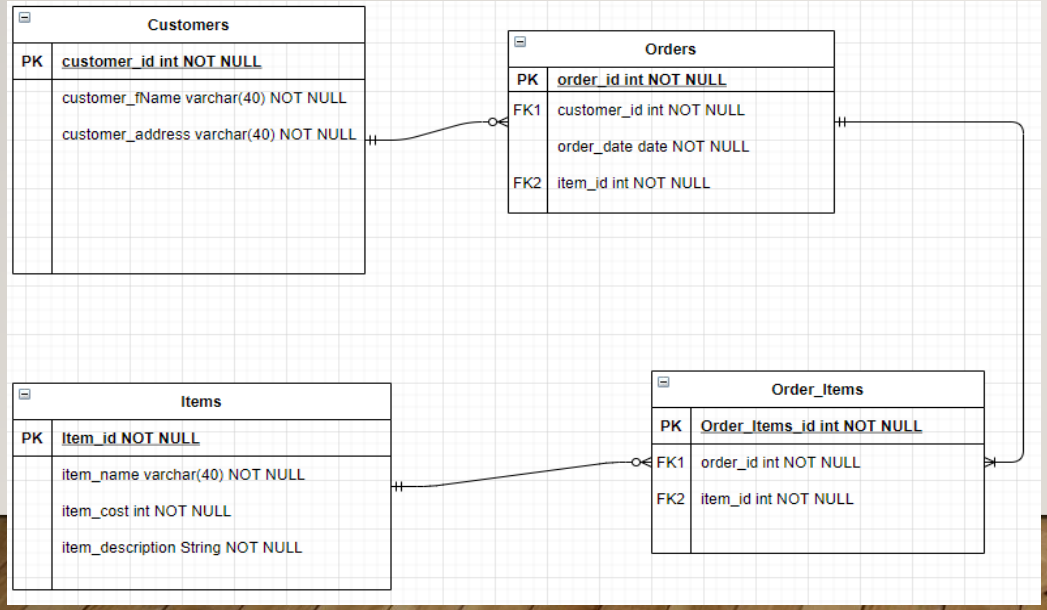
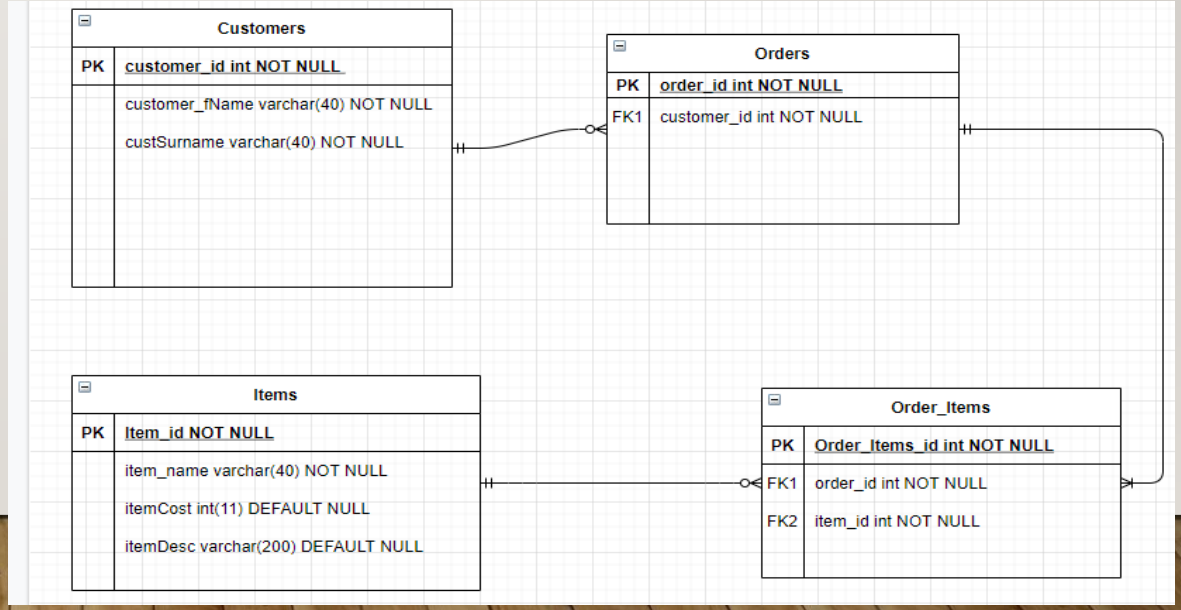
CHARLES HERRIOTT

