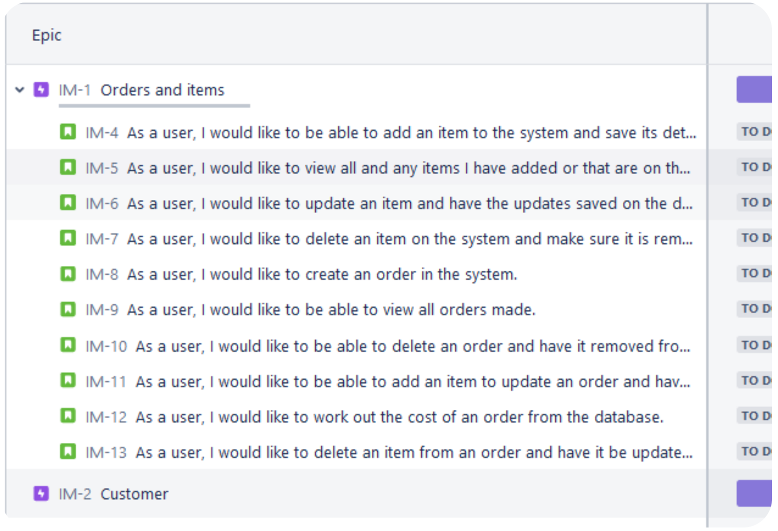
IMS PROJECT

BY NICK ADAMS

FIRST STEP APPROACH

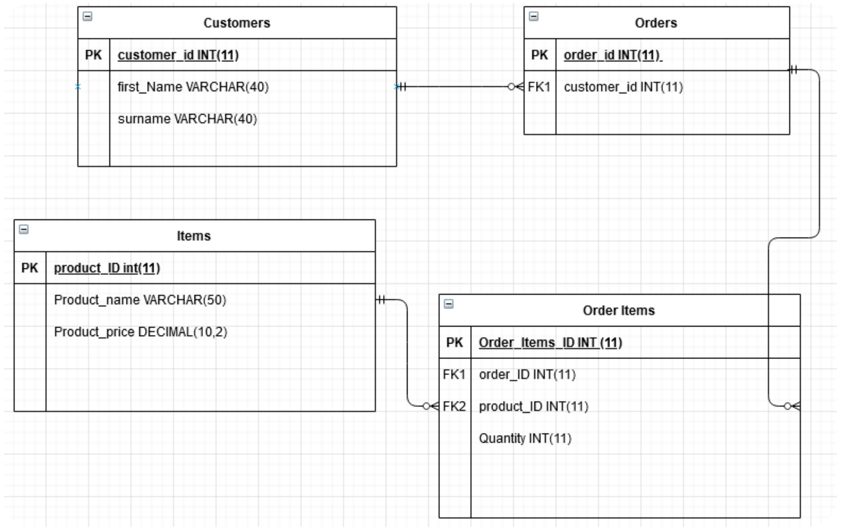


Epic

IM-1 Orders and items

IM-4 As a user, I would like to be able to add an item to the system and save its det...	TO D
IM-5 As a user, I would like to view all and any items I have added or that are on th...	TO D
IM-6 As a user, I would like to update an item and have the updates saved on the d...	TO D
IM-7 As a user, I would like to delete an item on the system and make sure it is rem...	TO D
IM-8 As a user, I would like to create an order in the system.	TO D
IM-9 As a user, I would like to be able to view all orders made.	TO D
IM-10 As a user, I would like to be able to delete an order and have it removed fro...	TO D
IM-11 As a user, I would like to be able to add an item to update an order and hav...	TO D
IM-12 As a user, I would like to work out the cost of an order from the database.	TO D
IM-13 As a user, I would like to delete an item from an order and have it be update...	TO D

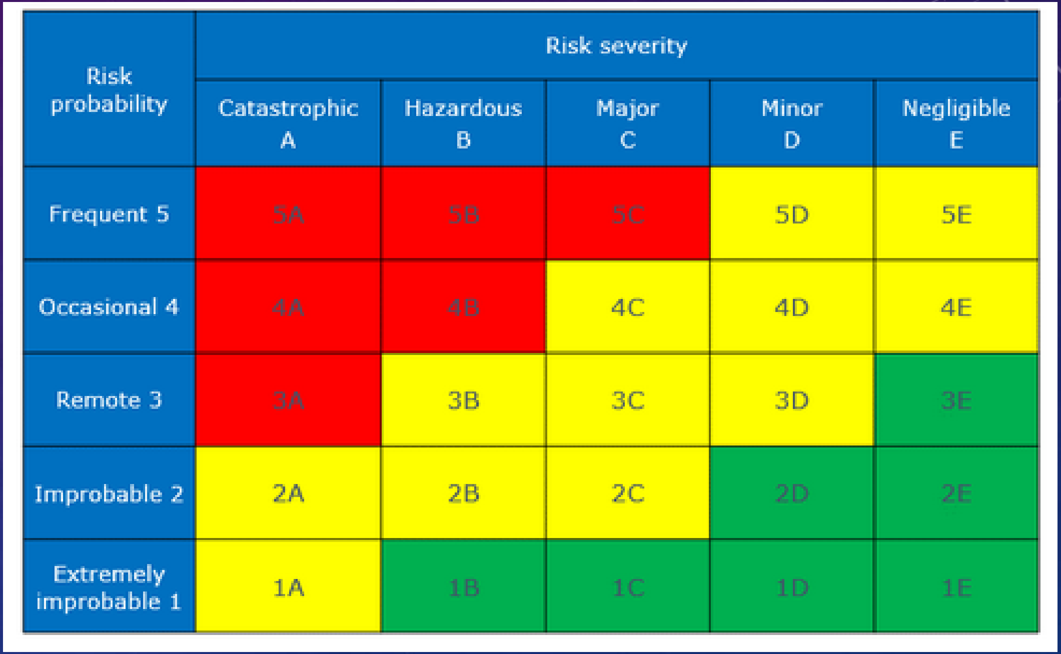
IM-2 Customer



- I had to begin my project with a series of planning tasks.
- These tasks consisted of a Jira board, risk assessment and an ERD.
- These gave me a base on my project and allowed me to have an organised and methodical approach to how I wanted to complete the program.
- These will be touched on again later.

