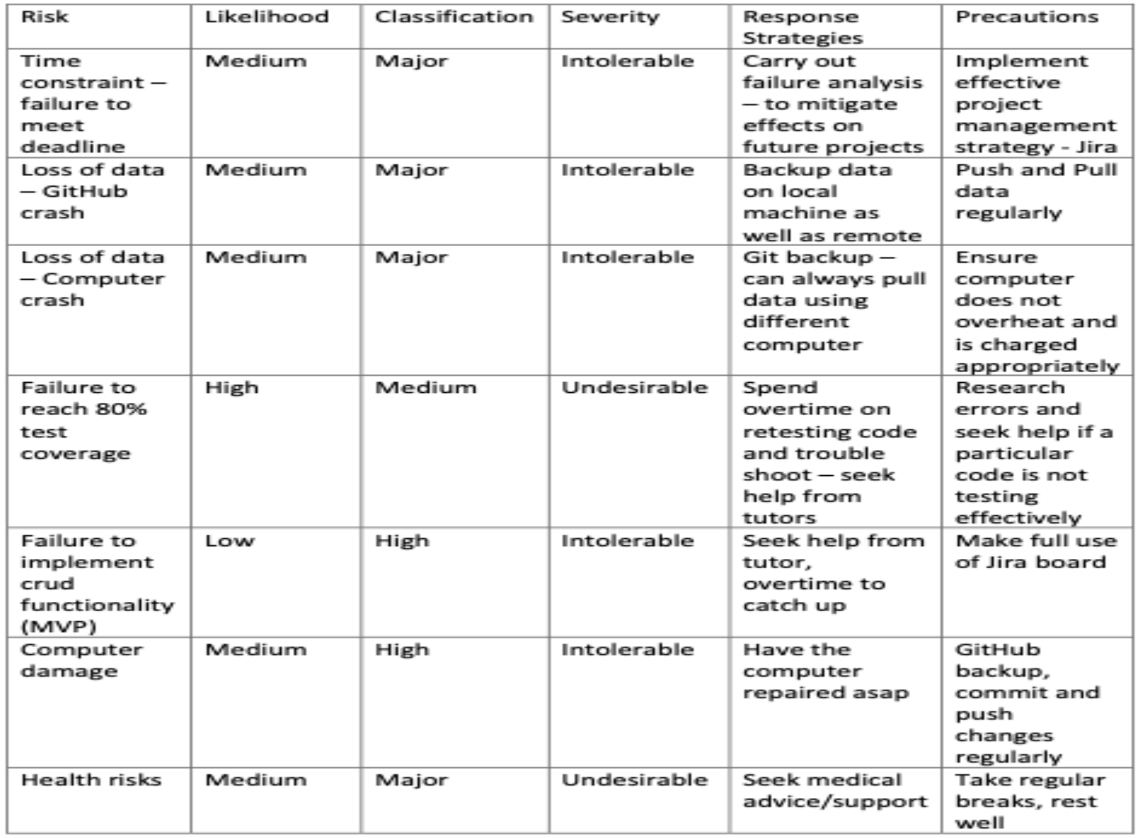
FUNDAMENTAL PROJECT: INVENTORY MANAGEMENT SYSTEM

By Abdirizak Osman

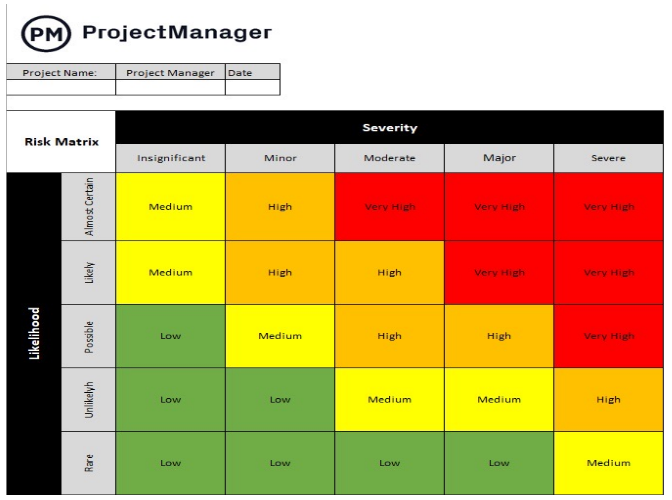
INTRODUCTION

- At the time of completing of this project, I have undertaken 4 weeks of effective training as a QA IT Consultant Trainee
- I have had hands on practical training with Java, Git, and MySQL, and have proven to learn under fast paced environment
- I have really gained great knowledge from this project, mainly from troubleshooting coding errors, and being able to research from websites such as Stack Overflow

