IMS DATABASE DESIGN

Presented by Arsalan Asad

Introduction

My background:
- Studied chemical engineering at university.
- During that period, the only coding learnt was with MatLab.
- After university, I had a coding job which involved me learning very basic java to teach children and create fun exercises to teach them with.
- For a while I learnt python and wanted to start sql.

Breaking down the specification:
- 3 main parts.
- Beginning; Jira board, ERD and planning
- Middle; implementing code from starting point to create workable IMS.
- End; completion of tests uploaded to github along with readme and documentation.

Consultant Journey

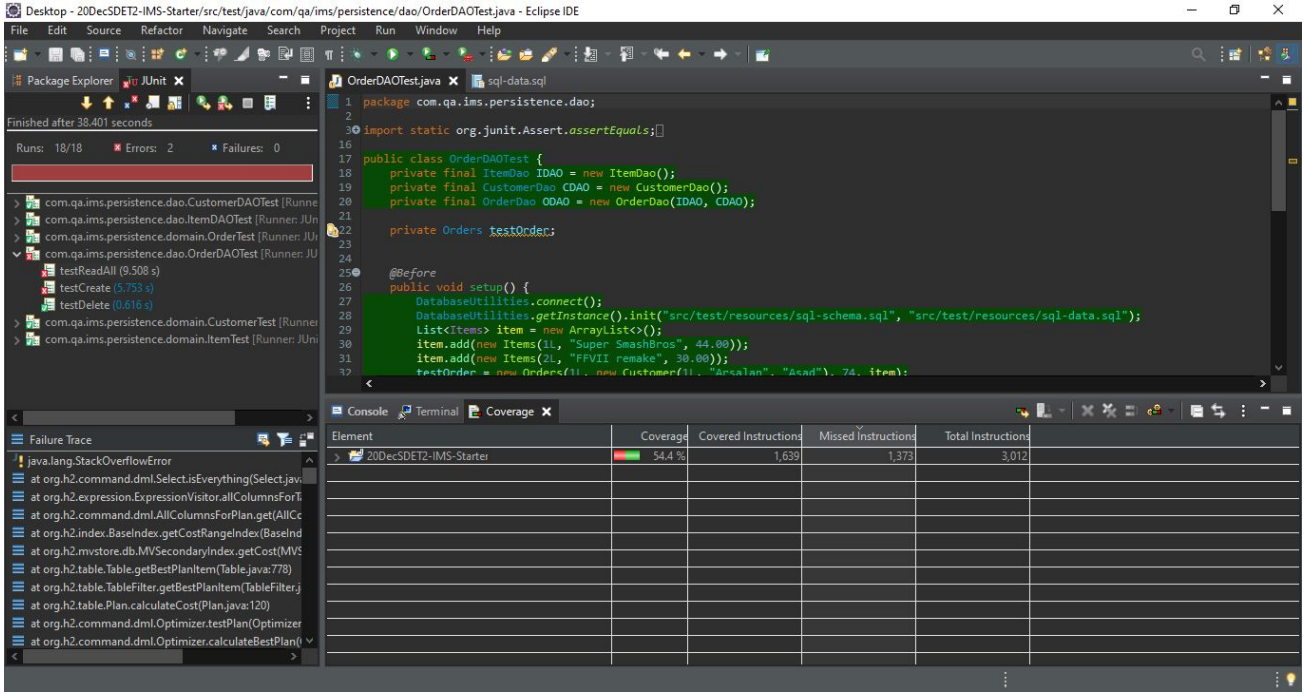
From where I started, I have improved my coding knowledge and skill tenfold.
This project required me to learn a variety of new technologies and implement them in different ways, for example;
-The use of Git to fork the original repo provided by Nicholas, cloning that repo to make it our own and work upon the code from there.
-The use of Eclipse IDE to implement Java code.
-Mysql to create a database and create tables within in which we can then add our own values.
- The use of JDBC to connect sql to Java.
- J unit and mockito to allow for testing.