Risks

Risk	Risk Description	Risk avoidance and steps to tackle risk	Risk Level
Covid-19	An outbreak of covid-19 could lead to a developer being unable to physically complete any work due to an illness, which could completely stop development and work on the program.	Ensure that everyone present is taking necessary covid precautions such a mask, 2m distance and common hand sanitising to prevent any spread of the disease.	5D
Power cut	A power cut could lead to a complete loss of a large loss of work which is being worked on that has not been saved. This could lead to an extended time taken to complete the project and disallow deadlines to be met.	Make sure work is saved regularly in case of a power cut and allow regular checks on power sources and plugs to make sure you do not lose power.	2B
Internet cut out	If your internet, you could become unable to communicate with colleagues to discuss development of the project and similarly to the power cut become unable to use github to commit your newly developed code online.	Ensure a strong and stable wi-fi or ethernet connection at all times, and making regular checks to prevent any unplanned downtime.	2A



Risk probability	Risk severity				
	Catastrophic A	Hazardous B	Major C	Minor D	Negligible E
Frequent 5	5A	5B	5C	5D	5E
Occasional 4	4A	4B	4C	4D	4E
Remote 3	3A	3B	3C	3D	3E
Improbable 2	2A	2B	2C	2D	2E
Extremely improbable 1	1A	1B	1C	1D	1E

JOURNEY AS A CONSULTANT

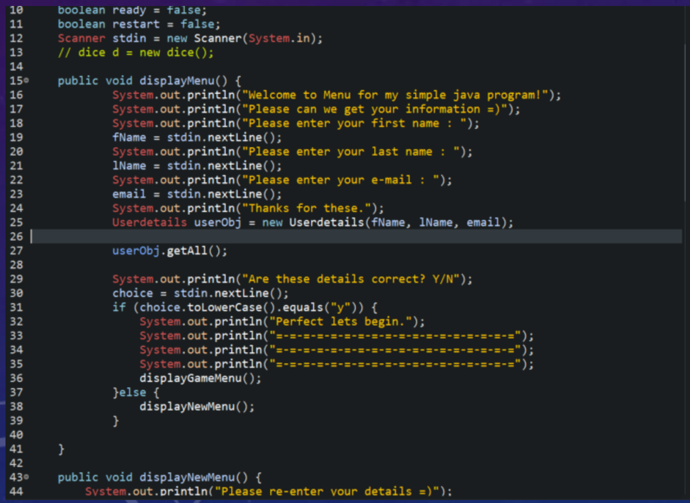
- As a trainee consultant with no prior experience, I needed to learn java from the bottom up to be able to complete this project. Alongside the lecturers and calls in QA, I also worked on projects outside of hours to ensure I had the capability to do so.

- These side projects consisted of the likes of menu orientated programs for example a program with games, along with a menu to be able to choose between them. I did this because I knew it would be a vital playing factor in the project as it is also menu based.

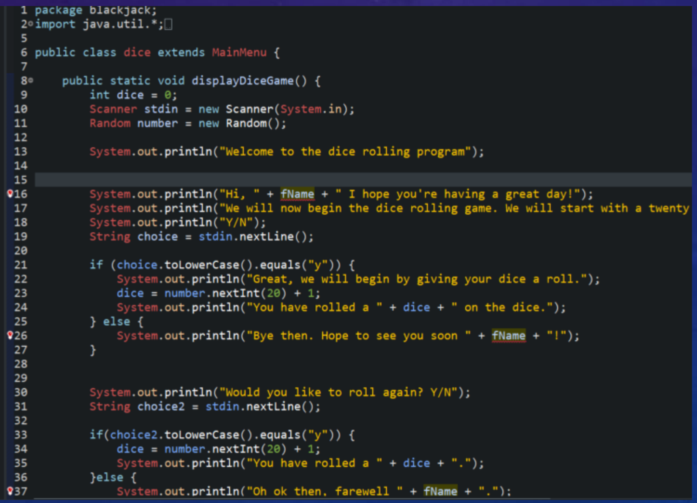
This really helped me with my project as it gave me a much more solidified understanding on the basic principles of OOP in Java, and the relationships between classes and their functions.

<u>Technologies learned</u>

CRUD
JUnit
Mockito
JDBC
Jira
GitHub
SQL
Maven
Java



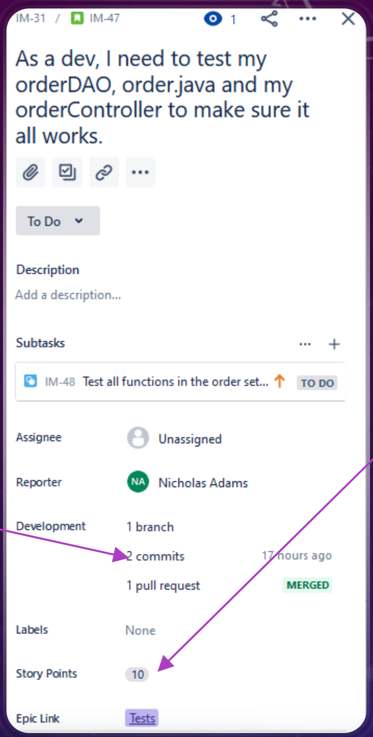
```java
    boolean ready = false;
    boolean restart = false;
    Scanner stdin = new Scanner(System.in);
    // dice d = new dice();

    public void displayMenu() {
            System.out.println("Welcome to Menu for my simple java program!");
            System.out.println("Please can we get your information =)");
            System.out.println("Please enter your first name : ");
            fName = stdin.nextLine();
            System.out.println("Please enter your last name : ");
            lName = stdin.nextLine();
            System.out.println("Please enter your e-mail : ");
            email = stdin.nextLine();
            System.out.println("Thanks for these.");
            Userdetails userObj = new Userdetails(fName, lName, email);

            userObj.getAll();

            System.out.println("Are these details correct? Y/N");
            choice = stdin.nextLine();
            if (choice.toLowerCase().equals("y")) {
                System.out.println("Perfect lets begin.");
                System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
                System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
                System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
                displayGameMenu();
            }else {
                displayNewMenu();
            }

    }

    public void displayNewMenu() {
        System.out.println("Please re-enter your details =)");
```



```java
package blackjack;
import java.util.*;

public class dice extends MainMenu {

    public static void displayDiceGame() {
        int dice = 0;
        Scanner stdin = new Scanner(System.in);
        Random number = new Random();

        System.out.println("Welcome to the dice rolling program");


        System.out.println("Hi, " + fName + " I hope you're having a great day!");
        System.out.println("We will now begin the dice rolling game. We will start with a twenty
        System.out.println("Y/N");
        String choice = stdin.nextLine();

        if (choice.toLowerCase().equals("y")) {
            System.out.println("Great, we will begin by giving your dice a roll.");
            dice = number.nextInt(20) + 1;
            System.out.println("You have rolled a " + dice + " on the dice.");
        } else {
            System.out.println("Bye then. Hope to see you soon " + fName + "!");
        }


        System.out.println("Would you like to roll again? Y/N");
        String choice2 = stdin.nextLine();

        if(choice2.toLowerCase().equals("y")) {
            dice = number.nextInt(20) + 1;
            System.out.println("You have rolled a " + dice + ".");
        }else {
            System.out.println("Oh ok then, farewell " + fName + ".");
```

CONTINUOUS INTERGRATION AND VERSION CONTROL

Smart commits used here for example, these are commits made from git bash which are uploading my changes to my testing files on the order testing which are noted on git hub and jira.



IM-31 / IM-47

As a dev, I need to test my orderDAO, order.java and my orderController to make sure it all works.