RISK ASSESSMENT



Risk	Likelihood	Classification	Severity	Response Strategies	Precautions
Time constraint – failure to meet deadline	Medium	Major	Intolerable	Carry out failure analysis – to mitigate effects on future projects	Implement effective project management strategy - Jira
Loss of data – GitHub crash	Medium	Major	Intolerable	Backup data on local machine as well as remote	Push and Pull data regularly
Loss of data – Computer crash	Medium	Major	Intolerable	Git backup – can always pull data using different computer	Ensure computer does not overheat and is charged appropriately
Failure to reach 80% test coverage	High	Medium	Undesirable	Spend overtime on retesting code and trouble shoot – seek help from tutors	Research errors and seek help if a particular code is not testing effectively
Failure to implement crud functionality (MVP)	Low	High	Intolerable	Seek help from tutor, overtime to catch up	Make full use of Jira board
Computer damage	Medium	High	Intolerable	Have the computer repaired asap	GitHub backup, commit and push changes regularly
Health risks	Medium	Major	Undesirable	Seek medical advice/support	Take regular breaks, rest well

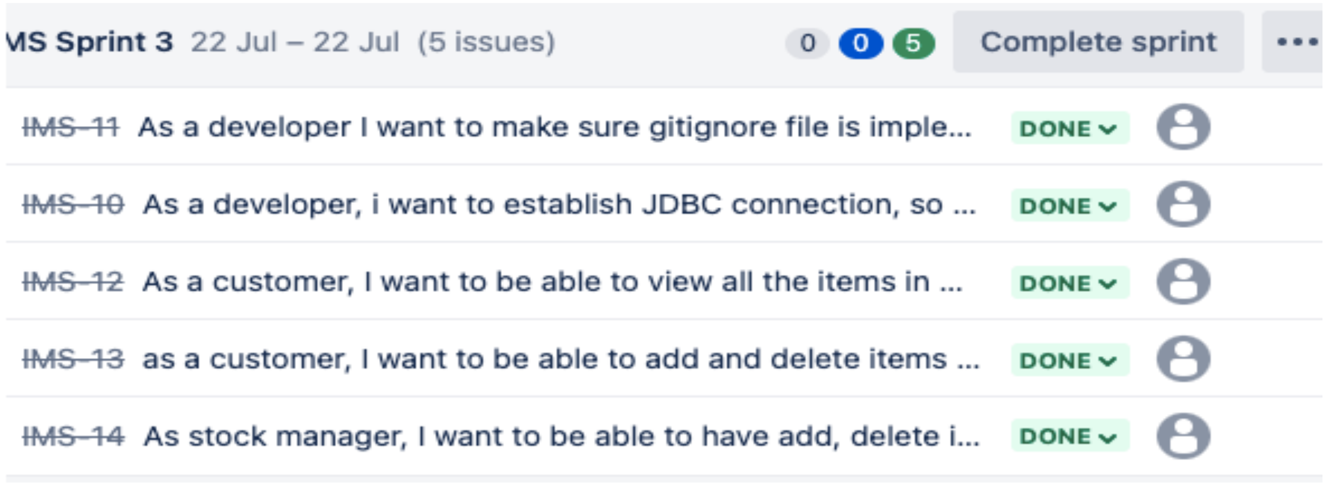


ProjectManager

Project Name:	Project Manager	Date

Risk Matrix		Severity				
		Insignificant	Minor	Moderate	Major	Severe
Likelihood	Almost Certain	Medium	High	Very High	Very High	Very High
	Likely	Medium	High	High	Very High	Very High
	Possible	Low	Medium	High	High	Very High
	Unlikelyh	Low	Low	Medium	Medium	High
	Rare	Low	Low	Low	Low	Medium

