# Creating an Inventory Management System (IMS)

David Barringer

# Introduction

The Team:

- David Barringer - QA Academy Trainee

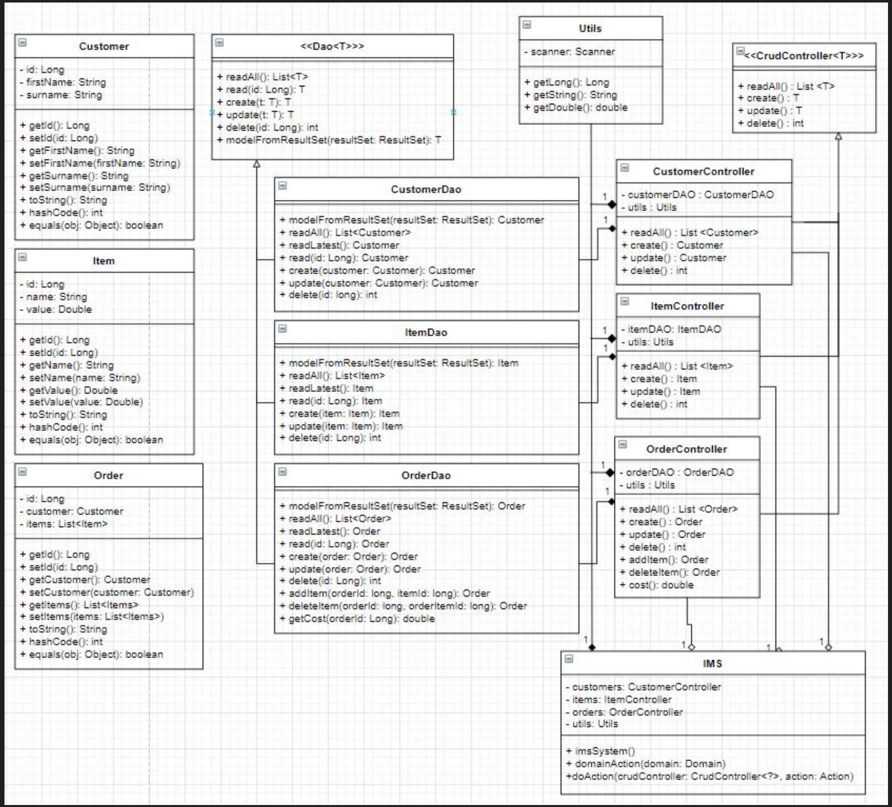
The Task:

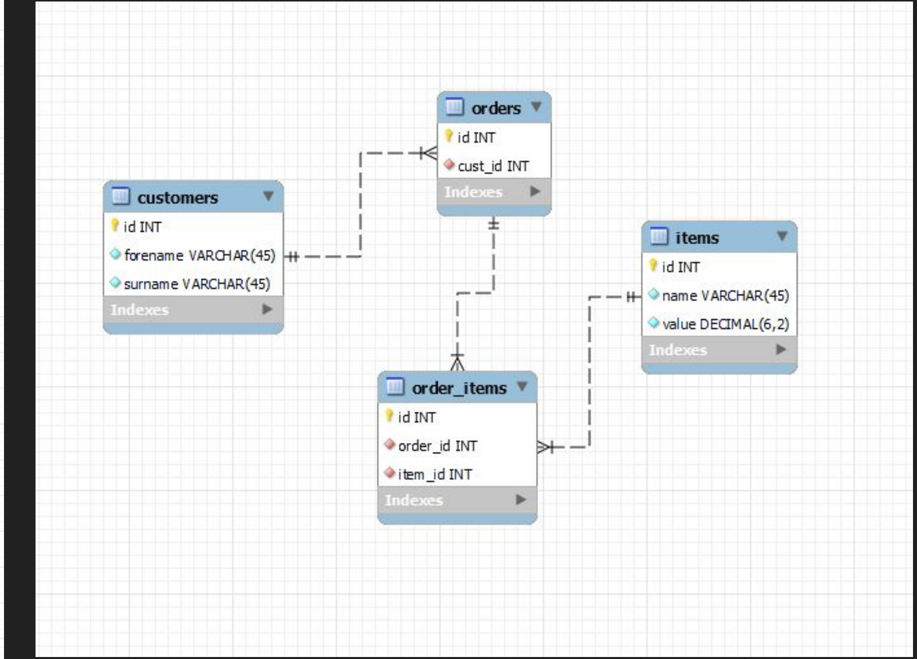
- Create an Inventory Management System
- CRUD functionality for customers, items and orders
- Allow items to be added/removed from orders
- Stored in a database