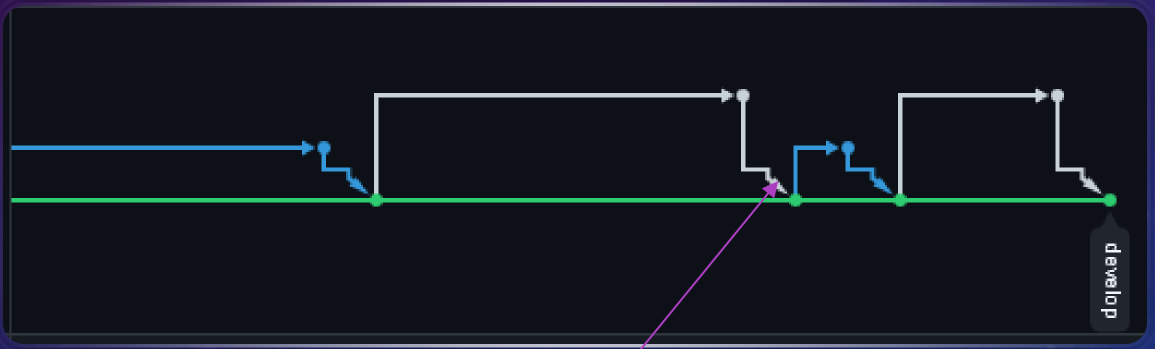
To Do

Description
Add a description...

Subtasks
IM-48 Test all functions in the order set... TO DO

Assignee	Unassigned
Reporter	Nicholas Adams
Development	1 branch
	2 commits 17 hours ago
	1 pull request MERGED
Labels	None
Story Points	10
Epic Link	Tests

The story points tell the developer how much effort will be required to complete the task, here is a 10 as I believed the testing needed a lot of effort to make sure it is working well.

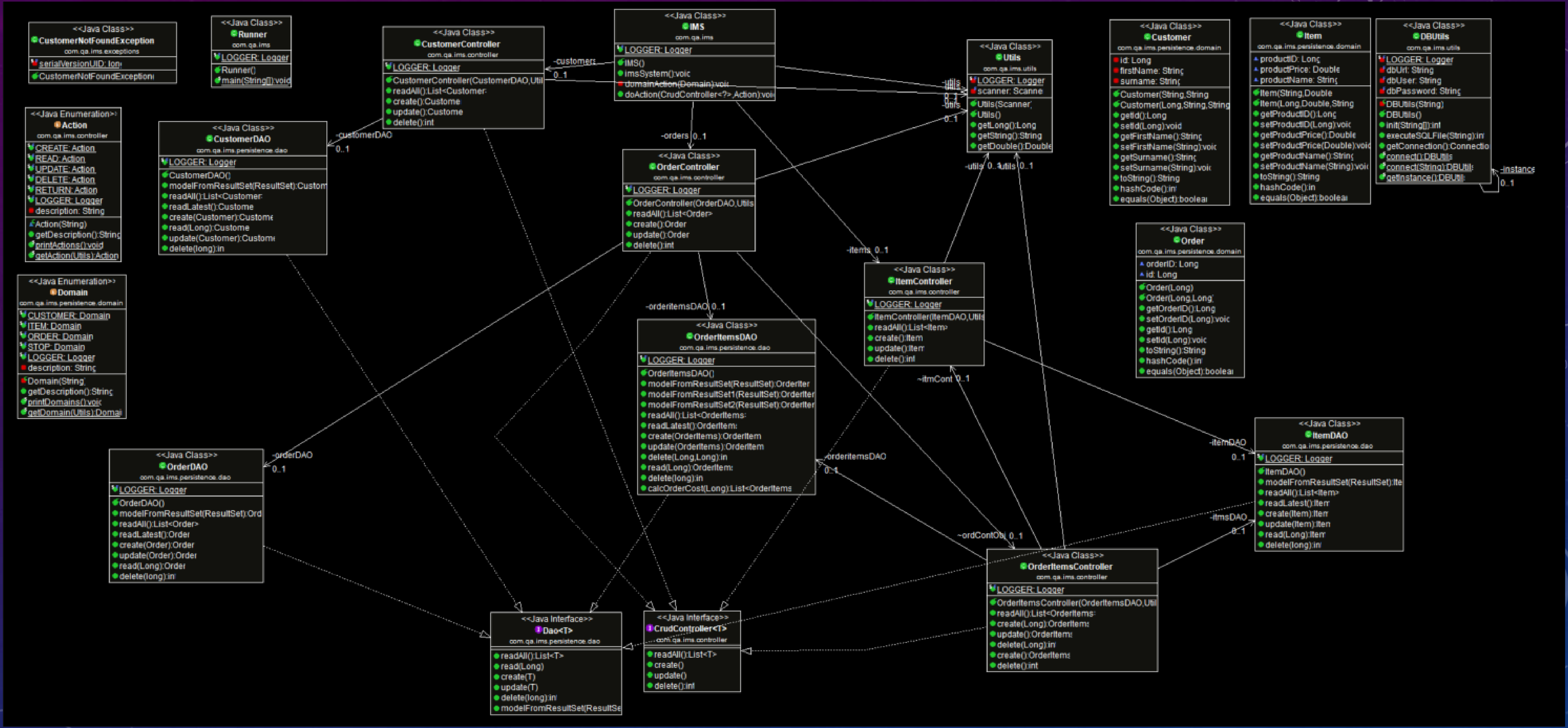


This picture shows my develop branch in green and then multiple feature branches being committed back into the develop branch. Develop has not yet been implemented into the master.

•In terms of continuous intgration, I used github and Jira to make multiple commits using feature branches and my dev branch to ensure there was a steady and methodical approach to completing the project in an optimal way following principles of version control.

•Github – Using github, I was able to make feature branches for each time I created new code. This was specifcally helpful on the testing as my code was often going wrong and having the option to revert to an older version saved me a lot of time.