IMS STARTER PROJECT

INTRODUCTION

- My approach:
  - My first step was to create an ERD diagram to describe what I expected the relationships between the database tables to be.





- I considered the structure in which I would approach the project. This meant including Jira and GitHub in my work approach. I adhered to this flow:

Jira-User Story -> Implement Some Code -> GitHub

- I create a user story which describes the end goal and reason behind it
- I implemented functionality to complete this task
- I then committed the implementation through a feature branch into my GitHub repository

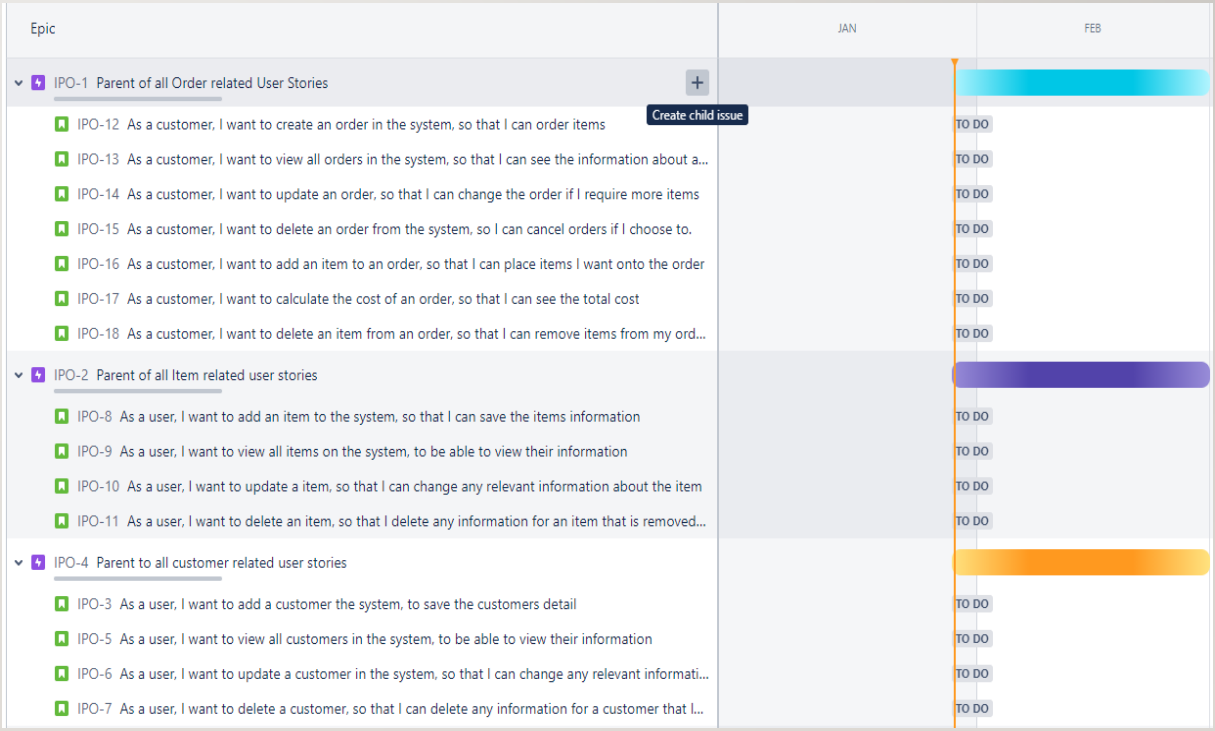
RISK ASSESSMENT

Risk Probability	Risk Severity				
	Catastrophic A	Hazardous B	Major C	Minor D	Negligible E
Highly Likely 5	5A	5B	5C	5D	5E
Probably Likely 4	4A	4B	4C	4D	4E
Likely 3	3A	3B	3C	3D	3E
Less Likely 2	2A	2B	2C	2D	2E
Unlikely 1	1A	1B	1C	1D	1E