# Requirement Analysis

- Neither specification or template suggested handling full CRUD for order items
- Decision was to handle items as a part of the order in the interface
- User can see order in full when creating
- All order database actions handled at the same time
- Command line interface ⇒ inputs kept simple

# Initial UML diagram and ERD





# Technologies Used

- Management - Jira
- Version Control - git
- Testing - JUnit, Mockito
- Building - Maven

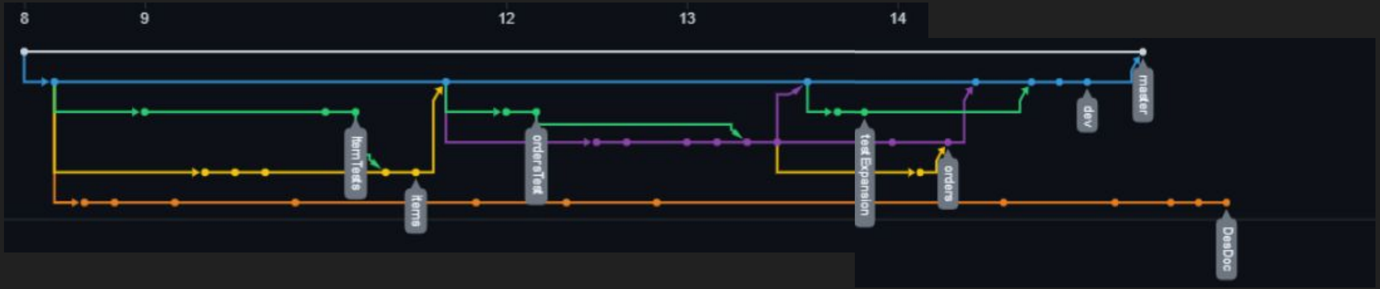
# CI - Version Control

Project developed using feature-branch model

Components and their tests were written on separate branches

Merged to dev when unit tests pass

Merged to main on completing a version

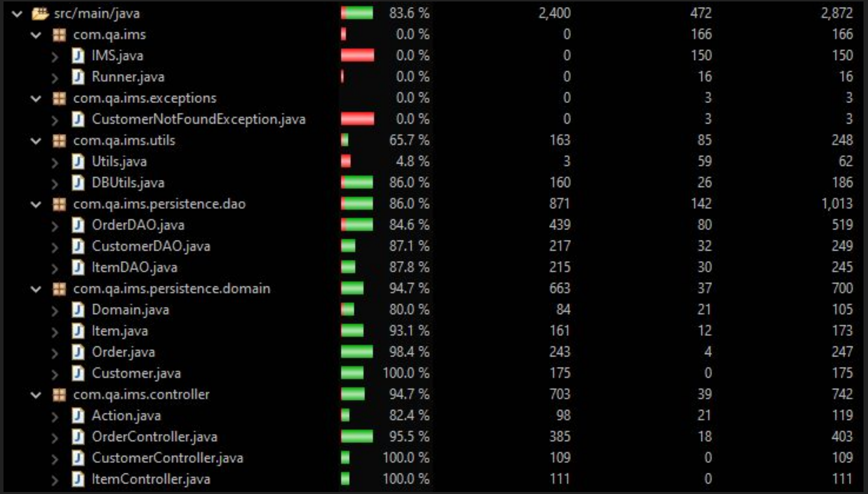


# Testing

Unit tests were written using JUnit and Mockito

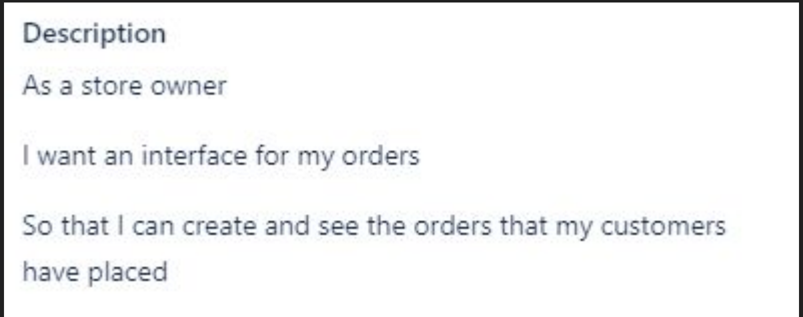
Using TDD, tests were written first, using the UML plan

Target was 80% code coverage



| | | | | |
|---|---|---|---|---|
| src/main/java | 83.6 % | 2,400 | 472 | 2,872 |
| com.qa.ims | 0.0 % | 0 | 166 | 166 |
| IMS.java | 0.0 % | 0 | 150 | 150 |
| Runner.java | 0.0 % | 0 | 16 | 16 |
| com.qa.ims.exceptions | 0.0 % | 0 | 3 | 3 |
| CustomerNotFoundException.java | 0.0 % | 0 | 3 | 3 |
| com.qa.ims.utils | 65.7 % | 163 | 85 | 248 |