PROJECT MANAGEMENT
JIRA KANBAN BOARD



MS Sprint 3 22 Jul – 22 Jul (5 issues) 0 0 5 Complete sprint •••

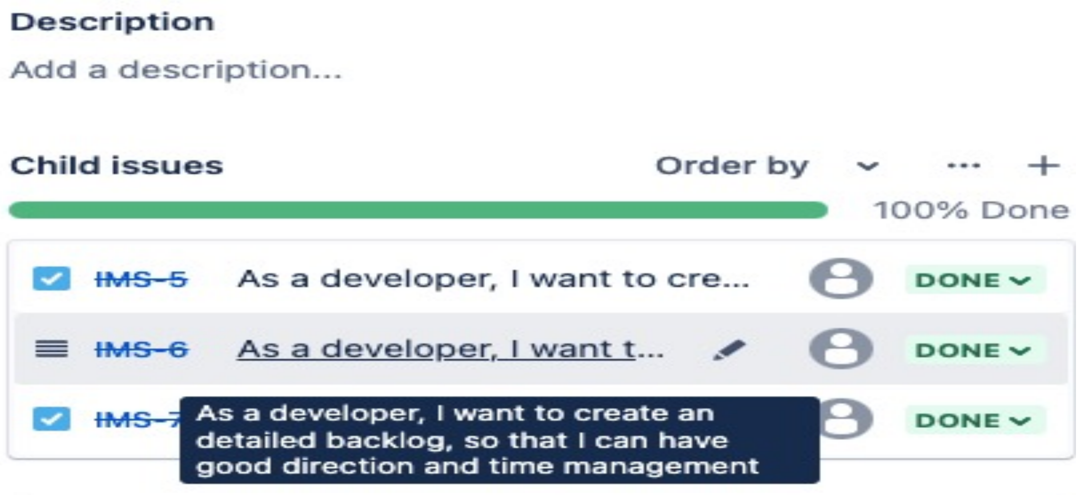
~~IMS-11~~ As a developer I want to make sure gitignore file is imple... DONE

~~IMS-10~~ As a developer, i want to establish JDBC connection, so ... DONE

~~IMS-12~~ As a customer, I want to be able to view all the items in ... DONE

~~IMS-13~~ as a customer, I want to be able to add and delete items ... DONE

~~IMS-14~~ As stock manager, I want to be able to have add, delete i... DONE



Description

Add a description...

Child issues Order by ••• +

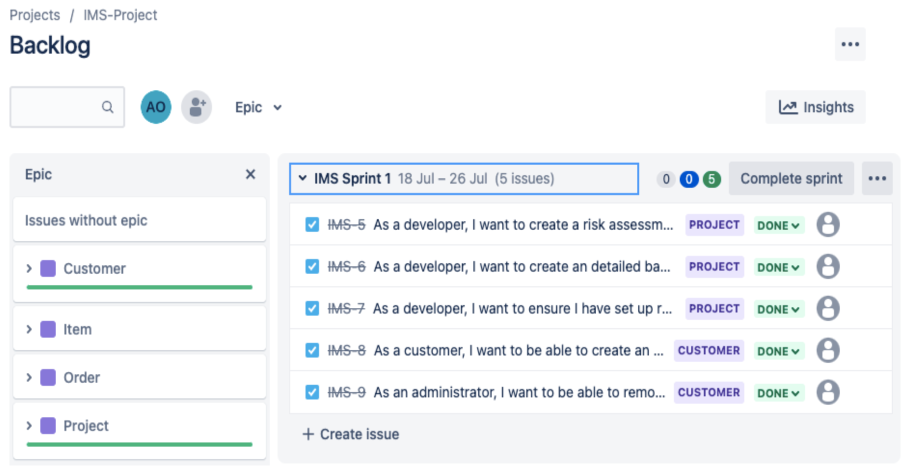
100% Done

~~IMS-5~~ As a developer, I want to cre... DONE

~~IMS-6~~ As a developer, I want t... DONE

~~IMS-7~~ DONE

As a developer, I want to create an detailed backlog, so that I can have good direction and time management



Projects / IMS-Project

Backlog

AO Epic Insights

Epic

Issues without epic

Customer

Item

Order

Project

IMS Sprint 1 18 Jul – 26 Jul (5 issues) 0 0 5 Complete sprint

~~IMS-5~~ As a developer, I want to create a risk assessm... PROJECT DONE

~~IMS-6~~ As a developer, I want to create an detailed ba... PROJECT DONE

~~IMS-7~~ As a developer, I want to ensure I have set up r... PROJECT DONE

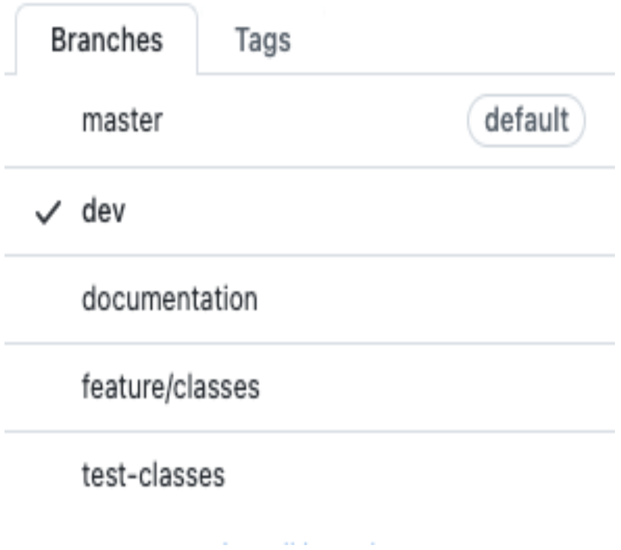
~~IMS-8~~ As a customer, I want to be able to create an ... CUSTOMER DONE