Risk	Description	Risk Rating	Actions	Time cost
Hardware Failure	Time could be lost due to a hardware failure. For example, a hard drive being corrupted may lose the current unsaved work.	5C	Consistent backups. Utilising the GitHub repository to maintain versions.	Ranging from <1 hour to a day.

JIRA

- Jira is made up of Epics, User Stories, Tasks and various values/descriptions that can be applied



BREAKING DOWN THE CODE

- My first step was break down the specification into smaller logical steps.
- For example, 'Add a Item to the system'. I took this requirement and broke it down into the steps that I would take to accomplish this task.
- I would need to be able to: create an object in Java, pass that object through an SQL query, add it to the database and then read the row back from the database, including the primary key.

CONSULTANT JOURNEY

- Technologies:
  - Eclipse/ Java
  - Jira
  - GitHub
  - SonarQube
  - Junit
  - Mockito
  - SQL

  - Research

ECLIPSE / JAVA

- Eclipse is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customising the environment.

OBJECT ORIENTED PROGRAMMING PRINCIPLES

- I have attempted to stick to the OOP principles throughout my project, making understandable, clean and efficient code under this structure.

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

SOLID PRINICPLES

- S – single responsibility
- O – open for extension, closed for modification
- L – liskov substitution
- I – interface segregation
- D – dependency inversion

JIRA

- Jira is a proprietary issue tracking product developed by Atlassian that allows bug tracking and agile project management.

JIRA USES

- Jira can be used simply, or in much more detail. For example, a basic user story allows for a brief description of the task.
- Jira story points allow for effort-required values to applied to each task.
- Jira priority can be used to denote the urgency of the task.
- Descriptions can be added to communicate between members of the team, and to highlight issues.
- Smart commits when integrated with GitHub

JIRA USER STORIES

- IPO-4 Parent to all customer related user stories
- IPO-2 Parent of all Item related user stories
- IPO-1 Parent of all Order related User Stories
- IPO-60 Parent of all joined tables related issues
- IPO-62 Issues related to the readability and comments of code
- IPO-92 Parent of all testing user stories

IPO-92 / IPO-79 1

As a developer, I want to test items using Junit, to resolve any issues and check functionality

Done Done

Description

Add a description...

Subtasks

0% Done

IPO-98 Create test methods for each ... TO DO

IPO-99 Test various outcomes to cover... TO DO

What needs to be done?

Create Cancel

Labels	None
Story Points	5
Epic Link	Testing
Sprint	None +1
Priority	Highest

GITHUB

- GitHub is a distributed version-control system for tracking changes in any set of files.

GITHUB USES

- GitHub allows for controlled commits of features through GitBash.
- This allows for projects to be worked on in tandem with other developers.
- New code can be committed and then merge request can be pulled and reviewed before actually being implemented.
- The power of version control really shines when a version rollback is needed, and the previous can be simply pulled from GitHub.

GITHUB COMMIT PROGRESSION

First Commit

This was the first commit that I made. Committing some implementation.

Commit and then create new branch

w
This commit is from around the middle of the timeline, where I was completing user story tasks and then merging the branches.