CI

The version control involved in this project was github. Initially I had to fork the repository already created by Nicholas and clone that repo to my local machine. This was done by first opening the project as a repo once it was forked and cloned and then initialising the project as a git. Once this had been done I had to create a remote to allow me to push all my changes back to my repo in github. Throughout the project any changes made to the projects would be pushed and commits made so that anyone would be able to see the steps taken in creating this database.

Testing

For my testing J unit was used to conduct tests.
The main classes that were tested were the customers, items and order domains. The custome Dao, itemDao and orderDao were also tested.
The overall coverage can be seen below.



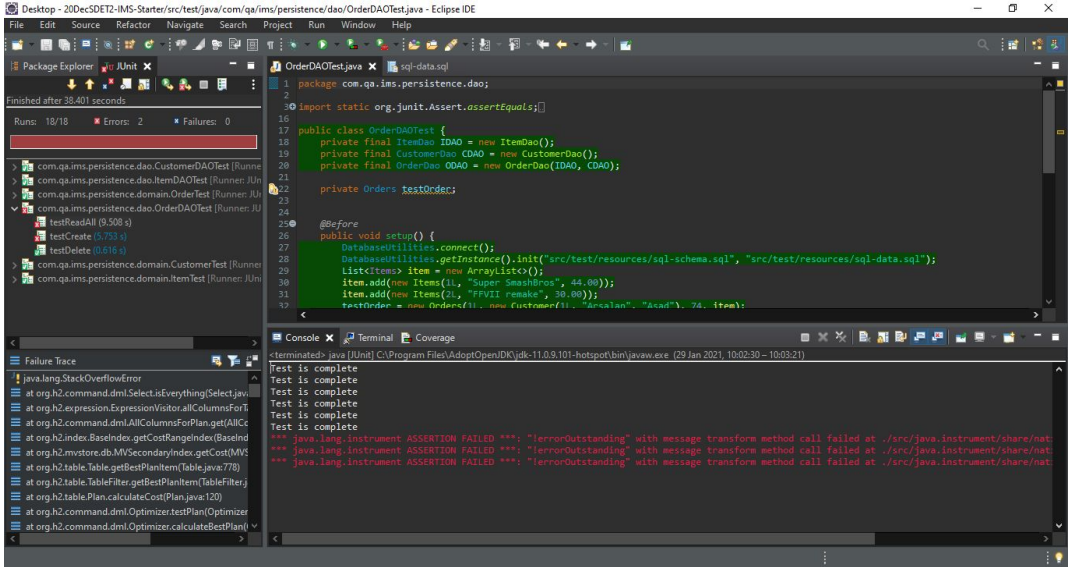
Solving of Errors

```java
24
25    @Test
26    public void testCreate() {
27        final Customer created = new Customer(2L, "Arsalan", "Asad");
28        assertEquals(created, DAO.create(created));
29    }
30
31    @Test
32    public void testReadAll() {
33        List<Customer> expected = new ArrayList<>();
34        expected.add(new Customer(1L, "jordan", "harrison"));
35        assertEquals(expected, DAO.readAll());
36    }
37
38    @Test
39    public void testReadLatest() {
40        assertEquals(new Customer(1L, "jordan", "harrison"), DAO.readLatest());
41    }
42
43    @Test
44    public void testRead() {
45        final long ID = 1L;
46        assertEquals(new Customer(ID, "jordan", "harrison"), DAO.read(ID));
47    }
48
49    @Test
50    public void testUpdate() {
51        final Customer updated = new Customer(1L, "nick", "johnson");
52        assertEquals(updated, DAO.update(updated));
53
54    }
55
56    @Test
```

```java
14 public class CustomerDAOTest {
15
16
17     private final CustomerDao DAO = new CustomerDao();
18
19     @Before
20     public void setup() {
21         DatabaseUtilities.connect();
22         DatabaseUtilities.getInstance().init("src/test/resources/sql-schema.sql", "src/test/resources/sql-data.sql");
23     }
24
25     @Test
26     public void testCreate() {
27         final Customer created = new Customer(2L, "Arsalan", "Asad");
28         assertEquals(created, DAO.create(created));
29     }
30
31     @Test
32     public void testReadAll() {
33         List<Customer> expected = new ArrayList<>();
34         expected.add(new Customer(1L, "jordan", "harrison"));
35         assertEquals(expected, DAO.readAll());
36     }
37
38     @Test
39     public void testReadLatest() {
40         assertEquals(new Customer(1L, "jordan", "harrison"), DAO.readLatest());
41     }
42
43     @Test
44     public void testRead() {
45         final long ID = 1L;
46         assertEquals(new Customer(ID, "jordan", "harrison"), DAO.read(ID));
47     }
```