~~IMS-9~~ As an administrator, I want to be able to remo... CUSTOMER DONE

+ Create issue

VERSION CONTROL

- Initially generated repository from IMS-Starter
- Set up remote repository using Feature Branch Model
- Regularly pushed commits as data loss precaution

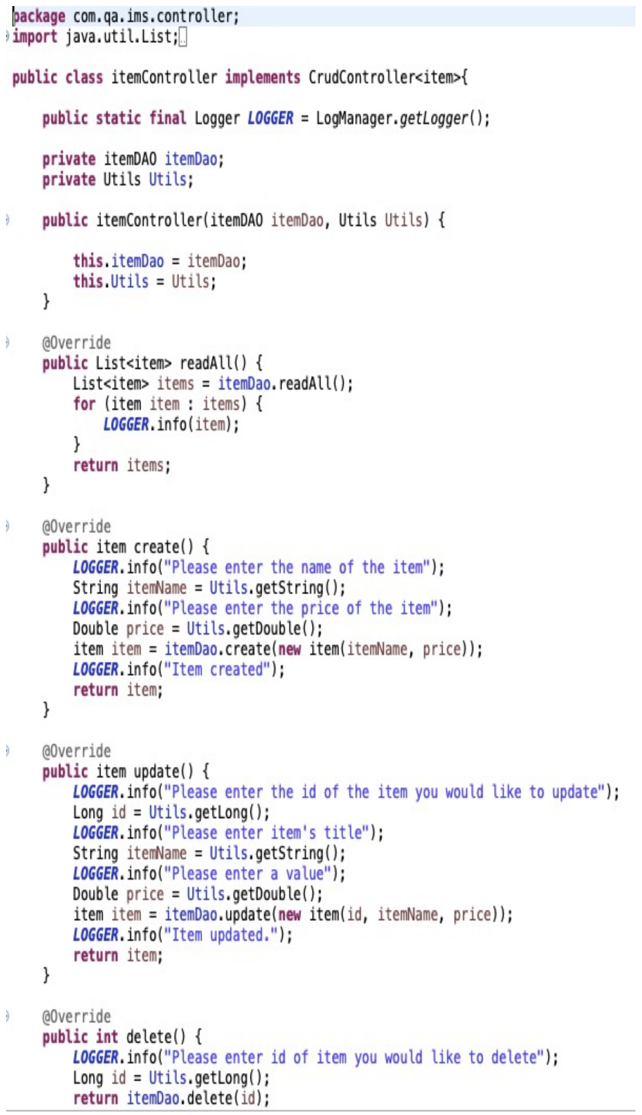


DEVELOPMENT MVP

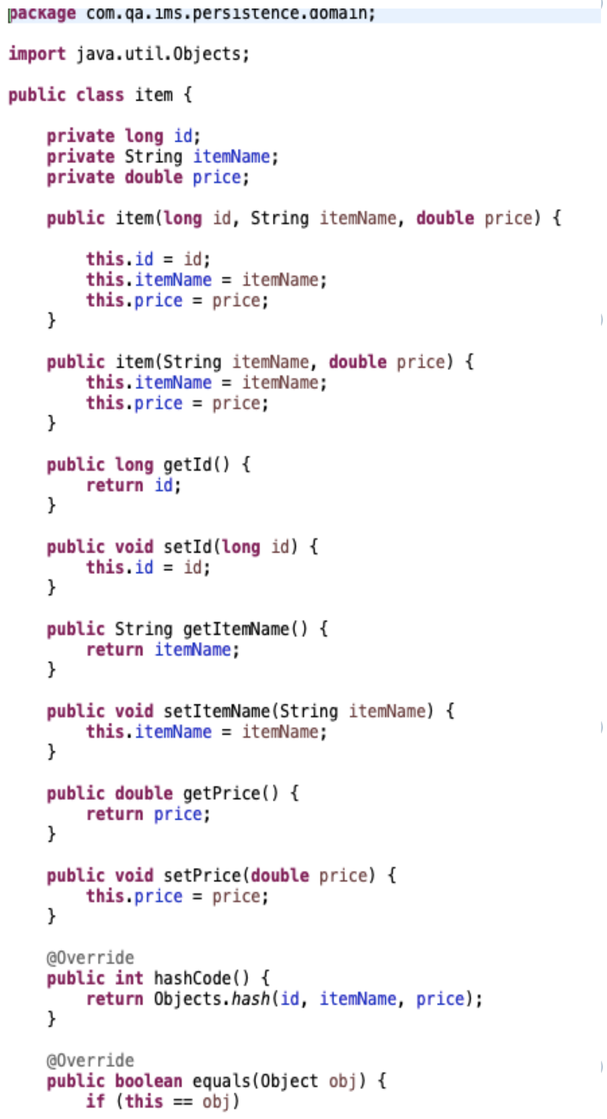
I spent much overtime over the weekend revising on core design principles, analyzing the project spec – allowed me to start and get some progress by Tuesday

