IMS-STARTER – WEEK 5 EXERCISE

ELINA MCGLINCHEY – 22APRENABLE3
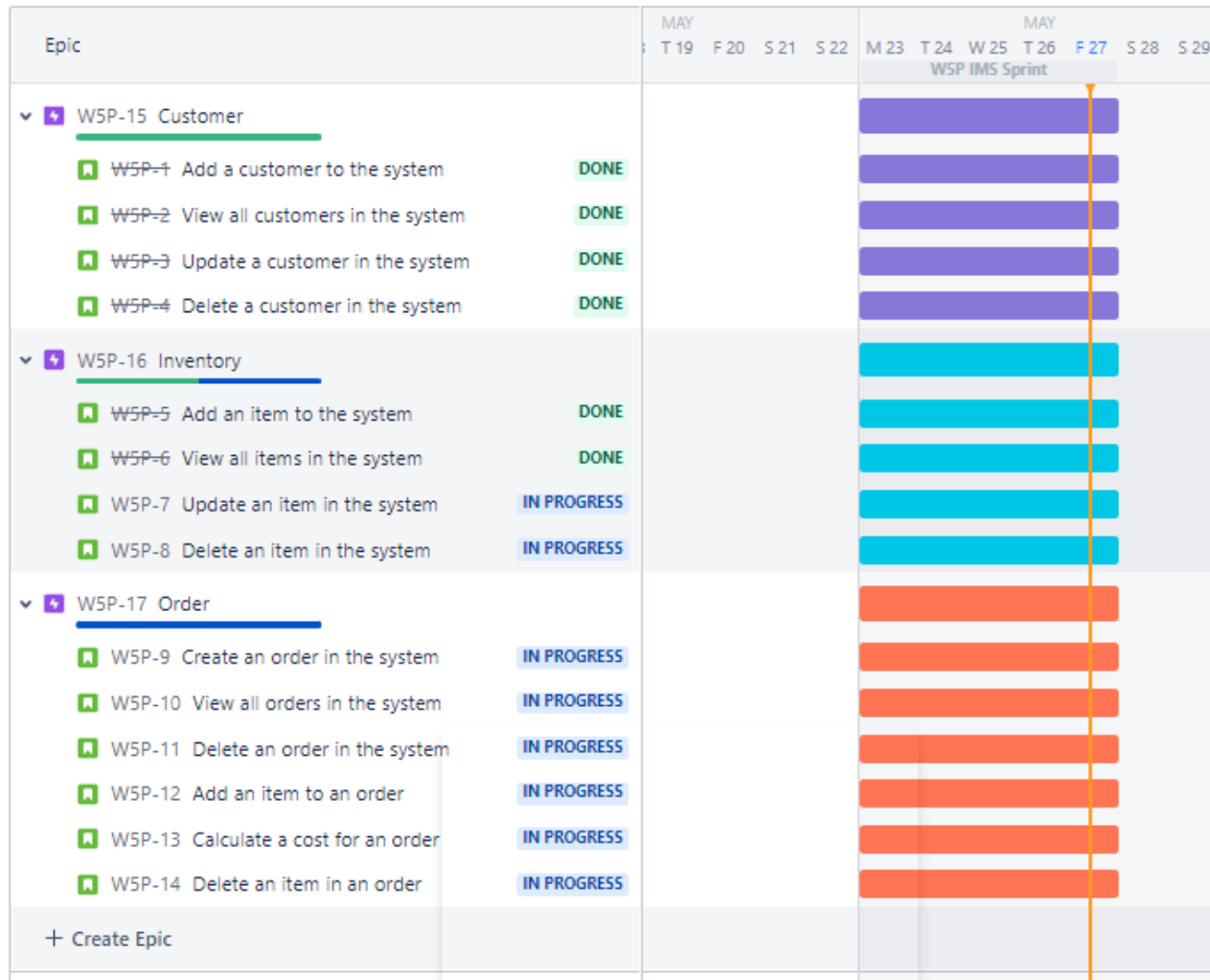
WEEK TIMETABLE