| Utils.java | 4.8 % | 3 | 59 | 62 |
| DBUtils.java | 86.0 % | 160 | 26 | 186 |
| com.qa.ims.persistence.dao | 86.0 % | 871 | 142 | 1,013 |
| OrderDAO.java | 84.6 % | 439 | 80 | 519 |
| CustomerDAO.java | 87.1 % | 217 | 32 | 249 |
| ItemDAO.java | 87.8 % | 215 | 30 | 245 |
| com.qa.ims.persistence.domain | 94.7 % | 663 | 37 | 700 |
| Domain.java | 80.0 % | 84 | 21 | 105 |
| Item.java | 93.1 % | 161 | 12 | 173 |
| Order.java | 98.4 % | 243 | 4 | 247 |
| Customer.java | 100.0 % | 175 | 0 | 175 |
| com.qa.ims.controller | 94.7 % | 703 | 39 | 742 |
| Action.java | 82.4 % | 98 | 21 | 119 |
| OrderController.java | 95.5 % | 385 | 18 | 403 |
| CustomerController.java | 100.0 % | 109 | 0 | 109 |
| ItemController.java | 100.0 % | 111 | 0 | 111 |

# Demonstration

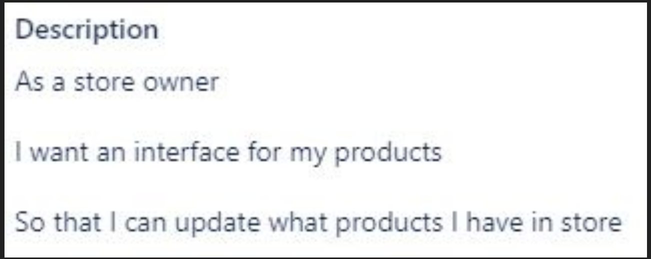


**Description**

As a store owner

I want an interface for my orders

So that I can create and see the orders that my customers have placed



**Description**

As a store owner

I want an interface for my products

So that I can update what products I have in store

# Sprint Review

What was completed:

- Plan: UML diagram, ERD and User Stories
- CRUD for customers, items and orders
- Command line interface for user input
- Unit tests with > 80% code coverage

