INVENTORY
MANAGEMENT
SYSTEM

HELLO WORLD!

I AM Gie-Anne Fortuna

"The expert at anything was once a beginner"

TECHNOLOGIES

VERSION CONTROL SYSTEM: **GIT**

SOURCE CODE MANAGEMENT: **GITHUB**

PROJECT MANAGEMENT: **JIRA**

DATABASE MANAGEMENT: **MYSQL**

BACKEND: **JAVA**

BUILD TOOL: **MAVEN**

UNIT TESTING: **JUNIT**

PLANNING

PLANNING:
RISK ASSESSMENT

EQUIPMENT ISSUES

SOFTWARE COMPATIBILITY ISSUES

JAR FILE COMPILATION WITH DEPENDENCIES
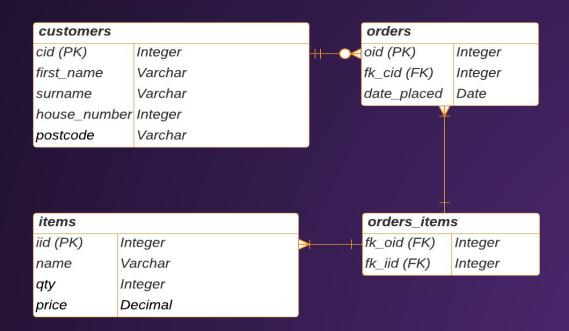
GITHUB: MERGE CONFLICTS

INSUFFICIENT UNDERSTANDING OF STARTER CODE

PUNCTUALITY TO DEADLINE

PLANNING:
ENTITY RELATIONSHIP DIAGRAM

PLANNING:
SQL SCHEMA

```
drop schema ims;
CREATE SCHEMA IF NOT EXISTS `ims`;
USE `ims` ;
CREATE TABLE IF NOT EXISTS `ims`.`customers` (
        `cid` INT(11) NOT NULL AUTO_INCREMENT,
        `first_name` VARCHAR(40) DEFAULT NULL,
        `surname` VARCHAR(40) DEFAULT NULL,
        `house_number` INT(10) NOT NULL,
        `postcode` VARCHAR(10) NOT NULL,
        PRIMARY KEY (`cid`)
        );
CREATE TABLE IF NOT EXISTS `ims`.`items` (
        `iid` INT(10) NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(20) NOT NULL,
        `price` DEC(5,2) NOT NULL,
        PRIMARY KEY (`iid`)
        );
CREATE TABLE IF NOT EXISTS `ims`.`orders` (
        `oid` INT(10) NOT NULL AUTO_INCREMENT,
        `fk_cid` INT(10) NOT NULL,
        `order_value` DOUBLE,
        PRIMARY KEY (`oid`),
        FOREIGN KEY (`fk_cid`) references customers(`cid`) ON UPDATE CASCADE ON DELETE CASCADE
        );
CREATE TABLE IF NOT EXISTS `ims`.`orders_items` (
        `fk_oid` INT(10) NOT NULL,
        `fk_iid` INT(10) NOT NULL,
        FOREIGN KEY (`fk_oid`) references orders(`oid`) ON UPDATE CASCADE ON DELETE CASCADE,
        FOREIGN KEY (`fk_iid`) references items(`iid`) ON UPDATE CASCADE ON DELETE CASCADE
        );
```

DEMONSTRATION

USER STORIES IN https://qaacademydemo.atlassian.net/jira/software/c/projects/IMS/boards/2/roadmap

APPROACH TO CREATING CRUD METHODS

FOR EACH FUNCTIONALITY:

DATA ACCESS OBJECT	CONTROLLER	RUN

IMPLEMENT IDOMAINDAO AND ADD UNIMPLEMENTED METHODS
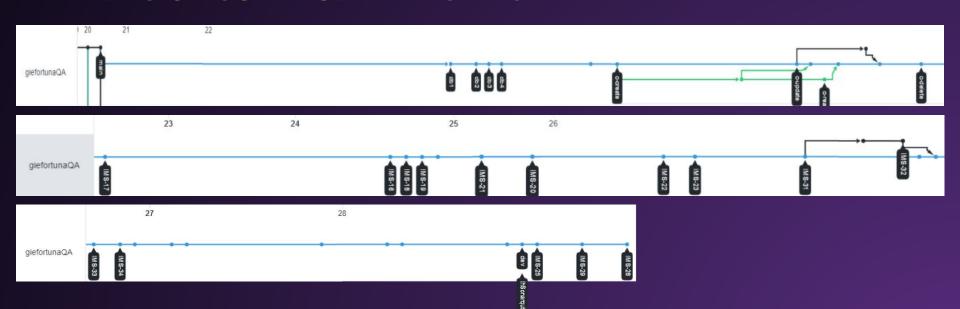
IMPLEMENT ICRUDCONTROLLER AND CALL METHODS FROM DAO CLASS

CREATE PROMPTS TO BE DISPLAYED IN CONSOLE

'TRY' THE CODE BY RUNNING THE METHOD AND CHECKING FUNCTIONALITY

— VERSION CONTROL WITH GITHUB

STARTED TO USE BRANCH NAMES LIKE 'DB1', 'DB2', 'C-CREATE', 'C-READ', …

USED SMART COMMITS FROM IMS-17

VERSION CONTROL WITH GITHUB

Git checkout -b

Running and Checking

Switch to dev branch

1

3

5

2

4

6

Make changes

Git add,
Git commit,
Git push

Merge

TESTING
WITH JUNIT

TESTING WITH JUNIT
WORKING COVERAGE (in Eclipse)