	DAY 1	DAY 2	DAY 3	DAY 4
STEP 1	Read through task and made notes (am)	Started reading through and assessing IMS-Starter code (am)	Copied and pasted missing files back from Github to Eclipse (am)	Copied IMS-Starter code in file explorer to keep safe before pushing to Github (am)
STEP 2	Forked project (am)	Began making changes to code – username and password (am)	Began DAO and Controller tests (am)	Pushed to Github but experienced issues – debugged connection issues (am)
STEP 3	Created Jira (am)	Added Item, Order and OrderItems (am)	Pushed to Github and files disappeared, needed to retrieve (am)	Created imstest database (am)
STEP 4	Created EER Diagram (am)	Continued to push up to Github – trouble (am)	Began testing Customer and Item DAOTests and ControllerTests – failures (am)	Ran further tests and debugged (am and pm)
STEP 5	Created Risk Assessment (am)	Debugging all afternoon and retrieving missing code after git push (pm)	Pushed to Github – lost files and needed to retrieve (pm)	Created Order DAOTest and ControllerTest – got failures 20/33 (pm)
STEP 6	Set up IMS database SQL (pm)		Debugging tests in code (pm)	Debugging for better connection – databases not found, app partially working (pm)
STEP 7	Create tables and set relationships (pm)			Merge to main (NEED TO DO)

JIRA Management

https://elinamcglinchey.atlassian.net/jira/software/projects/W5P/boards/1/roadmap?timeline=WEEKS

https://github.com/elinamcglinchey/IMS-Starter

What went wrong?
1. Time 2. Technical Issues 3. Pulling to early on Github branches

EER DIAGRAM

Diagram 1 – Day 1

Orders	
OrderID	int (FK)
CustomerID	int (FK)

Customer	
id	int (PK)
firstName	varchar
surname	varchar
username	varchar
password	varchar

Order Items	
orderItemID	int (PK)
cost	float
name	varchar
itemID	int (FK)

Items	
id	int (PK)
cost	float
name	varchar

Diagram 2 – Day 4

customers
- id INT
- First_name VARCHAR(40)
- Last_name VARCHAR(40)
- Username VARCHAR(40)
- Password VARCHAR(40)
- Indexes

orders
- orderItem ID INT
- customerID INT
- cost DOUBLE
- items_id INT
- items_id1 INT
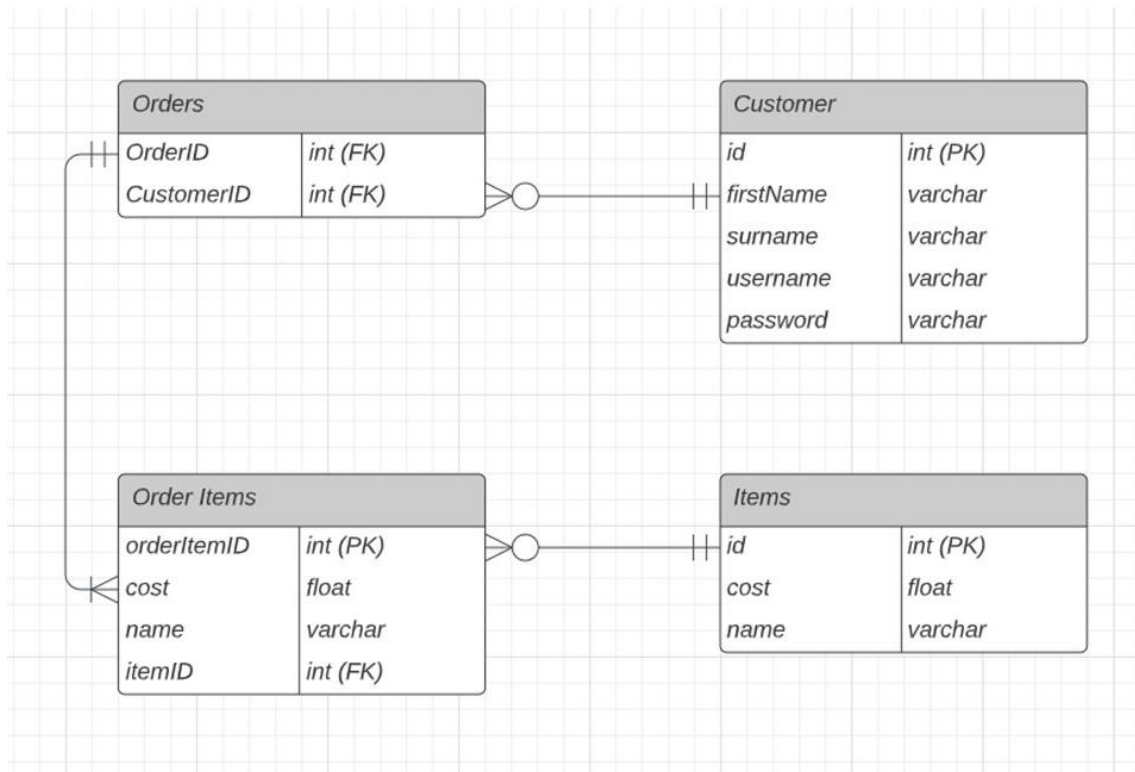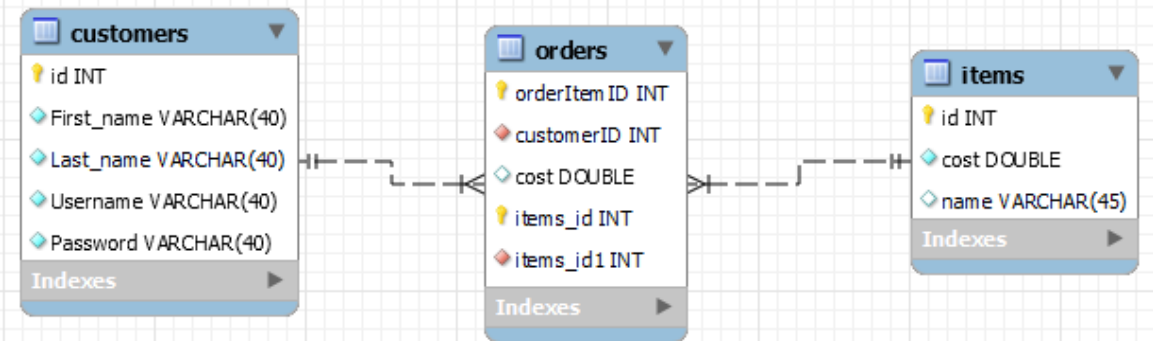- Indexes