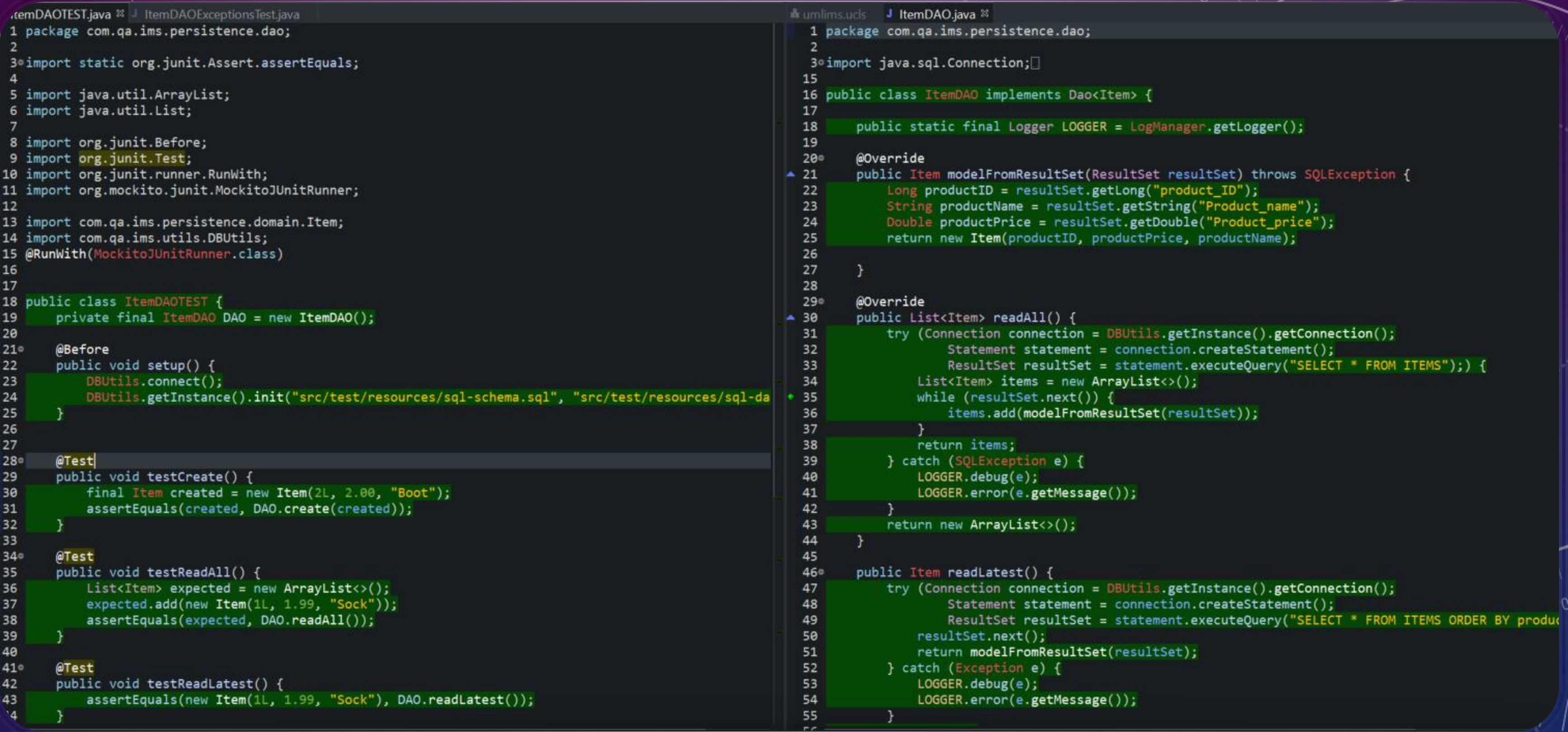
•Jira – Using Jira, I was able to use smart commits for concise and accurate commits personal to each user story, which allowed a higher degree of accuracy and it pin-pointed that the user story had been seen to.



<<Java Class>>
CustomerNotFoundException
com.qa.ims.exceptions
serialVersionUID: long
CustomerNotFoundException(
<<Java Class>>
Runner
com.qa.ims
LOGGER: Logger
Runner()
main(String[]):void
<<Java Class>>
CustomerController
com.qa.ims.controller
LOGGER: Logger
CustomerController(CustomerDAO,Util
readAll():List<Customer
create():Custome
update():Custome
delete():int
<<Java Class>>
IMS
com.qa.ims
LOGGER: Logger
IMS()
imsSystem():voic
domainAction(Domain):voic
doAction(CrudController<?>,Action):voi
-customers
0..1
<<Java Class>>
Utils
com.qa.ims.utils
LOGGER: Logger
scanner: Scanne
Utils(Scanner)
Utils()
getLong():Long
getString():String
getDouble():Double
-utils
0..1
-utils
0..1
-utils 0..1
-utils 0..1
<<Java Class>>
Customer
com.qa.ims.persistence.domain
id: Long
firstName: String
surname: String
Customer(String,String
Customer(Long,String,String
getId():Long
setId(Long):void
getFirstName():String
setFirstName(String):voic
getSurname():String
setSurname(String):voic
toString():String
hashCode():in
equals(Object):boolea
<<Java Class>>
Item
com.qa.ims.persistence.domain
productID: Long
productPrice: Double
productName: String
Item(String,Double
Item(Long,Double,String
getProductID():Long
setProductID(Long):voic
getProductPrice():Double
setProductPrice(Double):voic
getProductName():String
setProductName(String):voic
toString():String
hashCode():in
equals(Object):boolea
<<Java Class>>
DBUtils
com.qa.ims.utils
LOGGER: Logger
dbUrl: String
dbUser: String
dbPassword: String
DBUtils(String)
DBUtils()
init(String[]):int
executeSQLFile(String):in
getConnection():Connectio
connect():DBUtils
connect(String):DBUtils
getInstance():DBUtils
-instance
0..1
<<Java Enumeration>>
Action
com.qa.ims.controller
CREATE: Action
READ: Action
UPDATE: Action
DELETE: Action
RETURN: Action
LOGGER: Logger
description: String
Action(String)
getDescription():String
printActions():void
getAction(Utils):Action
<<Java Class>>
CustomerDAO
com.qa.ims.persistence.dao
LOGGER: Logger
CustomerDAO()
modelFromResultSet(ResultSet):Custom
readAll():List<Customer
readLatest():Custome
create(Customer):Custome
read(Long):Custome
update(Customer):Custome
delete(long):in
-customerDAO
0..1
-orders 0..1
<<Java Class>>
OrderController
com.qa.ims.controller
LOGGER: Logger
OrderController(OrderDAO,Utils
readAll():List<Order>
create():Order
update():Order
delete():int
-items 0..1
<<Java Class>>
ItemController
com.qa.ims.controller
LOGGER: Logger
ItemController(ItemDAO,Utils
readAll():List<Item>
create():Item
update():Item
delete():int
<<Java Class>>
Order
com.qa.ims.persistence.domain
orderID: Long
id: Long
Order(Long)
Order(Long,Long)
getOrderID():Long
setOrderID(Long):voic
getId():Long
setId(Long):voic
toString():String
hashCode():in
equals(Object):boolea
<<Java Enumeration>>
Domain
com.qa.ims.persistence.domain
CUSTOMER: Domain
ITEM: Domain
ORDER: Domain
STOP: Domain
LOGGER: Logger
description: String
Domain(String)
getDescription():String
printDomains():voic
getDomain(Utils):Domai
-orderItemsDAO 0..1
<<Java Class>>
OrderItemsDAO
com.qa.ims.persistence.dao
LOGGER: Logger
OrderItemsDAO()
modelFromResultSet(ResultSet):OrderIter
modelFromResultSet1(ResultSet):OrderIter
modelFromResultSet2(ResultSet):OrderIter
readAll():List<OrderItems
readLatest():OrderItem
create(OrderItems):OrderItem
update(OrderItems):OrderItem
delete(Long,Long):in
read(Long):OrderItem
delete(long):in
calcOrderCost(Long):List<OrderItems
-itmCont 0..1
-orderItemsDAO
0..1
-itemDAO
0..1
<<Java Class>>
ItemDAO
com.qa.ims.persistence.dao
LOGGER: Logger
ItemDAO()
modelFromResultSet(ResultSet):Ite
readAll():List<Item>
readLatest():Item
create(Item):Item
update(Item):Item
read(Long):Item
delete(long):in
-itmsDAO
0..1
<<Java Class>>
OrderDAO
com.qa.ims.persistence.dao
LOGGER: Logger
OrderDAO()
modelFromResultSet(ResultSet):Ord
readAll():List<Order>
readLatest():Order
create(Order):Order
update(Order):Order
read(Long):Order
delete(long):in
-orderDAO
0..1
-ordContObj 0..1
<<Java Class>>
OrderItemsController
com.qa.ims.controller
LOGGER: Logger
OrderItemsController(OrderItemsDAO,Util
readAll():List<OrderItems
create(Long):OrderItems
update():OrderItems
delete(Long):in
create():OrderItems
delete():int
<<Java Interface>>
Dao<T>
com.qa.ims.persistence.dao
readAll():List<T>
read(Long)
create(T)
update(T)
delete(long):in
modelFromResultSet(ResultSe
<<Java Interface>>
CrudController<T>
com.qa.ims.controller
readAll():List<T>
create()
update()
delete():int