Coverage ✕

20DecSDET2-IMS-Starter (29 Jan 2021 13:00:22)

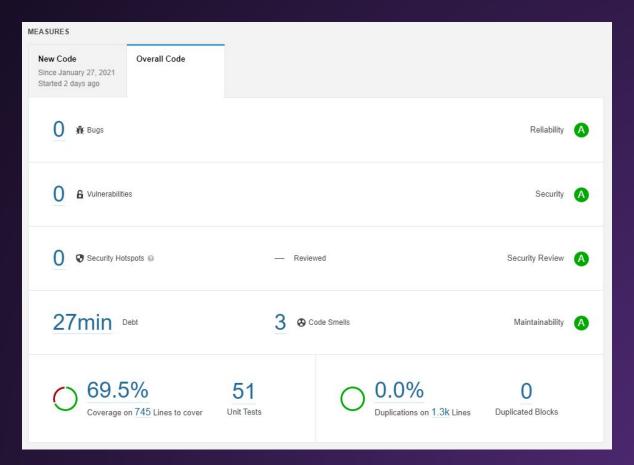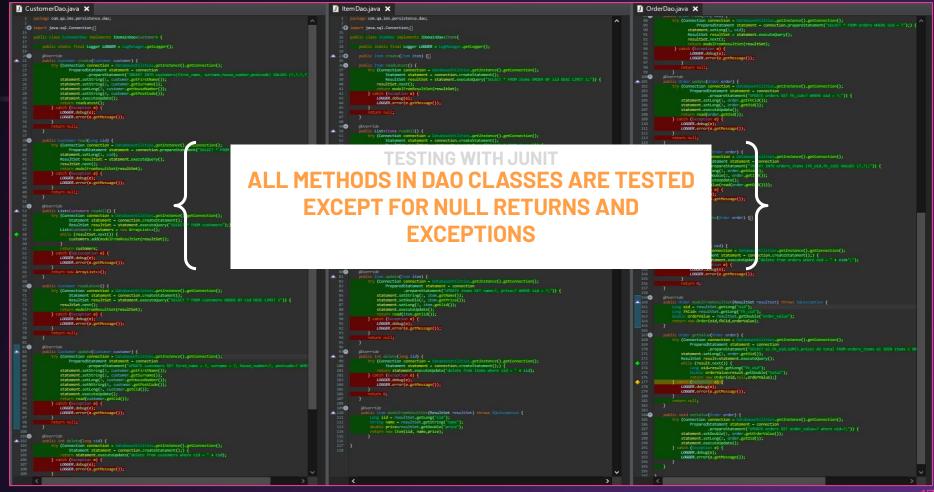| Element	Coverage		Covered Instructions	Missed Instructions	Total Instructions
∨ com.qa.ims.cli	▬	0.0 %	0	365	365
> IMSEntryPoint.java	▬	0.0 %	0	141	141
> ActionMenu.java	▬	0.0 %	0	119	119
> DomainMenu.java	▬	0.0 %	0	105	105
∨ com.qa.ims.persistence.dao	▬	77.2 %	751	222	973
> OrderDao.java	▬	78.9 %	367	98	465
> CustomerDao.java	▬	76.9 %	206	62	268
> ItemDao.java	▬	74.2 %	178	62	240
∨ com.qa.ims.controller	▬	78.6 %	418	114	532
> OrderController.java	▬	58.5 %	161	114	275
> CustomerController.java	▬	100.0 %	141	0	141
> ItemController.java	▬	100.0 %	116	0	116
> com.qa.ims.utils	▮	64.6 %	155	85	240
> com.qa.ims		0.0 %	0	19	19
∨ com.qa.ims.persistence.domain	▬	98.2 %	667	12	679
> Order.java	▬	94.8 %	217	12	229
> Customer.java	▬	100.0 %	272	0	272
> Item.java	▬	100.0 %	178	0	178
> com.qa.ims.controller	▬	100.0 %	761	0	761
> com.qa.ims.persistence.dao	▬	100.0 %	546	0	546
> com.qa.ims.persistence.domain	▬	100.0 %	340	0	340

WORKING COVERAGE (in Sonarqube)

TESTING WITH JUNIT

TESTING WITH JUNIT

ALL METHODS IN DAO CLASSES ARE TESTED EXCEPT FOR NULL RETURNS AND EXCEPTIONS

BURNDOWN CHART

NOTABLE ACHIEVEMENTS

SPRINT RETROSPECTIVE

Jira sprint was successful with no issues left to do.

Able to add extra features that work

Sonarqube code smells were minimised

Tested ALL methods in DAO classes

UML Class Diagram

CONCLUSION

This project was the perfect challenge at this stage of training.
Most tools and knowledge needed were taught in the first four weeks of training.
With more experience in Java, there are a lot more that can be improved:
- More testing with Mockito
- Custom exceptions
- Added functionalities

THANK YOU FOR LISTENING!

Any questions?