items
- id INT
- cost DOUBLE
- name VARCHAR(45)
- Indexes

- Changed int -> Long
- Changed Float -> Double
- Removed OrderItems – didn't deem necessary

CODEBASE

- Partially working app
- Zero errors in tests yet only partially passing
- Demo

```
ITEM: Individual Items
ORDER: Purchases of items
STOP: To close the application
CUSTOMER
What would you like to do with customer:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
CREATE
Please enter a first name
ELINA
Please enter a surname
MCGLINCHEY
Please enter a username
ELINAMCGLINCHEY
Please enter a password
ELINA123
Customer created
What would you like to do with customer:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
READ
Customer [id=1, firstName=elina, surname=mcglin, username=mcslgdi, password=884395]
Customer [id=2, firstName=ELINA, surname=MCGLINCHEY, username=ELINAMCGLINCHEY, password=ELINA123]
What would you like to do with customer:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
```

Runs: 33/33   Errors: 0   Failures: 19

> com.qa.ims.controllers.CustomerControllerTest [Runner: JUnit 4] (5.762 s)
> com.qa.ims.persistence.domain.CustomerTest [Runner: JUnit 4] (0.845 s)
> com.qa.ims.persistence.domain.ItemTest [Runner: JUnit 4] (0.024 s)
> com.qa.ims.persistence.domain.OrderTest [Runner: JUnit 4] (0.024 s)
> com.qa.ims.controllers.ItemControllerTest [Runner: JUnit 4] (0.134 s)
> com.qa.ims.persistence.dao.OrderDAOTest [Runner: JUnit 4] (3.905 s)
> com.qa.ims.controllers.OrderControllerTest [Runner: JUnit 4] (0.088 s)
> com.qa.ims.persistence.dao.CustomerDAOTest [Runner: JUnit 4] (1.930 s)
> com.qa.ims.persistence.dao.ItemDAOTest [Runner: JUnit 4] (3.028 s)

```
STOP: To close the application
ITEM
What would you like to do with item:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
CREATE
Exception in thread "main" java.lang.NullPointerException: Cannot i
        at com.qa.ims.IMS.doAction(IMS.java:78)
        at com.qa.ims.IMS.domainAction(IMS.java:70)
        at com.qa.ims.IMS.imsSystem(IMS.java:38)
        at com.qa.ims.Runner.main(Runner.java:12)
```

CODEBASE

- Partially working app
- https://github.com/elinamcglinchey/IMS-Starter
- Summary

