

WEEKTIMETABLE							
	DAY I	DAY 2	DAY 3	DAY 4			
STEP I	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

Epic		MAY T 19	F 20	S 21	S 22	M 23	W 25 IMS Spr	F 27	S 28	S 29
✓ <u>▼</u> W5P-15 Customer										
■ W5P-1 Add a customer to the system	DONE									
■ <del>W5P-2</del> View all customers in the system	DONE									
■ <del>W5P-3</del> Update a customer in the system	DONE									
■ W5P-4 Delete a customer in the system	DONE									
▼ 5 W5P-16 Inventory										
■ <del>W5P-5</del> Add an item to the system	DONE									
■ W5P-6 View all items in the system	DONE									
■ W5P-7 Update an item in the system	IN PROGRESS									
■ W5P-8 Delete an item in the system	IN PROGRESS									
✓ M5P-17 Order										
■ W5P-9 Create an order in the system	IN PROGRESS									
■ W5P-10 View all orders in the system	IN PROGRESS									
■ W5P-11 Delete an order in the system	IN PROGRESS									
■ W5P-12 Add an item to an order	IN PROGRESS									
■ W5P-13 Calculate a cost for an order	IN PROGRESS									
■ W5P-14 Delete an item in an order	IN PROGRESS							ф		
+ Create Epic										

#### **➤ Backlog** (8 issues) IN PROGRESS V ■ W5P-7 Update an item in the system INVENTORY IN PROGRESS > ■ W5P-8 Delete an item in the system INVENTORY W5P-9 Create an order in the system **ORDER** IN PROGRESS V W5P-10 View all orders in the system **ORDER** IN PROGRESS V W5P-11 Delete an order in the system **ORDER** IN PROGRESS V W5P-12 Add an item to an order **ORDER** IN PROGRESS V W5P-13 Calculate a cost for an order ORDER IN PROGRESS V W5P-14 Delete an item in an order ORDER IN PROGRESS V

## What went wrong?

1. Time 2. Technical Issues 3. Pulling to early on Github branches

## **EER DIAGRAM**

Diagram I – Day I

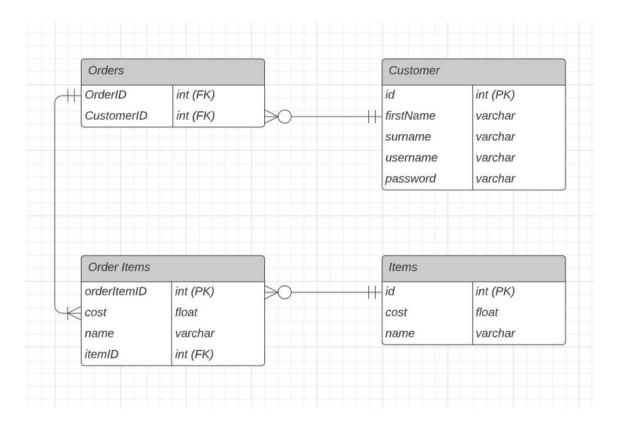
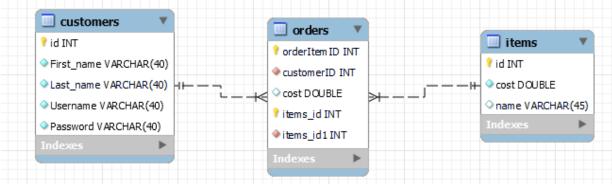


Diagram 2 – Day 4



- Changed int -> Long
- Changed Float -> Double
- Removed Orderltems 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
ELTNA123
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