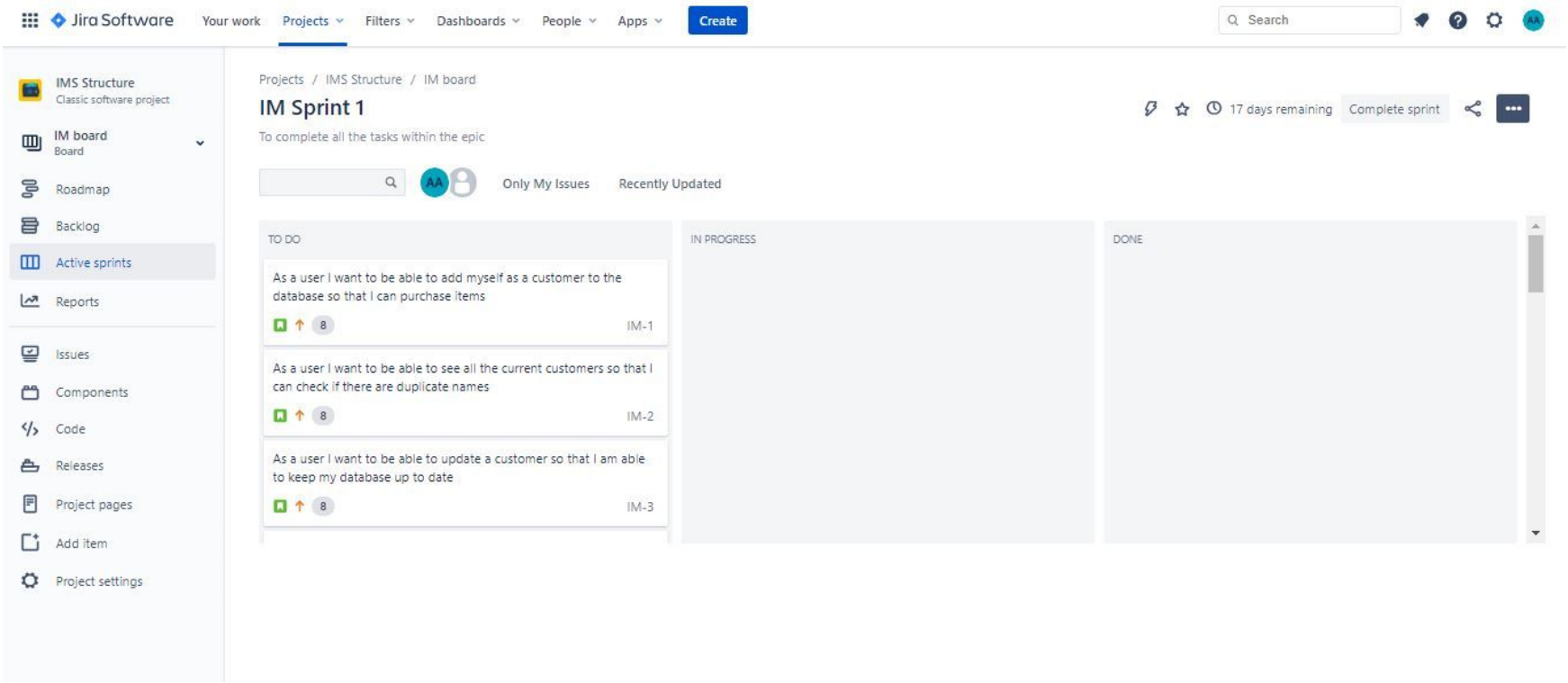
The StackOverflow Error

This error was very unprecedented and occurred in the very final parts of my project. The stackOverflow error would occur any time the orderDaoTest would be run. At first, this issue was believed to be caused by too many connections being open at once, meaning that my sql database had not been closed and was crashing. However, after further inspection and re-installation of sql, I discovered that one of the tests was looping through a few methods in my original orderDao class. Unfortunately, due to time constraints, I haven't been unable to complete this test and remove the stackOverflow error.



Sprint Review

Now we will take a look at my initial sprint and how I progressed through the sprint.



(19) DOCX File viewer | Micro | Untitled presentation - Google Sl | IM board - Agile Board - Jira

academytraining.atlassian.net/secure/RapidBoard.jspa?projectKey=IM&rapidView=4

Jira Software | Your work | Projects | Filters | Dashboards | People | Apps | Create | Search

IMS Structure
Classic software project

IM board
Board

- Roadmap
- Backlog
- Active sprints
- Reports
- Issues
- Components
- Code
- Releases
- Project pages
- Add item
- Project settings

Projects / IMS Structure / IM board

IM Sprint 1

To complete all the tasks within the epic

10 days remaining | Complete sprint

Only My Issues | Recently Updated

TO DO

Create Presentation
IMS Database
IM-18

Create README
IMS Database
IM-19

IN PROGRESS

DONE

As a user I want to be able to add myself as a customer to the database so that I can purchase items
8 ~~IM-1~~

As a user I want to be able to see all the current customers so that I can check if there are duplicate names
8 ~~IM-2~~

As a user I want to be able to update a customer so that I am able to keep my database up to date
8 ~~IM-3~~

As a user I want to be able to delete a customer so that I can see who is still currently in my database
8 ~~IM-4~~

As a user I want to be able to add an item to the database so customers can buy them
8 ~~IM-5~~

As a user I want to be able to update my items so that I can add a

Items codes

/qa/ims/persistence/domain/Items.java - Eclipse IDE