TESTING

To allow the release of my program, I needed to perform testing. I used unit testing and mock testing to allow group method testing. I attempted to cover as many areas of the program as possible to provide maximum coverage.

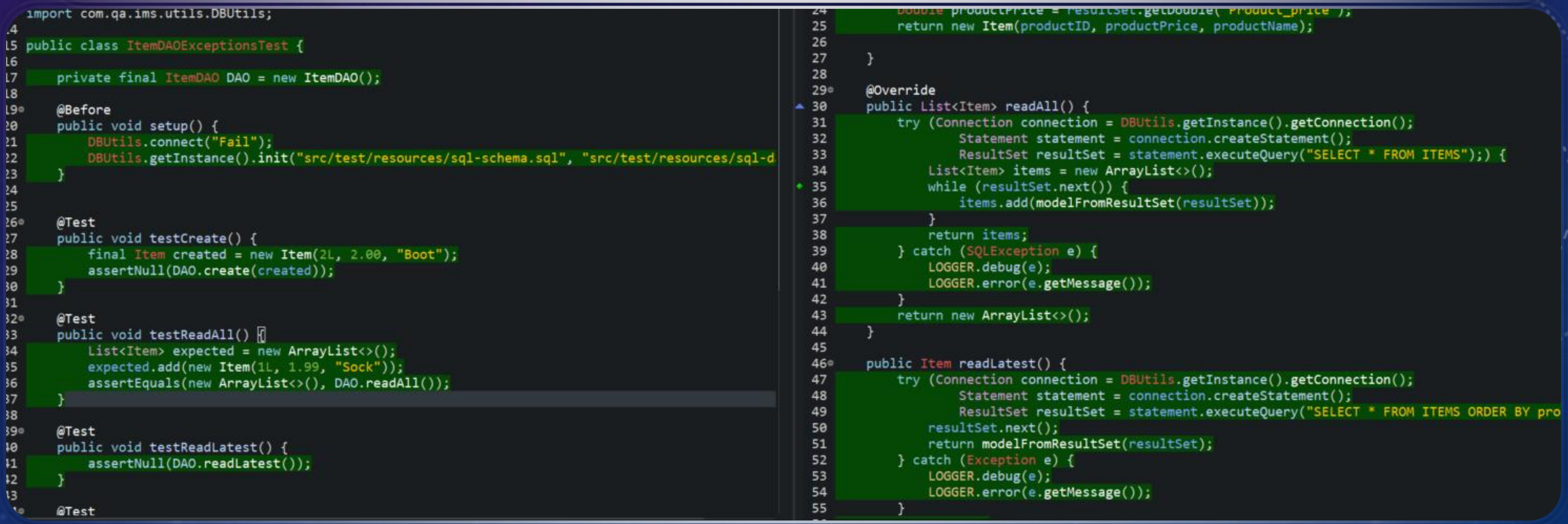


temDAOTEST.java ItemDAOExceptionsTest.java

```java
package com.qa.ims.persistence.dao;

import static org.junit.Assert.assertEquals;

import java.util.ArrayList;
import java.util.List;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.junit.MockitoJUnitRunner;

import com.qa.ims.persistence.domain.Item;
import com.qa.ims.utils.DBUtils;
@RunWith(MockitoJUnitRunner.class)


public class ItemDAOTEST {
    private final ItemDAO DAO = new ItemDAO();

    @Before
    public void setup() {
        DBUtils.connect();
        DBUtils.getInstance().init("src/test/resources/sql-schema.sql", "src/test/resources/sql-da
    }


    @Test
    public void testCreate() {
        final Item created = new Item(2L, 2.00, "Boot");
        assertEquals(created, DAO.create(created));
    }

    @Test
    public void testReadAll() {
        List<Item> expected = new ArrayList<>();
        expected.add(new Item(1L, 1.99, "Sock"));
        assertEquals(expected, DAO.readAll());
    }

    @Test
    public void testReadLatest() {
        assertEquals(new Item(1L, 1.99, "Sock"), DAO.readLatest());
    }
```

umlims.ucls ItemDAO.java

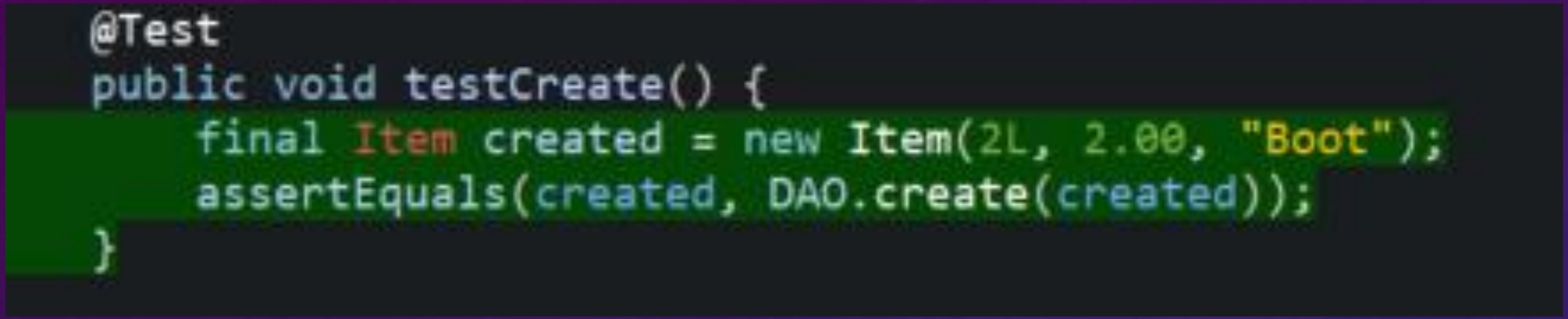
```java
package com.qa.ims.persistence.dao;

import java.sql.Connection;

public class ItemDAO implements Dao<Item> {

    public static final Logger LOGGER = LogManager.getLogger();

    @Override
    public Item modelFromResultSet(ResultSet resultSet) throws SQLException {
        Long productID = resultSet.getLong("product_ID");
        String productName = resultSet.getString("Product_name");
        Double productPrice = resultSet.getDouble("Product_price");
        return new Item(productID, productPrice, productName);

    }

    @Override
    public List<Item> readAll() {
        try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM ITEMS");) {
            List<Item> items = new ArrayList<>();
            while (resultSet.next()) {
                items.add(modelFromResultSet(resultSet));
            }
            return items;
        } catch (SQLException e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return new ArrayList<>();
    }

    public Item readLatest() {
        try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM ITEMS ORDER BY produc
            resultSet.next();
            return modelFromResultSet(resultSet);
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
```