Crud functionality was tested through the CLI in Eclipse, was really impressed as functionality was achieved

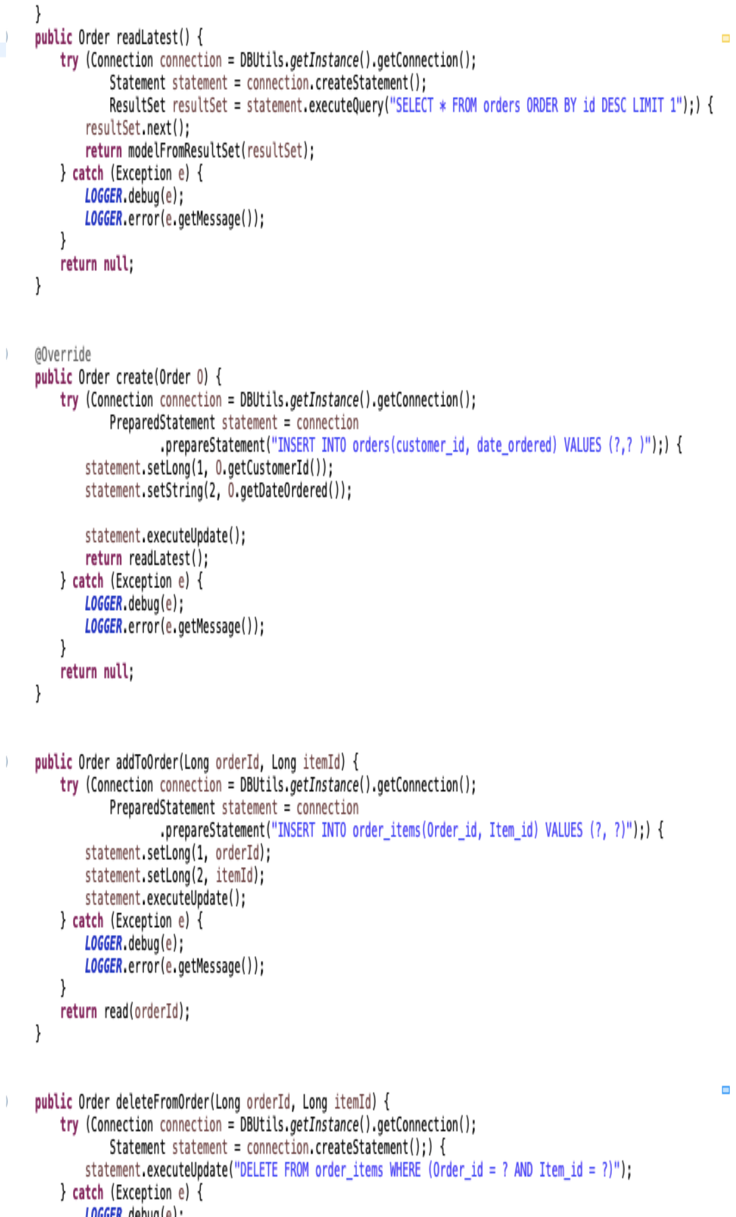
CLASSES



```java
package com.qa.ims.controller;
import java.util.List;

public class itemController implements CrudController<item>{

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

    private itemDAO itemDao;
    private Utils Utils;

    public itemController(itemDAO itemDao, Utils Utils) {

        this.itemDao = itemDao;
        this.Utils = Utils;
    }

    @Override
    public List<item> readAll() {
        List<item> items = itemDao.readAll();
        for (item item : items) {
            LOGGER.info(item);
        }
        return items;
    }

    @Override
    public item create() {
        LOGGER.info("Please enter the name of the item");
        String itemName = Utils.getString();
        LOGGER.info("Please enter the price of the item");
        Double price = Utils.getDouble();
        item item = itemDao.create(new item(itemName, price));
        LOGGER.info("Item created");
        return item;
    }

    @Override
    public item update() {
        LOGGER.info("Please enter the id of the item you would like to update");
        Long id = Utils.getLong();
        LOGGER.info("Please enter item's title");
        String itemName = Utils.getString();
        LOGGER.info("Please enter a value");
        Double price = Utils.getDouble();
        item item = itemDao.update(new item(id, itemName, price));
        LOGGER.info("Item updated.");
        return item;
    }

    @Override
    public int delete() {
        LOGGER.info("Please enter id of item you would like to delete");
        Long id = Utils.getLong();
        return itemDao.delete(id);
```