Project Run Window Help

OrderDAOTest.java | sql-data.sql | Items.java

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

public class Items {

    private Long iID;
    private String itemName;
    private double price;

    public Items(String itemName, double Price) {
        this.setItemName(itemName);
        this.setPrice(Price);
    }

    public Items(Long iID, String itemName, double Price) {
        this.setIid(iID);
        this.setItemName(itemName);
        this.setPrice(Price);
    }

    public Long getIid() {
        return iID;
    }

    public void setIid(Long iid) {
        iID = iid;
    }

    public String getItemName() {
        return itemName;
    }

    public void setItemName(String itemName) {
        this.itemName = itemName;
    }
```

Console | Terminal | Coverage

```
<terminated> java [JUnit] C:\Program Files\AdoptOpenJDK\jdk-11.0.9.101-hotspot\bin\java
Test is complete
```

ms/persistence/dao/ItemDao.java - Eclipse IDE

Project Run Window Help

OrderDAOTest.java | sql-data.sql | Items.java | *ItemDao.java

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

import java.sql.Connection;

public class ItemDao implements IDomainDao<Items> {

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

    @Override
    public Items create(Items items) {
        try (Connection connection = DatabaseUtilities.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("INSERT INTO items (item_name, price) VALUES (?,?)");) {
            statement.setString(1, items.getItemName());
            statement.setDouble(2, items.getPrice());
            statement.executeUpdate();
            return readLatest();
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());

        }

        return null;
    }

    public Items read(Long iID) {

        try (Connection connection = DatabaseUtilities.getInstance().getConnection();
                PreparedStatement statement = connection.prepareStatement("SELECT * FROM items WHERE iid = ?");) {
            statement.setLong(1, iID);
            ResultSet resultSet = statement.executeQuery();
            resultSet.next();
            return modelFromResultSet(resultSet);
```

