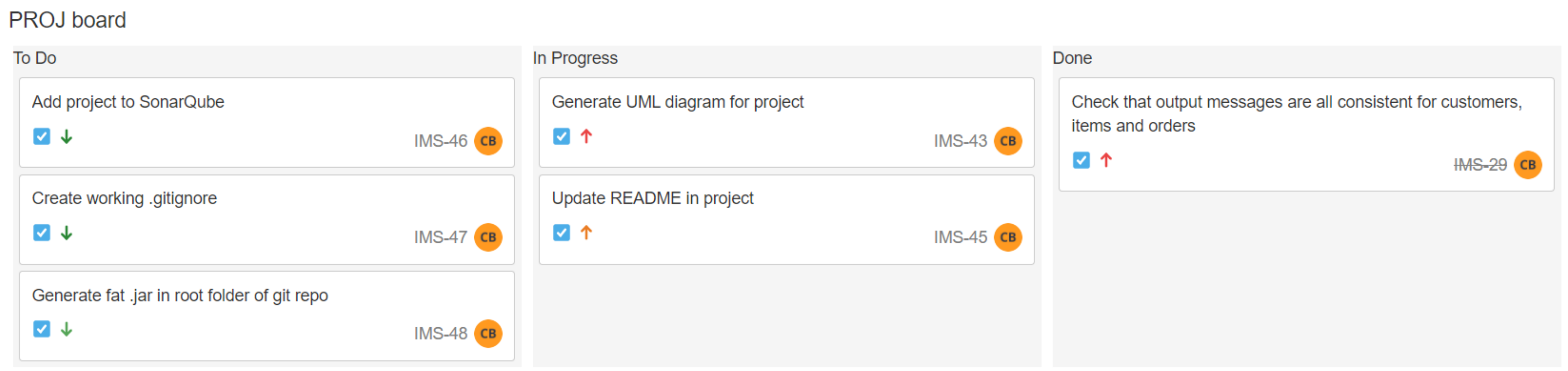
As a user I want to be able to create an order in the system so that I can store information on orders placed by customers
ORDER CRUD
8 ~~IMS-12~~ CB

As a user I want to be able to view all orders in the system so that I can retrieve information relating to a customer's orders
ORDER CRUD
5 ~~IMS-13~~ CB

As a user I want to be able to delete an order in the system so that the system correctly reflects when a customer has cancelled an order
ORDER CRUD
5 ~~IMS-14~~ CB

Sprint review

Final week:



PROJ board

To Do

Add project to SonarQube
IMS-46 CB

Create working .gitignore
IMS-47 CB

Generate fat .jar in root folder of git repo
IMS-48 CB

In Progress

Generate UML diagram for project
IMS-43 CB

Update README in project
IMS-45 CB

Done

Check that output messages are all consistent for customers, items and orders
~~IMS-29~~ CB

Sprint review

What was completed:

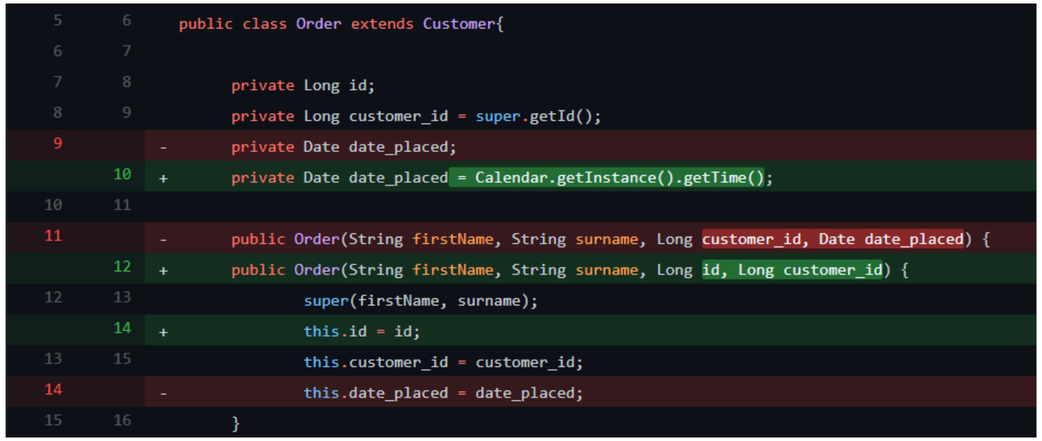
- CRUD functionality for customers, items and orders
- Project connects to GCP-based MySQL instance
- Test coverage of 80% reached
- Fat .jar able to run from the command-line
- Supporting documentation

What was left behind:

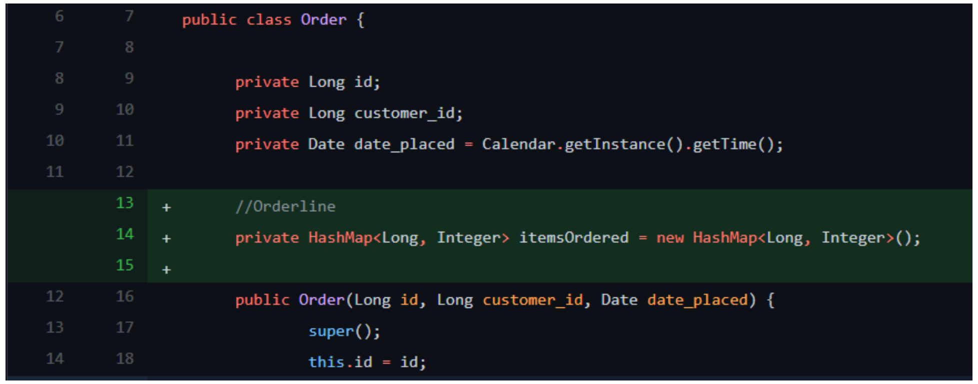
- Unit tests for IMS and Runner class
- Using SonarQube