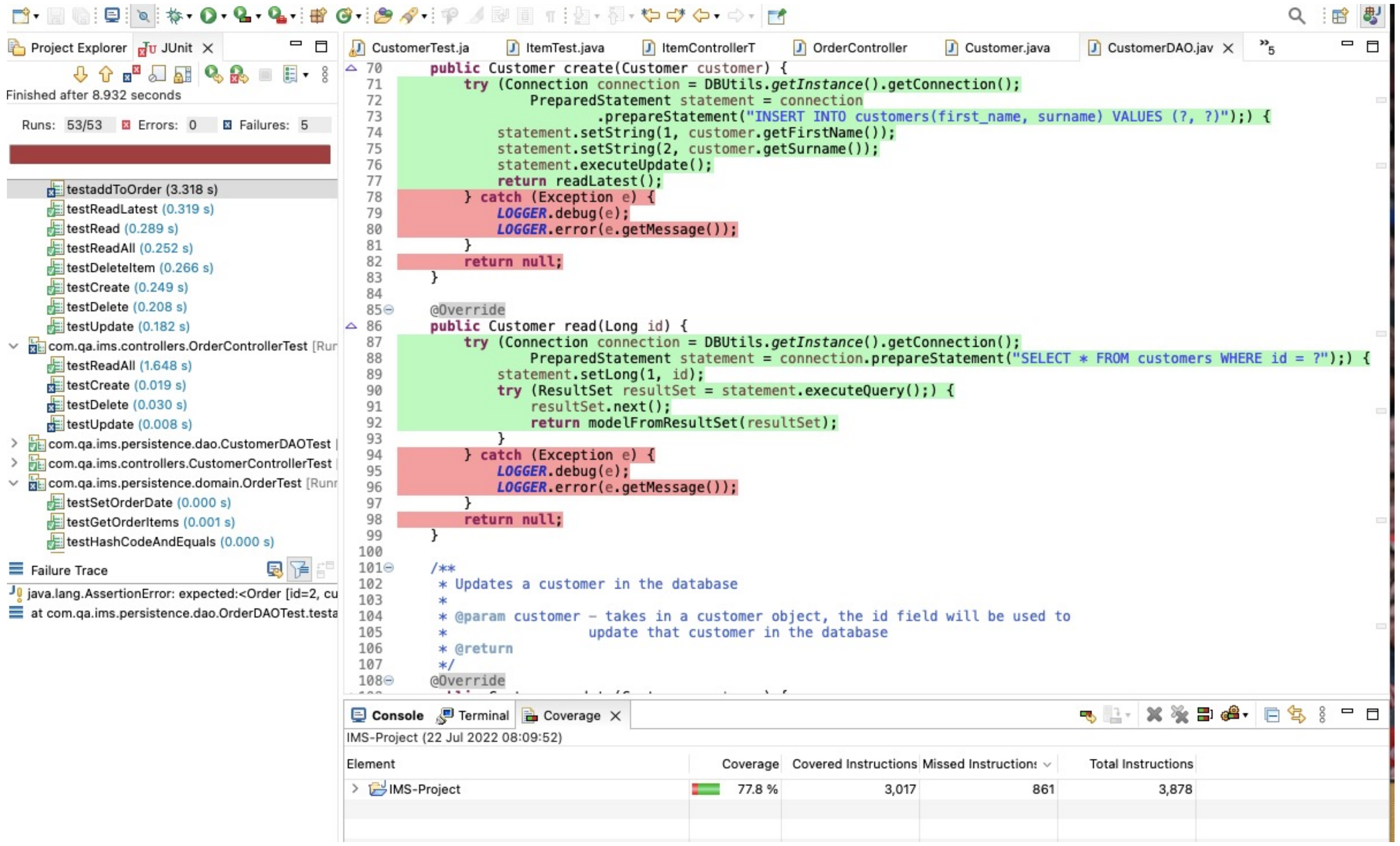


```java
package com.qa.ims.persistence.domain;

import java.util.Objects;

public class item {

    private long id;
    private String itemName;
    private double price;

    public item(long id, String itemName, double price) {

        this.id = id;
        this.itemName = itemName;
        this.price = price;
    }

    public item(String itemName, double price) {
        this.itemName = itemName;
        this.price = price;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getItemName() {
        return itemName;
    }

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

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, itemName, price);
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
```



```java
}
public Order readLatest() {
    try (Connection connection = DBUtils.getInstance().getConnection();
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM orders ORDER BY id DESC LIMIT 1");) {
        resultSet.next();
        return modelFromResultSet(resultSet);
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
}


@Override
public Order create(Order O) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection
                    .prepareStatement("INSERT INTO orders(customer_id, date_ordered) VALUES (?,? )");) {
        statement.setLong(1, O.getCustomerId());
        statement.setString(2, O.getDateOrdered());

        statement.executeUpdate();
        return readLatest();
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
}


public Order addToOrder(Long orderId, Long itemId) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection
                    .prepareStatement("INSERT INTO order_items(Order_id, Item_id) VALUES (?, ?)");) {
        statement.setLong(1, orderId);
        statement.setLong(2, itemId);
        statement.executeUpdate();
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return read(orderId);
}


public Order deleteFromOrder(Long orderId, Long itemId) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            Statement statement = connection.createStatement();) {
        statement.executeUpdate("DELETE FROM order_items WHERE (Order_id = ? AND Item_id = ?)");
    } catch (Exception e) {
        LOGGER.debug(e);
```