Console | Terminal | Coverage

```
<terminated> java [JUnit] C:\Program Files\AdoptOpenJDK\jdk-11.0.9.101-hotspot\bin\javaw.exe (29 Jan 2021, 10:02:30 – 10:03:21)
Test is complete
```

Writable | Smart Insert | 31 : 30 : 1009

Orders Code

ersistence/domain/Orders.java - Eclipse IDE

t Run Window Help

OrderDAOTest.java | sql-data.sql | Orders.java | OrderDao.java

```java
public class Orders {

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

    private Long oID;
    private Long cID;
    private Customer customer;
    private double orderValue;
    private List<Items> items = new ArrayList<>();

    public Orders(Long oID, Customer customer, double orderValue, List<Items> items) {
        super();
        this.oID = oID;
        this.customer = customer;
        this.orderValue = orderValue;
        this.items = items;
    }

    public Orders(Long oID, Customer customer, double orderValue) {
        super();
        this.oID = oID;
        this.customer = customer;
        this.orderValue = orderValue;
    }

    public Orders(Customer customer, double orderValue) {
        super();
        this.customer = customer;
        this.orderValue = orderValue;
    }

    public List<Items> showItems() {
        for (Items item : items) {
            LOGGER.info(item);
        }
```

Console | Terminal | Coverage

minated> java [JUnit] C:\Program Files\AdoptOpenJDK\jdk-11.0.9.101-hotspot\bin\javaw.exe (29 Jan 2021, 10:02:30 – 10:03:21

t is complete

Writable | Smart Insert | 10 : 2 : 188

```java
    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append("Orders [oID=");
        builder.append(oID);
        builder.append(", cID=");
        builder.append(customer.getId());
        builder.append(", customer=");
        builder.append(customer.getFirstName());
        builder.append(" ");
        builder.append(customer.getSurname());
        builder.append(", orderValue=");
        builder.append(orderValue);
        builder.append(", items=");
        builder.append(items);
        builder.append("]");
        return builder.toString();
    }
```

OrderDAOTest.java | sql-data.sql | Orders.java | OrderDao.java

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

    public OrderDao(ItemDao itemDao, CustomerDao customerDao) {
        super();
        this.itemDao = itemDao;
        this.customerDao = customerDao;
    }

    public List<Items> getAllItems(Long oID){
        List<Long> itemIds = new ArrayList<>();
        try(Connection connection = DatabaseUtilities.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("SELECT * FROM orderline WHERE fk_oid = ?");){
                            statement.setLong(1, oID);
                            try (ResultSet resultSet = statement.executeQuery();){
                                while (resultSet.next()) {
                                    itemIds.add(resultSet.getLong("fk_iid"));
                                }

                            }

        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());


        }

        return itemIds.stream().map(itemId -> itemDao.read(itemId)).collect(Collectors.toList());
    }
```

Sprint Review

So far I have completed all the user stories that I had created. The only ones I have left are the presentation stories and as well as that to add the README file to my documentation.

Sprint Retrospective

What went well:

- I was able to take on board the teachings provided throughout the past few weeks and implement them, such as using if statements and loops.
- Was able to learn how to use J unit and implement that in making tests.
- Thoroughly designed the code and managed to use logic to solve issues that I had came across.

What did not go as planned:

- Being unable to complete orderDaoTest.
- My sonarQube crashed and I have been unable to use that to provide details on how I improved parts of my code or provide snapshots.
- Not constantly creating commits.

Conclusion

To conclude, I have thoroughly enjoyed this database challenged as it has pointed out my strengths and weaknesses and where I need to develop to build myself as a successful consultant. The project showed me my use of logic when trying to understand a difficult process has become significantly enhanced and I am able to derive solutions when I cannot find the simplest path. As well as this, my knowledge of Java, sql and J unit have strengthened allowing me to rise another step further closer to a high level consultant.

In the future, I believe that I can improve upon my timekeeping skills, making sure all the small details like committing constantly throughout the project, as well as completing all the simpler tasks must come first.

Thank you for your time.