CODEBASE

- Partially working app
- Zero errors in tests yet only partially passing
- Summary

```java
 1  package com.qa.ims.controllers;
 2
 3  import static org.junit.Assert.assertEquals;□
20
21  @RunWith(MockitoJUnitRunner.class)
22  public class ItemControllerTest {
23
24      @Mock
25      private Utils utils;
26
27      @Mock
28      private ItemDAO dao;
29
30      @InjectMocks
31      private ItemController controller;
32
33      @Test
34      public void testCreate() {
35          //final String ITEM_ID = "4", ITEM_COST = "0.65", ITEM_NAME = "pencil";
36          final Long ITEM_ID = 1L;
37          final Double ITEM_COST = 0.65;
38          final String ITEM_NAME = "pencil";
39          final Item created = new Item(ITEM_ID, ITEM_COST, ITEM_NAME);
40
41          //final String F_NAME = "barry", L_NAME = "scott", U_NAME = "scottyb123", P_WORD
42          //final Customer created = new Customer(F_NAME, L_NAME, U_NAME, P_WORD);
43
44
45          Mockito.when(utils.getLong()).thenReturn(ITEM_ID);
46          Mockito.when(utils.getDouble()).thenReturn(ITEM_COST);
47          Mockito.when(utils.getString()).thenReturn(ITEM_NAME);
48          //Mockito.when(utils.getString()).thenReturn(ITEM_ID, ITEM_COST, ITEM_NAME);
49          // Mockito.when(utils.getString()).thenReturn(ITEM_NAME);
50          Mockito.when(dao.create(created)).thenReturn(created);
51
52          assertEquals(created, controller.create());
53
52          assertEquals(created, controller.create());
53
54          Mockito.verify(utils, Mockito.times(1)).getString();
55          Mockito.verify(dao, Mockito.times(1)).create(created);
56      }
57
58      @Test
59      public void testReadAll() { // add to sql data
60          List<Item> items = new ArrayList<>();
61          items.add(new Item(1L,0.35, "rubber"));
62
63          Mockito.when(dao.readAll()).thenReturn(items);
64
65          assertEquals(items, controller.readAll());
66
67          Mockito.verify(dao, Mockito.times(1)).readAll();
68      }
69
70      @Test
71      public void testUpdate() {
72          Item updated = new Item(1L, 5.99, "calculator");
73
74          Mockito.when(this.utils.getLong()).thenReturn(1L);
75          Mockito.when(this.utils.getDouble()).thenReturn(5.99);
76          Mockito.when(this.utils.getString()).thenReturn(updated.getName());
77          Mockito.when(this.dao.update(updated)).thenReturn(updated);
78
79          assertEquals(updated, this.controller.update());
80
81          Mockito.verify(this.utils, Mockito.times(1)).getLong();
82          Mockito.verify(this.utils, Mockito.times(1)).getDouble();
83          Mockito.verify(this.utils, Mockito.times(1)).getString();
84          Mockito.verify(this.dao, Mockito.times(1)).update(updated);
85      }
86
```

RISK ASSESSMENT
- Risk Matrix

Risks	Negligible	Minor	Major	Critical	Catastrophic
Time	Completed all tasks	Unable to do stretch goals	No time to do any successful application functions	No time to debug errors in tests	No time to do any necessary classes
Injury/Illness	Completely healthy	Cold, slightly anxious or stress	Flu/Stomach bug/Food Poisoning, anxiety	Miss majority of week due to injury or illness	Injury or illness – missing assessed week
Theft	No theft	Theft of other possessions related to laptop/pc	Theft of appliances necessary for task (mouse/keyboard)	Theft of laptop or pc	Theft of laptop or pc with no external drive or cloud upload
Software Corruption	No Corruption	Github/Gitbash failure	Complex version history	Files unable to load or lost	Files completed deleted
Github pushes (i.e pushed too early)	No github issues	Pull request on Github too early	Didn't separate branches	Merged to main too early	Corrupted main
Missed commit	No missed commits	Some missed commits	Big commits missed	Most commits missed	All commits missed
Poor implementation (lack of experience)	Complete understanding	Basic debugging issues	Lack of eclipse understanding	Barely have knowledge of any softwares	No understanding of any softwares

REFLECTION
- What went well?
- What did not go well?
- What can I improve for future projects?

Did go well	Did not go well
Worked independently to debug	Needed help on simple syntax errors missed ({ ;)) etc
Collaborated with peers in a team – helping each other	Failures in many tests
Extended Java, SQL and Git understanding	Occasionally needed help in debugging
Got some tests working with no prior Java knowledge	Git uploads

What can I improve for future projects?
- Dissect complete code and make pen and paper notes before beginning
- Revise Github, SQL and Gitbash skills (had forgotten many since not using since Week 1 or 2)
- Work more into the evenings
- Take small interval breaks to remove self from the screen during intense projects
- Practice making branches on Git
- Ask for more help in the beginning to avoid unnecessary loopholes
- Practice SQL, Git, Github and Java skills – look back on previous videos to remind and revise skills learned
- Make more notes

NEXT STEPS

What is left do to in this Sprint/Project?
- Finish off testing (attempt to get more successful tests)
- Debug DAO issues in SQL
- Run MVN clean package if I finish tests
- Ensure jar file with dependencies is in main repo NOT target
- Push to Github and merge dev to main
- Create a README

THANK YOU

ELINAMCGLINCHEY1@GMAIL.COM