TESTING COVERAGE: 77%



Project Explorer | JUnit

Finished after 8.932 seconds

Runs: 53/53 Errors: 0 Failures: 5

```
    testaddToOrder (3.318 s)
    testReadLatest (0.319 s)
    testRead (0.289 s)
    testReadAll (0.252 s)
    testDeleteItem (0.266 s)
    testCreate (0.249 s)
    testDelete (0.208 s)
    testUpdate (0.182 s)
v com.qa.ims.controllers.OrderControllerTest [Rur
    testReadAll (1.648 s)
    testCreate (0.019 s)
    testDelete (0.030 s)
    testUpdate (0.008 s)
> com.qa.ims.persistence.dao.CustomerDAOTest
> com.qa.ims.controllers.CustomerControllerTest
v com.qa.ims.persistence.domain.OrderTest [Runr
    testSetOrderDate (0.000 s)
    testGetOrderItems (0.001 s)
    testHashCodeAndEquals (0.000 s)
```

Failure Trace

```
java.lang.AssertionError: expected:<Order [id=2, cu
at com.qa.ims.persistence.dao.OrderDAOTest.testa
```

CustomerTest.ja | ItemTest.java | ItemControllerT | OrderController | Customer.java | CustomerDAO.jav

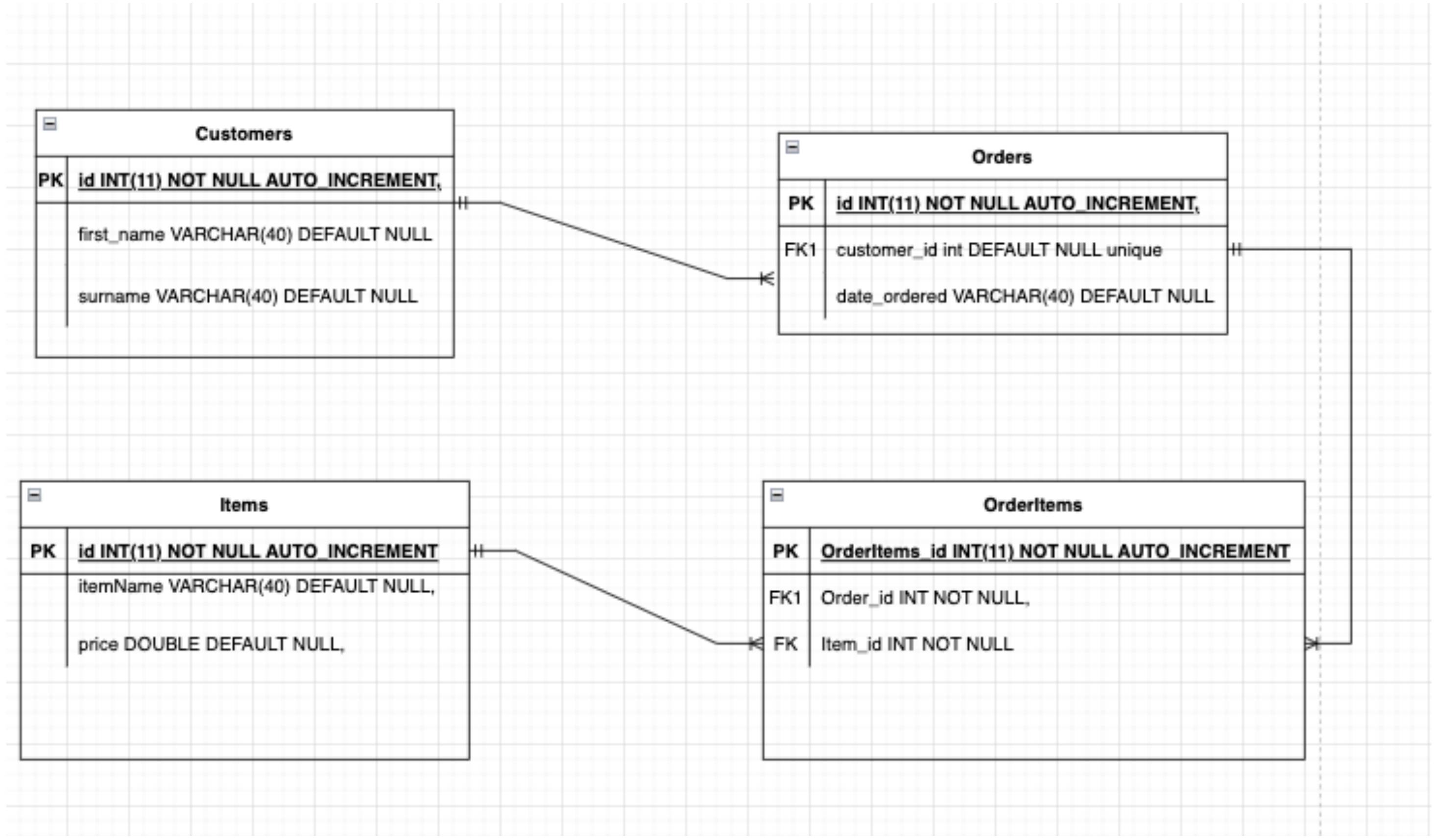
```java
	public Customer create(Customer customer) {
		try (Connection connection = DBUtils.getInstance().getConnection();
				PreparedStatement statement = connection
						.prepareStatement("INSERT INTO customers(first_name, surname) VALUES (?, ?)");) {
			statement.setString(1, customer.getFirstName());
			statement.setString(2, customer.getSurname());
			statement.executeUpdate();
			return readLatest();
		} catch (Exception e) {
			LOGGER.debug(e);
			LOGGER.error(e.getMessage());
		}
		return null;
	}

	@Override
	public Customer read(Long id) {
		try (Connection connection = DBUtils.getInstance().getConnection();
				PreparedStatement statement = connection.prepareStatement("SELECT * FROM customers WHERE id = ?");) {
			statement.setLong(1, id);