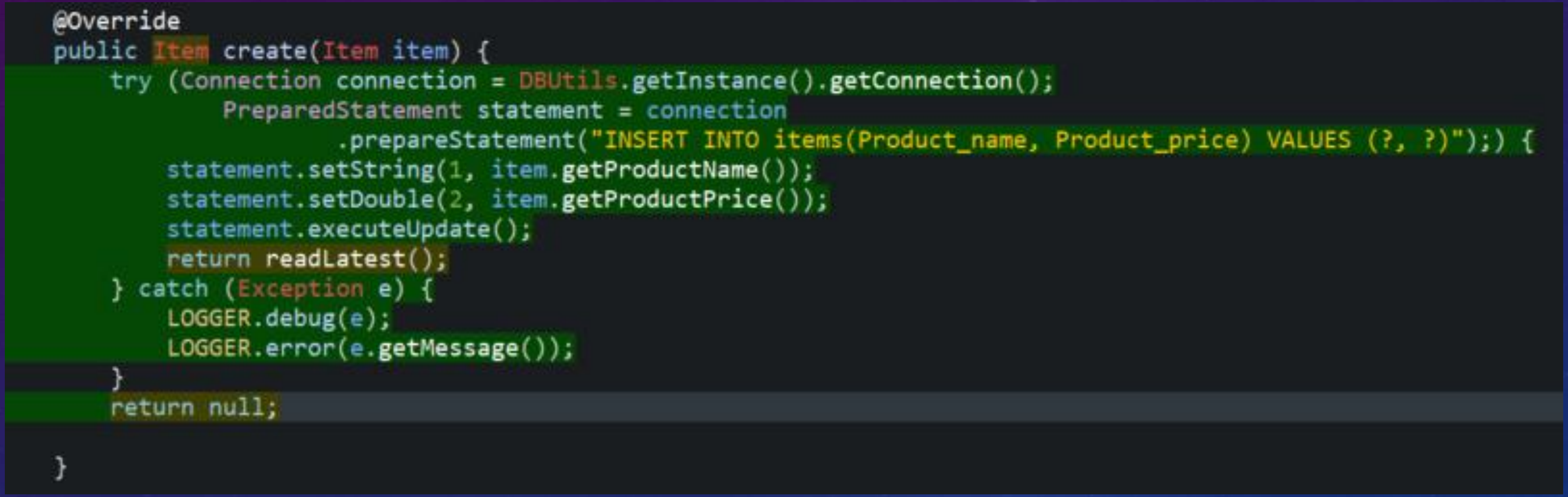
```java
import com.qa.ims.utils.DBUtils;

public class ItemDAOExceptionsTest {

    private final ItemDAO DAO = new ItemDAO();

    @Before
    public void setup() {
        DBUtils.connect("Fail");
        DBUtils.getInstance().init("src/test/resources/sql-schema.sql", "src/test/resources/sql-d
    }


    @Test
    public void testCreate() {
        final Item created = new Item(2L, 2.00, "Boot");
        assertNull(DAO.create(created));
    }

    @Test
    public void testReadAll() {
        List<Item> expected = new ArrayList<>();
        expected.add(new Item(1L, 1.99, "Sock"));
        assertEquals(new ArrayList<>(), DAO.readAll());
    }

    @Test
    public void testReadLatest() {
        assertNull(DAO.readLatest());
    }

    @Test
```

```java
        return new Item(productID, productPrice, productName);

    }

    @Override
    public List<Item> readAll() {
        try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM ITEMS");) {
            List<Item> items = new ArrayList<>();
            while (resultSet.next()) {
                items.add(modelFromResultSet(resultSet));
            }
            return items;
        } catch (SQLException e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return new ArrayList<>();
    }

    public Item readLatest() {
        try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM ITEMS ORDER BY pro
            resultSet.next();
            return modelFromResultSet(resultSet);
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
```

TEST USE CASE EXAMPLE



```java
@Test
public void testCreate() {
    final Item created = new Item(2L, 2.00, "Boot");
    assertEquals(created, DAO.create(created));
}
```

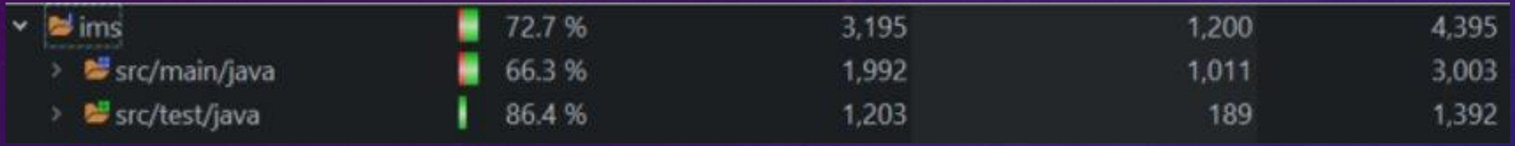


```java
@Override
public Item create(Item item) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection
                    .prepareStatement("INSERT INTO items(Product_name, Product_price) VALUES (?, ?)");) {
        statement.setString(1, item.getProductName());
        statement.setDouble(2, item.getProductPrice());
        statement.executeUpdate();
        return readLatest();
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;

}
```

DEMONSTRATION

Sprint review + Restrospective

In my sprint review, I concluded that I left behind 80% coverage which was in the specification. Having the opportunity to do this project again or given more time I would likely commit more time to testing as I feel like 2/3 testing is not sufficient for the user or client.



ims	72.7 %	3,195	1,200	4,395
src/main/java	66.3 %	1,992	1,011	3,003
src/test/java	86.4 %	1,203	189	1,392

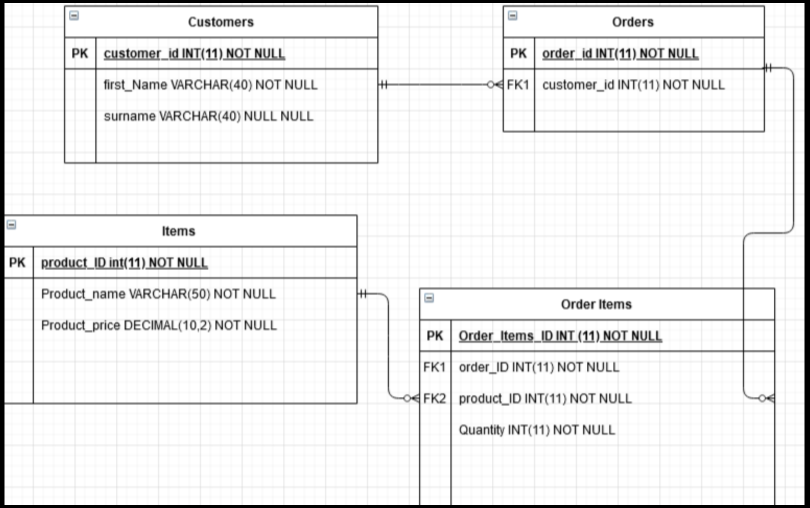
Heavier planning prior to beginning the project would also be on my agenda as the lack of github inegration at the beginning of my journey on this project caused me issues later-on when I realised the true strength of the tool as I needed to revert to an older version as my program malfunctioned

WHAT WENT WELL?