> tom.ga.ims.controllers.CustomerControllerTest [Runner: JUnit 4] (5.762 s)
> com.ga.ims.persistence.domain.CustomerTest [Runner: JUnit 4] (0.845 s)
> com.qa.ims.persistence.domain.ltemTest [Runner: JUnit 4] (0.024 s)
> com.qa.ims.persistence.domain.OrderTest [Runner: JUnit 4] (0.024 s)
> com.ga.ims.controllers.ltemControllerTest [Runner: JUnit 4] (0.134 s)
> com.ga.ims.persistence.dao.OrderDAOTest [Runner: JUnit 4] (3.905 s)
> com.qa.ims.controllers.OrderControllerTest [Runner: JUnit 4] (0.088 s)
> com.ga.ims.persistence.dao.CustomerDAOTest [Runner: JUnit 4] (1.930 s)
> com.qa.ims.persistence.dao.ltemDAOTest [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
```

Exception in thread "main" java.lang.NullPointerException: Cannot i

at com.ga.ims.IMS.doAction(IMS.java:78)

at com.qa.ims.IMS.imsSystem(IMS.java:38) at com.qa.ims.Runner.main(Runner.java:12)

at com.ga.ims.IMS.domainAction(IMS.java:70)

CREATE

# CODEBASE

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

Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions		
✓    IMS-Starter  IMS-Start  IMS-Starter  IMS-Starter  IMS-Starter  IMS-Starter  IMS-Start	68.9 %	2,415	1,090	3,505	1 •	<pre>INSERT INTO `ims`.`customers` (`first name`, `last name`, `username`, `password`) VALUES ('jordan', 'harrison', 'jharrisson', 'j1h2344');</pre>
✓	63.1 %	1,519	888	2,407		INSERT INTO `ims`.`items` (`id`, `cost`, `name`) VALUES ('1', 0.35, 'rubber');
<ul> <li>tom.qa.ims.persistence.domain</li> </ul>	58.8 %	429	301	730	3 •	<pre>INSERT INTO `ims`.`orders` (`orderitemID`, `customerID`, `cost`) VALUES ('23', '1231', 88.90);</pre>
> 🗾 Domain.java	0.0 %	0	105	105		
> 🚺 ltem.java	49.7 %	88	89	177	1 •	drop database ims;
> 🚺 Order.java	50.8 %	90	87	177	2 •	CREATE DATABASE IF NOT EXISTS 'ims';
> 🚺 Customer.java	92.6 %	251	20	271	3 •	USE `ims` ;
<ul> <li>tom.qa.ims.persistence.dao</li> </ul>	68.8 %	545	247	792	4	
> 🚺 CustomerDAO.java	68.5 %	191	88	279	5 •	
> 🚺 OrderDAO.java	68.6 %	175	80	255	6	'id' INT(11) NOT NULL AUTO_INCREMENT,
> 🚺 ItemDAO.java	69.4 %	179	79	258	7	`first_name` VARCHAR(40) DEFAULT NULL,
temperature temperatur	0.0 %	0	134	134	8	`last_name` VARCHAR(40) DEFAULT NULL,
> 🚺 IMS.java	0.0 %		118	118		`username` VARCHAR(40) DEFAULT NULL,
> 🚺 Runner.java	0.0 %		16		10	`password` VARCHAR(40) DEFAULT NULL,
com.qa.ims.controller	75.9 %		119	494		PRIMARY KEY (`id`)
> 🚺 Action.java	0.0 %		119	119		);
> 🚺 CustomerController.java	100.0 %		0	141		
> 🚺 ItemController.java	100.0 %		0			○ CREATE TABLE IF NOT EXISTS `ims`.`items` (
> 🚺 OrderController.java	100.0 %		0	117		`id` INT(11) NOT NULL AUTO_INCREMENT,
> 🌐 com.qa.ims.utils	68.5 %		78	248		`cost` INT(100) DEFAULT NULL,
> # com.qa.ims.exceptions	0.0 %		9		17	`name` VARCHAR(40) DEFAULT NULL,
✓	81.6 %		202	1,098		PRIMARY KEY (`id`)
→ ⊕ com.qa.ims.persistence.dao	53.8 %		171	370		· );
> 🚺 CustomerDAOTest.java	51.7 %		58	120		
> 🚺 OrderDAOTest.java	57.8 %		57		21 •	
> / ItemDAOTest.java	51.3 %		56	115	22 •	
→	95.6 %		31	704		orderID int not null AUTO_INCREMENT,
> 🚺 OrderControllerTest.java	87.1 %		31	240		customerID int not null,
>	100.0 %		0	234		cost double not null,
> 1 ItemControllerTest.java	100.0 %		0	230		primary key (orderID),
→   ⊕ com.qa.ims.persistence.domain	100.0 %		0	24		foreign key(customerID) references customers(id) on delete cascade on update cascade
> La CustomerTest.java	100.0 %		0		28	);
> 🔃 ItemTest.java	100.0 %		0	8	29	
> 📝 OrderTest.java	100.0 %	8	0	8		

#### CODEBASE

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

53