GITHUB SMART COMMITS

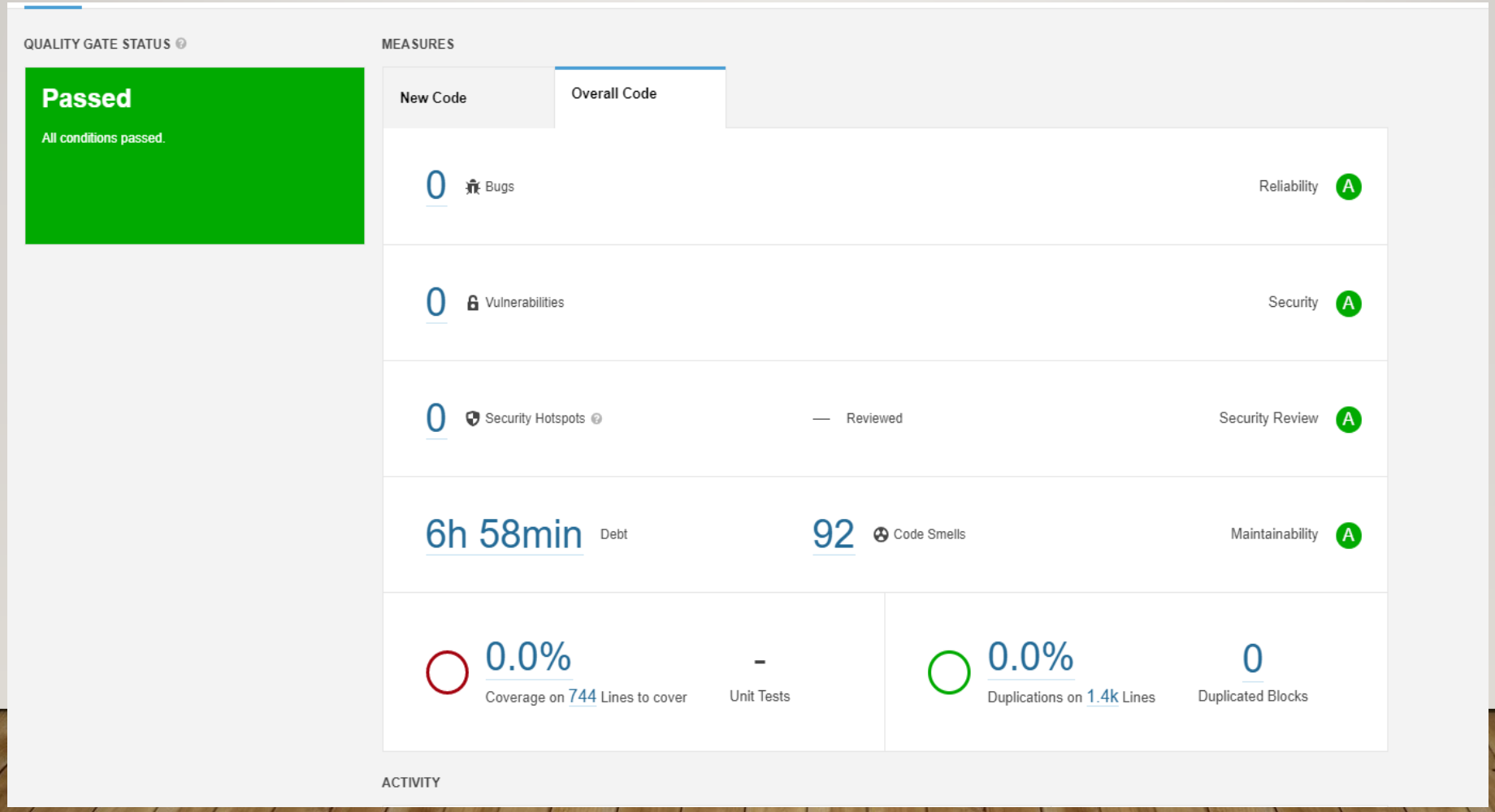
Assignee	Unassigned	
Reporter	CH Charles Herriott	
Development	2 commits	2 minutes ago
Labels	None	
Story Points	3	

```
Charl@COMPUTE-VA97HIP MINGW64 /d/IMS-Project-Test/IMS-Starter (master)
$ git add .

Charl@COMPUTE-VA97HIP MINGW64 /d/IMS-Project-Test/IMS-Starter (master)
$ git commit -m "IPO-76 #comment commiting presentation version"
[master 9bf8268] IPO-76 #comment commiting presentation version
 5 files changed, 12 insertions(+), 10 deletions(-)
 create mode 100644 Project-Documentation/~$rkflow_documentation.docx

Charl@COMPUTE-VA97HIP MINGW64 /d/IMS-Project-Test/IMS-Starter (master)
$ git push origin master
Enumerating objects: 41, done.
Counting objects: 100% (41/41), done.
Delta compression using up to 8 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (22/22), 309.45 KiB | 11.05 MiB/s, done.
Total 22 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 7 local objects.
To https://github.com/CharlesHerriott/IMS-Starter.git
   4b92505..9bf8268  master -> master

Charl@COMPUTE-VA97HIP MINGW64 /d/IMS-Project-Test/IMS-Starter (master)
$
```