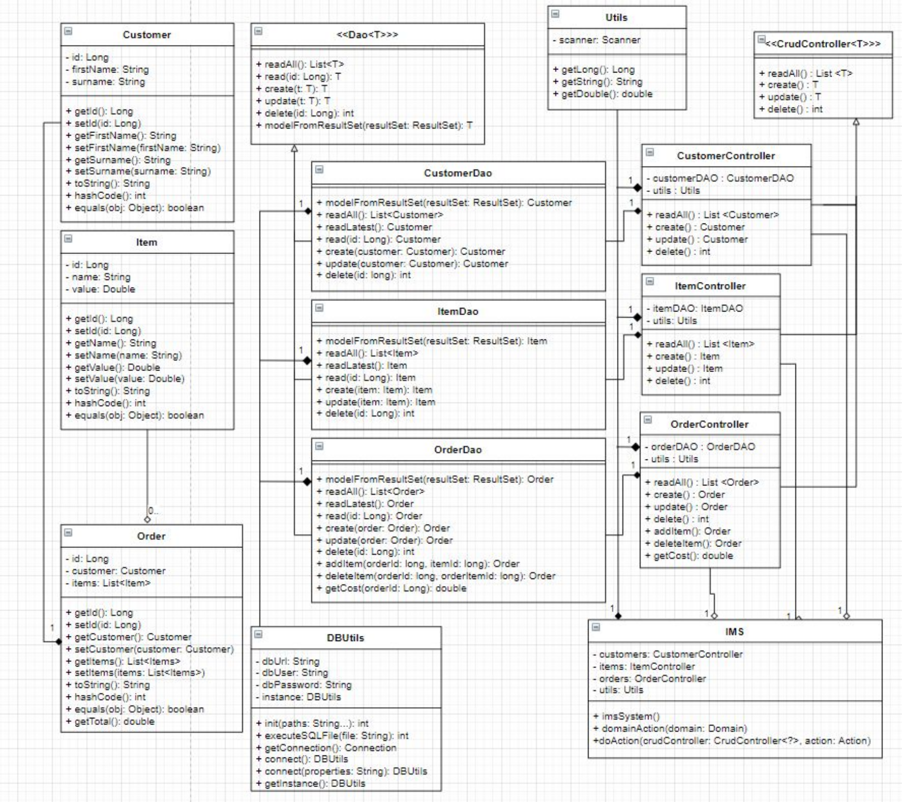
What wasn’t completed:

- No points of the MVP were missed
- Some stretch goals were considered, but not achieved

# Sprint Retrospective

- Completed all of the tasks initially set
- UML diagram required minimal changing from initial plan
- Some further tasks were added and completed
- Efficiency was greater during development compared to planning/documentation
- More effective planning could allow for stretch goals to be met

# Final UML diagram



Customer

- id: Long
- firstName: String
- surname: String

+ getId(): Long
+ setId(id: Long)
+ getFirstName(): String
+ setFirstName(firstName: String)
+ getSurname(): String
+ setSurname(surname: String)
+ toString(): String
+ hashCode(): int
+ equals(obj: Object): boolean

Item

- id: Long
- name: String
- value: Double

+ getId(): Long
+ setId(id: Long)
+ getName(): String
+ setName(name: String)
+ getValue(): Double
+ setValue(value: Double)
+ toString(): String
+ hashCode(): int
+ equals(obj: Object): boolean

Order

- id: Long
- customer: Customer
- items: List<Item>

+ getId(): Long
+ setId(id: Long)
+ getCustomer(): Customer
+ setCustomer(customer: Customer)
+ getItems(): List<Items>
+ setItems(items: List<Items>)
+ toString(): String
+ hashCode(): int
+ equals(obj: Object): boolean
+ getTotal(): double

<<Dao<T>>>

+ readAll(): List<T>
+ read(id: Long): T
+ create(t: T): T
+ update(t: T): T
+ delete(id: Long): int
+ modelFromResultSet(resultSet: ResultSet): T