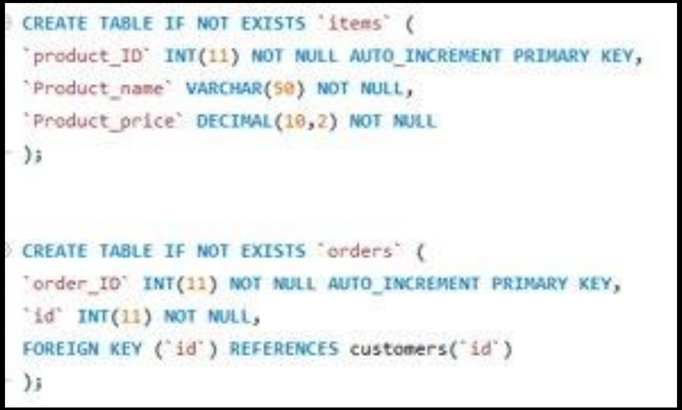
- JIRA
- GITHUB
- IMS Application
- Journey

WHAT COULD BE IMPROVED

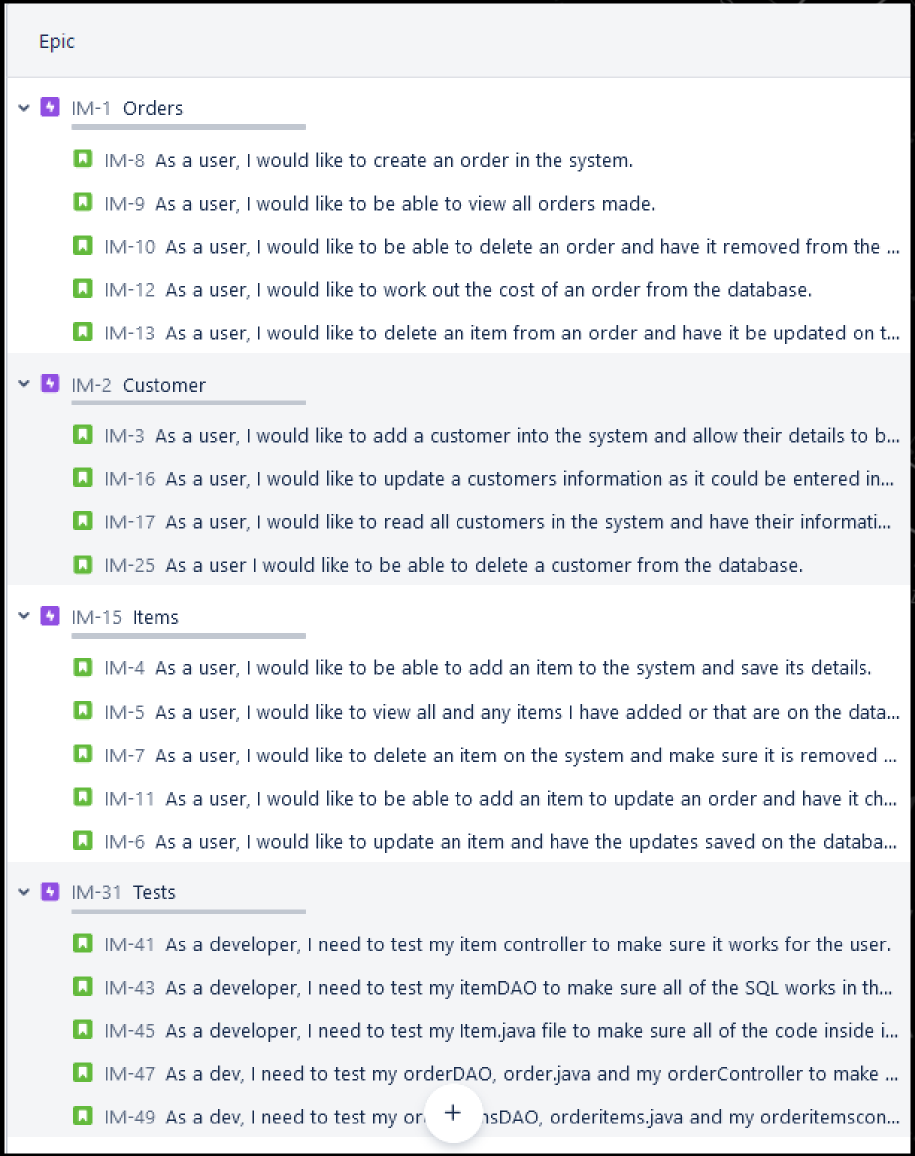
- Testing
- QOL for user on IMS app
- Time management
- SOLID principles



Customers
PK customer_id INT(11) NOT NULL
first_Name VARCHAR(40) NOT NULL
surname VARCHAR(40) NULL NULL
Orders
PK order_id INT(11) NOT NULL
FK1 customer_id INT(11) NOT NULL
Items
PK product_ID int(11) NOT NULL
Product_name VARCHAR(50) NOT NULL
Product_price DECIMAL(10,2) NOT NULL
Order Items
PK Order_Items_ID INT (11) NOT NULL
FK1 order_ID INT(11) NOT NULL
FK2 product_ID INT(11) NOT NULL
Quantity INT(11) NOT NULL



```
CREATE TABLE IF NOT EXISTS `items` (
`product_ID` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
`Product_name` VARCHAR(50) NOT NULL,
`Product_price` DECIMAL(10,2) NOT NULL
);


CREATE TABLE IF NOT EXISTS `orders` (
`order_ID` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
`id` INT(11) NOT NULL,
FOREIGN KEY (`id`) REFERENCES customers(`id`)
);
```



Epic

IM-1 Orders

- IM-8 As a user, I would like to create an order in the system.
- IM-9 As a user, I would like to be able to view all orders made.
- IM-10 As a user, I would like to be able to delete an order and have it removed from the ...
- IM-12 As a user, I would like to work out the cost of an order from the database.
- IM-13 As a user, I would like to delete an item from an order and have it be updated on t...

IM-2 Customer

- IM-3 As a user, I would like to add a customer into the system and allow their details to b...
- IM-16 As a user, I would like to update a customers information as it could be entered in...
- IM-17 As a user, I would like to read all customers in the system and have their informati...
- IM-25 As a user I would like to be able to delete a customer from the database.