IMS-Starter (GitHub) Show all files

Author	Commit	Message	Date	Files
	9bf826	IPO-76 #comment commiting presentation version	2 minutes ago	5 files
	6328e4	IPO-76 #comment testing Jira smart commits	3 days ago	2 files

SONARQUBE


QUALITY GATE STATUS
Passed
All conditions passed.
MEASURES
New Code
Overall Code
0 Bugs
Reliability A
0 Vulnerabilities
Security A
0 Security Hotspots
— Reviewed
Security Review A
6h 58min Debt
92 Code Smells
Maintainability A
0.0%
Coverage on 744 Lines to cover
-
Unit Tests
0.0%
Duplications on 1.4k Lines
0
Duplicated Blocks
ACTIVITY

SONARQUBE

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

ims master

Last analysis had 1 warning February 9, 2021, 4:17 PM Version 0.0.1

Overview Issues Security Hotspots Measures Code Activity

Project Settings Project Information

5 / 92 issues

...a/com/qa/ims/controller/ItemController.java

Immediately return this expression instead of assigning it to the temporary variable "item".
Code Smell

.../com/qa/ims/controller/OrderController.java

This block of commented-out lines of code should be removed.
Code Smell +16

Invoke method(s) only conditionally.
Code Smell

Invoke method(s) only conditionally.
Code Smell

Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
Code Smell

Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
Code Smell

Remove this useless assignment to local variable "item_id".
Code Smell

Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
Code Smell

Remove this useless assignment to local variable "quantity"

Invoke method(s) only conditionally. Why is this an issue? yesterday L49
Code Smell Major Open Not assigned 5min effort Comment performance

```
50 cherr…        }
51 cherr…        LOGGER.info("Would you like to read ALL orders or by an ID?");
52               String choice = utils.getString();
53
54               if (choice.equalsIgnoreCase("ALL")) {
55 cherr…            List<Order> orderItems = orderDAO.readAllItems(true);
56 cherr…            for (Order orderItem : orderItems) {
57                       LOGGER.info(orderItem.itemsToString());
```