CustomerDao

+ modelFromResultSet(resultSet: ResultSet): Customer
+ readAll(): List<Customer>
+ readLatest(): Customer
+ read(id: Long): Customer
+ create(customer: Customer): Customer
+ update(customer: Customer): Customer
+ delete(id: long): int

ItemDao

+ modelFromResultSet(resultSet: ResultSet): Item
+ readAll(): List<Item>
+ readLatest(): Item
+ read(id: Long): Item
+ create(item: Item): Item
+ update(item: Item): Item
+ delete(id: Long): int

OrderDao

+ modelFromResultSet(resultSet: ResultSet): Order
+ readAll(): List<Order>
+ readLatest(): Order
+ read(id: Long): Order
+ create(order: Order): Order
+ update(order: Order): Order
+ delete(id: Long): int
+ addItem(orderId: long, itemId: long): Order
+ deleteItem(orderId: long, orderItemId: long): Order
+ getCost(orderId: Long): double

DBUtils

- dbUrl: String
- dbUser: String
- dbPassword: String
- instance: DBUtils

+ init(paths: String...): int
+ executeSQLFile(file: String): int
+ getConnection(): Connection
+ connect(): DBUtils
+ connect(properties: String): DBUtils
+ getInstance(): DBUtils

Utils

- scanner: Scanner

+ getLong(): Long
+ getString(): String
+ getDouble(): double

<<CrudController<T>>>

+ readAll() : List <T>
+ create() : T
+ update() : T
+ delete() : int

CustomerController

- customerDAO : CustomerDAO
- utils : Utils

+ readAll() : List <Customer>
+ create() : Customer
+ update() : Customer
+ delete() : int

ItemController

- itemDAO: ItemDAO
- utils: Utils

+ readAll() : List <Item>
+ create() : Item
+ update() : Item
+ delete() : int

OrderController

- orderDAO : OrderDAO
- utils : Utils

+ readAll() : List <Order>
+ create() : Order
+ update() : Order
+ delete() : int
+ addItem(): Order
+ deleteItem(): Order
+ getCost(): double

IMS

- customers: CustomerController
- items: ItemController
- orders: OrderController
- utils: Utils

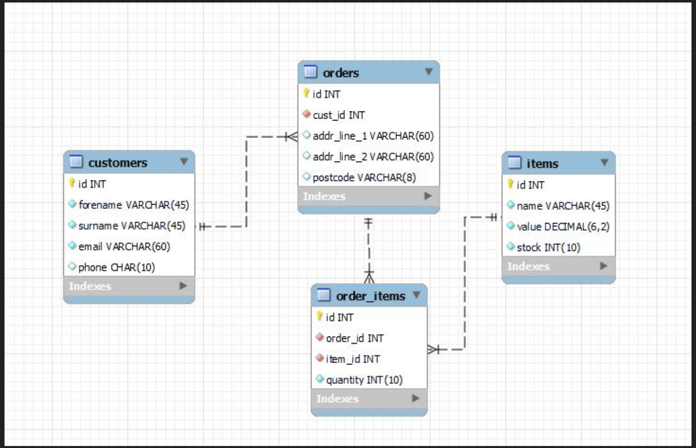
+ imsSystem()
+ domainAction(domain: Domain)
+doAction(crudController: CrudController<?>, action: Action)

# Conclusion

Project meets the requirements for the MVP

Further developments:

- Add quantities of items to orders
- Add more general info to tables



orders
id INT
cust_id INT
addr_line_1 VARCHAR(60)
addr_line_2 VARCHAR(60)
postcode VARCHAR(8)
Indexes

customers
id INT
forename VARCHAR(45)
surname VARCHAR(45)
email VARCHAR(60)
phone CHAR(10)
Indexes

items
id INT
name VARCHAR(45)
value DECIMAL(6,2)
stock INT(10)
Indexes

order_items
id INT
order_id INT
item_id INT
quantity INT(10)
Indexes

# Questions