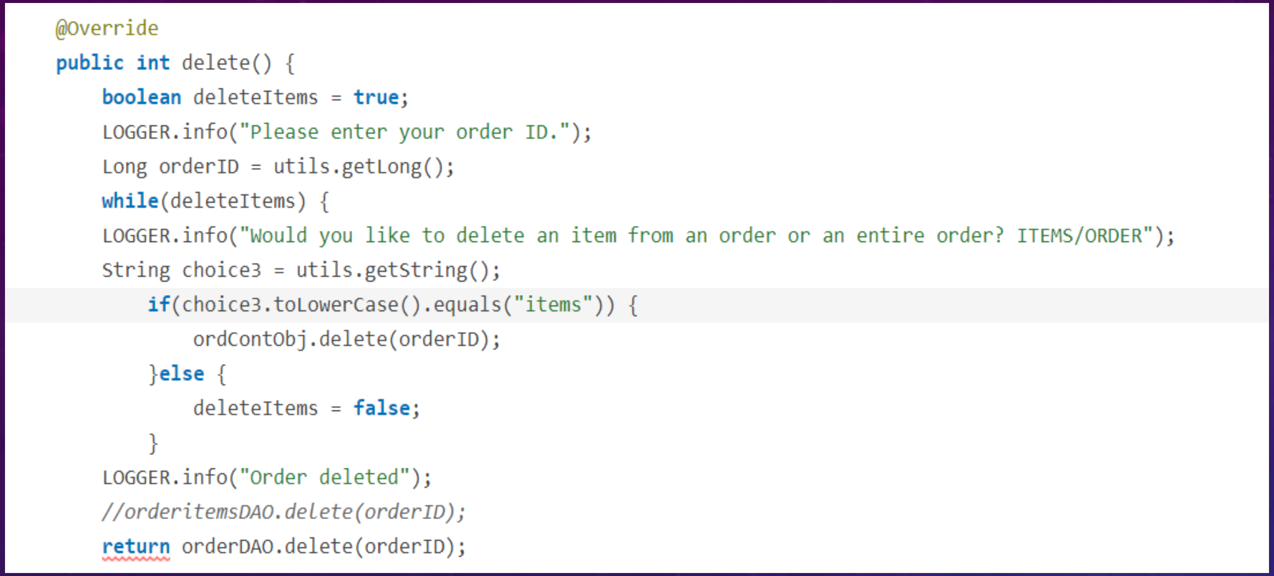
IM-15 Items

- IM-4 As a user, I would like to be able to add an item to the system and save its details.
- IM-5 As a user, I would like to view all and any items I have added or that are on the data...
- IM-7 As a user, I would like to delete an item on the system and make sure it is removed ...
- IM-11 As a user, I would like to be able to add an item to update an order and have it ch...
- IM-6 As a user, I would like to update an item and have the updates saved on the databa...

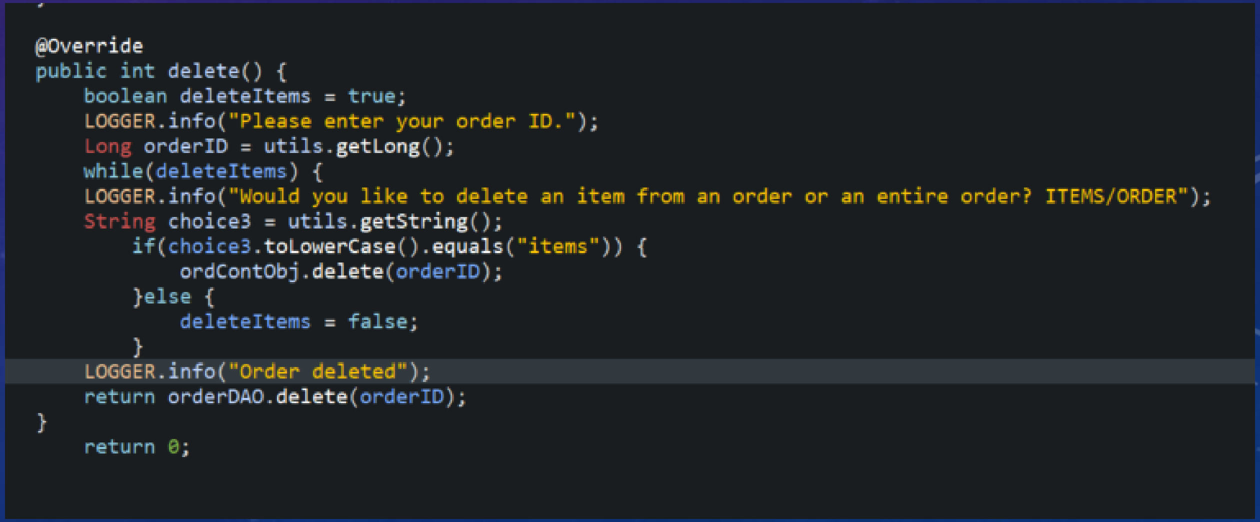
IM-31 Tests

- IM-41 As a developer, I need to test my item controller to make sure it works for the user.
- IM-43 As a developer, I need to test my itemDAO to make sure all of the SQL works in th...
- IM-45 As a developer, I need to test my Item.java file to make sure all of the code inside i...
- IM-47 As a dev, I need to test my orderDAO, order.java and my orderController to make ...
- IM-49 As a dev, I need to test my or[illegible]sDAO, orderitems.java and my orderitemscon...

SonarQube



```java
@Override
public int delete() {
    boolean deleteItems = true;
    LOGGER.info("Please enter your order ID.");
    Long orderID = utils.getLong();
    while(deleteItems) {
    LOGGER.info("Would you like to delete an item from an order or an entire order? ITEMS/ORDER");
    String choice3 = utils.getString();
        if(choice3.toLowerCase().equals("items")) {
            ordContObj.delete(orderID);
        }else {
            deleteItems = false;
        }
    LOGGER.info("Order deleted");
    //orderitemsDAO.delete(orderID);
    return orderDAO.delete(orderID);
```



```java
@Override
public int delete() {
    boolean deleteItems = true;
    LOGGER.info("Please enter your order ID.");
    Long orderID = utils.getLong();
    while(deleteItems) {
    LOGGER.info("Would you like to delete an item from an order or an entire order? ITEMS/ORDER");
    String choice3 = utils.getString();
        if(choice3.toLowerCase().equals("items")) {
            ordContObj.delete(orderID);
        }else {
            deleteItems = false;
        }
    LOGGER.info("Order deleted");
    return orderDAO.delete(orderID);
}
    return 0;
```

Conclusion

In conclusion I believe although the program and approach was not the most clean, concise or methodical way, the end result was most definitely good as it was a learning curve in-itself. Allowing me to realise the absolute necessity of git hub, a strong plan and ultimately the ability to write programs in Java.

I have also learned many new practises during the project, such as JDBC which will be imperative to my future learning.