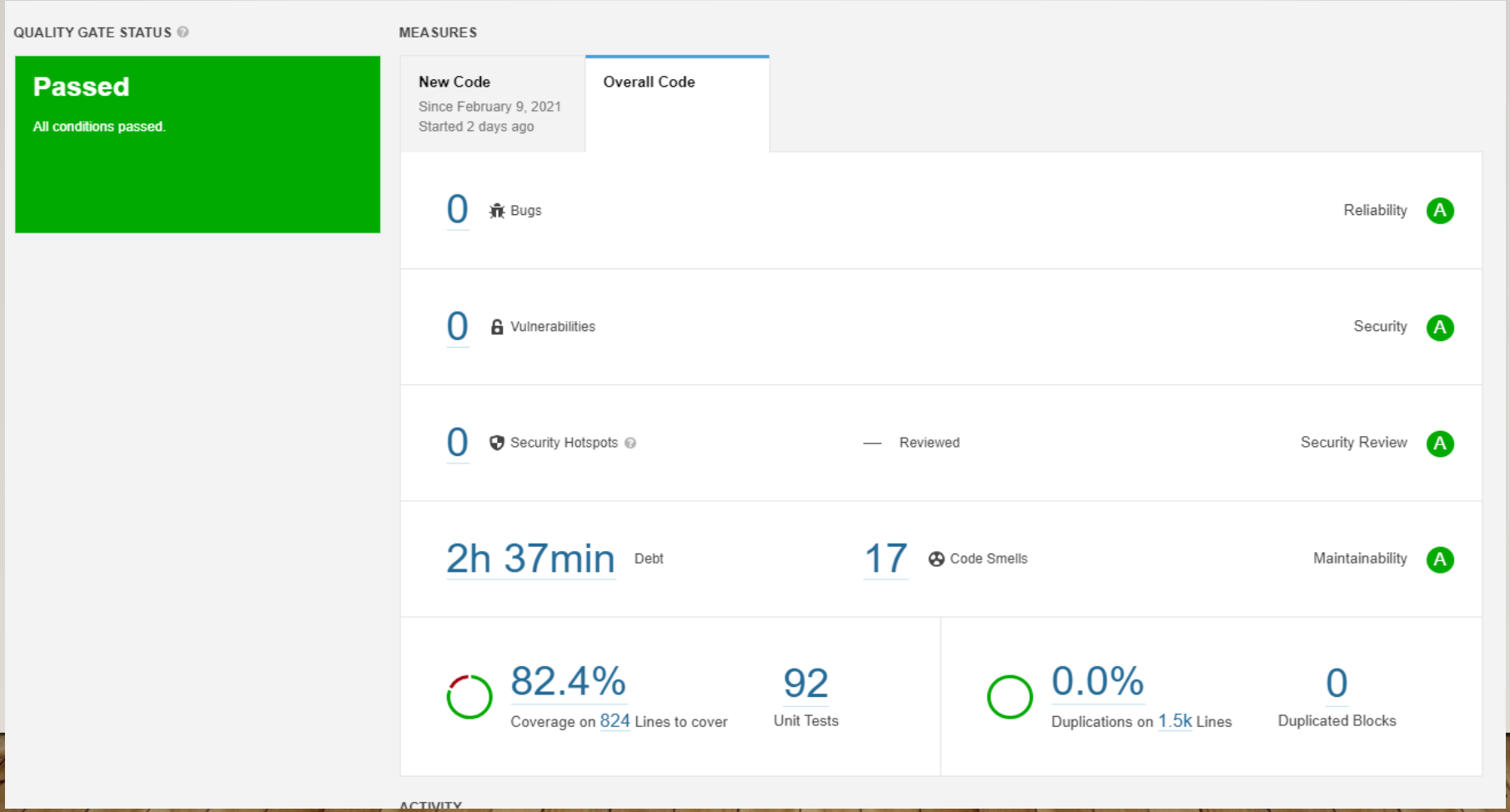
Invoke method(s) only conditionally. Why is this an issue? yesterday L57
Code Smell Major Open Not assigned 5min effort Comment performance

```
58                   }
59                   return orderItems;
60               } else if (choice.equalsIgnoreCase("ID")) {
61                   readSpecific();
62               }
63 cherr…        return orders;
64           }
65
66           @Override
67           public Order create() {
68               boolean complete = false;
69               String choice;
70               LOGGER.info("Enter your customer id");
71               Long cust_id = utils.getLong();
```

Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'. Why is this an issue? 5 days ago L71
Code Smell Minor Open Not assigned 2min effort Comment convention

```
72 cherr…        Order order = orderDAO.createUpdateOrder(new Order(cust_id),true);
73 cherr…        do {
74 cherr…            LOGGER.info("Would you like to order another item? Yes (Y)/No (N)");
75 cherr…            choice = utils.getString();
76                   if (choice.equalsIgnoreCase("Y")) {
77 cherr…                createOrderItem(order.getOrder_id());
78 cherr…            } else {
79                       complete = true;
```