			try (ResultSet resultSet = statement.executeQuery();) {
				resultSet.next();
				return modelFromResultSet(resultSet);
			}
		} catch (Exception e) {
			LOGGER.debug(e);
			LOGGER.error(e.getMessage());
		}
		return null;
	}

	/**
	 * Updates a customer in the database
	 * 
	 * @param customer - takes in a customer object, the id field will be used to
	 *                 update that customer in the database
	 * @return
	 */
	@Override
```

Console | Terminal | Coverage

IMS-Project (22 Jul 2022 08:09:52)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
IMS-Project	77.8 %	3,017	861	3,878

ERD DIAGRAM


Customers
PK id INT(11) NOT NULL AUTO_INCREMENT,
first_name VARCHAR(40) DEFAULT NULL
surname VARCHAR(40) DEFAULT NULL
Orders
PK id INT(11) NOT NULL AUTO_INCREMENT,
FK1 customer_id int DEFAULT NULL unique
date_ordered VARCHAR(40) DEFAULT NULL
Items
PK id INT(11) NOT NULL AUTO_INCREMENT
itemName VARCHAR(40) DEFAULT NULL,
price DOUBLE DEFAULT NULL,
OrderItems
PK OrderItems_id INT(11) NOT NULL AUTO_INCREMENT
FK1 Order_id INT NOT NULL,
FK Item_id INT NOT NULL

CONCLUSION

- I really enjoyed coding more than I could have imagined just from this week, the satisfaction of fixing errors through googling error codes was tedious at times but was worth it once I figured out the reason behind it

- Planning and time constraint: I really underestimated the time I would have spent on testing, as my Jira board shows – testing my controller classes really slowed me down as I had to revise on Mockito testing

- Unfinished: Building and compiling the code still not clear

HAPPY TO TAKE ANY QUESTIONS

- Special Thank you to my tutors; Jordan, Aswene, Andrew, Piers, and Ed, who have been very supportive during my training period before the project week