```
package com.qa.ims.controllers;
                                                                                             52
                                                                                                         assertEquals(created, controller.create());
                                                                                             53
 3⊕ import static org.junit.Assert.assertEquals; □
                                                                                             54
                                                                                                         Mockito.verify(utils, Mockito.times(1)).getString();
20
                                                                                             55
                                                                                                         Mockito.verify(dao, Mockito.times(1)).create(created);
21 @RunWith (MockitoJUnitRunner.class)
                                                                                             56
22 public class ItemControllerTest {
                                                                                             57
23
                                                                                             58⊖
                                                                                                     @Test
24⊖
        @Mock
                                                                                             59
                                                                                                    public void testReadAll() { // add to sgl data
25
       private Utils utils;
                                                                                                         List<Item> items = new ArrayList<>();
26
                                                                                             61
                                                                                                         items.add(new Item(1L, 0.35, "rubber"));
27⊖
        @Mock
                                                                                             62
28
       private ItemDAO dao;
                                                                                                         Mockito.when(dao.readAll()).thenReturn(items);
29
                                                                                             64
30⊖
        @InjectMocks
                                                                                                         assertEquals(items, controller.readAll());
31
       private ItemController controller;
                                                                                             66
32
                                                                                             67
                                                                                                         Mockito.verify(dao, Mockito.times(1)).readAll();
33⊖
       @Test
                                                                                             68
       public void testCreate() {
34
                                                                                             69
35
           //final String ITEM ID = "4", ITEM COST = "0.65", ITEM NAME = "pencil";
                                                                                             70⊖
                                                                                                     @Test
36
           final Long ITEM ID = 1L;
                                                                                             71
                                                                                                    public void testUpdate() {
37
           final Double ITEM COST = 0.65;
                                                                                                         Item updated = new Item(IL, 5.99, "calculator");
           final String ITEM NAME = "pencil";
38
                                                                                             73
           final Item created = new Item(ITEM ID, ITEM COST, ITEM NAME);
39
                                                                                             74
                                                                                                         Mockito.when(this.utils.getLong()).thenReturn(1L);
40
                                                                                             75
                                                                                                         Mockito.when(this.utils.getDouble()).thenReturn(5.99);
41
           //final String F NAME = "barry", L NAME = "scott", U NAME = "scottybl23", P WORI
                                                                                             76
                                                                                                         Mockito.when(this.utils.getString()).thenReturn(updated.getName());
42
           //final Customer created = new Customer(F NAME, L NAME, U NAME, P WORD);
                                                                                             77
                                                                                                         Mockito.when(this.dao.update(updated)).thenReturn(updated);
43
                                                                                             78
44
                                                                                             79
                                                                                                         assertEquals(updated, this.controller.update());
45
           Mockito.when(utils.getLong()).thenReturn(ITEM ID);
                                                                                             80
46
           Mockito.when(utils.getDouble()).thenReturn(ITEM COST);
                                                                                             81
                                                                                                         Mockito.verify(this.utils, Mockito.times(1)).getLong();
47
           Mockito.when(utils.getString()).thenReturn(ITEM NAME);
                                                                                                         Mockito.verify(this.utils, Mockito.times(1)).getDouble();
           //Mockito.when(utils.getString()).thenReturn(ITEM ID, ITEM COST, ITEM NAME);
48
                                                                                             83
                                                                                                         Mockito.verify(this.utils, Mockito.times(1)).getString();
           // Mockito.when(utils.getString()).thenReturn(ITEM NAME);
49
                                                                                             84
                                                                                                         Mockito.verify(this.dao, Mockito.times(1)).update(updated);
           Mockito.when(dao.create(created)).thenReturn(created);
50
                                                                                             85
51
                                                                                             86
52
           assertEquals(created, controller.create());
```

# RISK ASSESSMENT

- Risk Matrix

Risks	Negligible	Minor	Major	Critical	Catastrophic		
Time	e 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	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	rare Corruption No Corruption Github/Gitbash failure Complex vehictory		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 of		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

ELINAMCGLINCHEYI@GMAIL.COM