Sprint retrospective

Order domain (BEFORE):

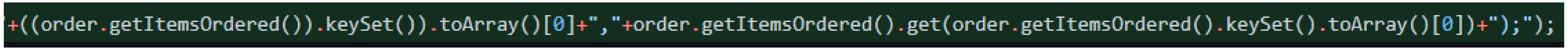


```
  public class Order extends Customer{

          private Long id;
          private Long customer_id = super.getId();
-         private Date date_placed;
+         private Date date_placed = Calendar.getInstance().getTime();

-         public Order(String firstName, String surname, Long customer_id, Date date_placed) {
+         public Order(String firstName, String surname, Long id, Long customer_id) {
                  super(firstName, surname);
+                 this.id = id;
                  this.customer_id = customer_id;
-                 this.date_placed = date_placed;
          }
```



```
  public class Order {

          private Long id;
          private Long customer_id;
          private Date date_placed = Calendar.getInstance().getTime();

+         //Orderline
+         private HashMap<Long, Integer> itemsOrdered = new HashMap<Long, Integer>();
+
          public Order(Long id, Long customer_id, Date date_placed) {
                  super();
                  this.id = id;
```

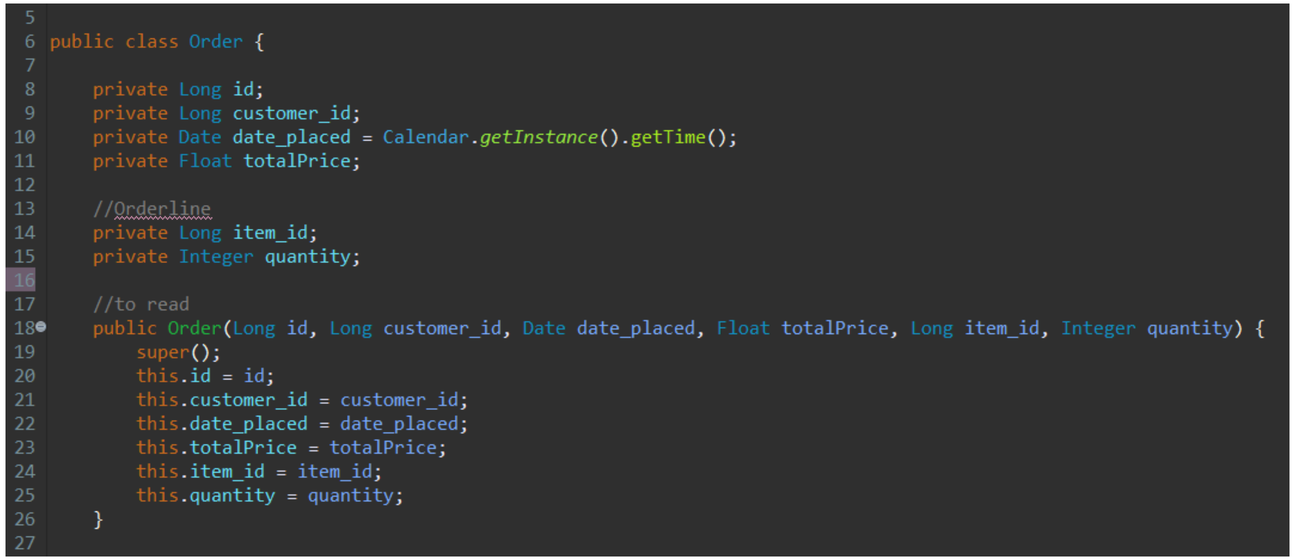
OrderDaoMysql create method (BEFORE):



```
'+((order.getItemsOrdered()).keySet()).toArray()[0]+","+order.getItemsOrdered().get(order.getItemsOrdered().keySet().toArray()[0])+");");
```

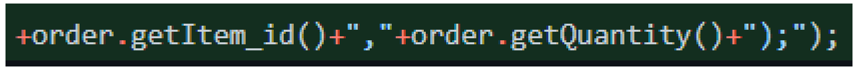
Sprint retrospective

Order domain (AFTER):



```

public class Order {

    private Long id;
    private Long customer_id;
    private Date date_placed = Calendar.getInstance().getTime();
    private Float totalPrice;

    //Orderline
    private Long item_id;
    private Integer quantity;

    //to read
    public Order(Long id, Long customer_id, Date date_placed, Float totalPrice, Long item_id, Integer quantity) {
        super();
        this.id = id;
        this.customer_id = customer_id;
        this.date_placed = date_placed;
        this.totalPrice = totalPrice;
        this.item_id = item_id;
        this.quantity = quantity;
    }

```

OrderDaoMysql create method (AFTER):



```
+order.getItem_id()+","+order.getQuantity()+");");
```

Conclusion

Reflections on the project

Future steps

Thank you for listening, please ask any questions you have!