SONARQUBE


QUALITY GATE STATUS
Passed
All conditions passed.
MEASURES
New Code
Since February 9, 2021
Started 2 days ago
Overall Code
0 Bugs
Reliability A
0 Vulnerabilities
Security A
0 Security Hotspots
— Reviewed
Security Review A
2h 37min Debt
17 Code Smells
Maintainability A
82.4%
Coverage on 824 Lines to cover
92
Unit Tests
0.0%
Duplications on 1.5k Lines
0
Duplicated Blocks
ACTIVITY

JUNIT

```java
public Item(Long id, String name, Double cost, String description) {
    this.setId(id);
    this.setName(name);
    this.setCost(cost);
    this.setDescription(description);
}

public Item(String name, Double cost, String description) {
    this.setName(name);
    this.setCost(cost);
    this.setDescription(description);
}

public long getId() {
    return id;
}

public String getName() {
    return name;
}

public Double getCost() {
    return cost;
}

public String getDescription() {
    return description;
}

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

public void setName(String name) {
    this.name = name;
}

public void setCost(Double cost) {
```

```java
public class ItemTest {

    Item i = new Item(12345L, "toothbrush", 15.0, "Teeth scrubber");

    @Test
    public void testEquals() {
        EqualsVerifier.simple().forClass(Item.class).verify();
    }

    @Test
    public void testConstructor() {
        Item iTest = new Item("Charlie", 18.5, "Tall man");
        Double testCost = iTest.getCost();
        assertEquals(18.5, testCost,1);
        assertEquals("Tall man", iTest.getDescription());
    }

    @Test
    public void testGetId() {
        i.setId(123L);
        long test = i.getId();
        assertEquals(123L, test);
    }

    @Test
    public void testGetName() {
        i.setName("Mouse");
        assertEquals("Mouse", i.getName());
    }


    @Test
    public void testToString(){
        assertEquals("id: 12345 Name: toothbrush Cost: 15.00 Description: Teeth scrubb
    }

}
```

JUNIT TEST CASE

```
@Test
public void testCreate() {
    final Item created = new Item(2L, "Mouse", 8.0, "Clicker");
    assertEquals(created, itemDAO.create(created));
}
```

```
public Item create(Item item) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection
                    .prepareStatement(Queries.CREATEITEM.getDescription())
        statement.setString(1, item.getName());
        statement.setDouble(2, item.getCost());
        statement.setString(3, item.getDescription());
        statement.executeUpdate();
        return readLatest();
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
}
```

MOCKITO

```java
    }

    @Override
    public Item create() {
        LOGGER.info("Please enter a name");
        String name = utils.getString();
        LOGGER.info("Please enter a cost");
        Double cost = utils.getDouble();
        LOGGER.info("Please enter a description");
        String description = utils.getString();
        Item item = itemDAO.create(new Item(name,cost,description));
        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 a name");
        String name = utils.getString();
        LOGGER.info("Please enter a cost");
        Double cost = utils.getDouble();
        LOGGER.info("Please enter a description");
        String description = utils.getString();
        Item item = itemDAO.update(new Item(id, name, cost, description));
        LOGGER.info("Item updated");
        return item;
    }

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

```java
        assertEquals(items, controller.readAll());

        Mockito.verify(itemDAO, Mockito.times(1)).readAll();
    }


    @Test
    public void updateTest() {
        Item updated = new Item(1L, "chris",8.0, "perrins");

        Mockito.when(this.utils.getLong()).thenReturn(1L);
        Mockito.when(this.utils.getString()).thenReturn(updated.getName()
        Mockito.when(this.utils.getDouble()).thenReturn(updated.getCost()
        Mockito.when(this.itemDAO.update(updated)).thenReturn(updated);

        assertEquals(updated, this.controller.update());

        Mockito.verify(this.utils, Mockito.times(1)).getLong();
        Mockito.verify(this.utils, Mockito.times(2)).getString();
        Mockito.verify(this.utils, Mockito.times(1)).getDouble();
        Mockito.verify(this.itemDAO, Mockito.times(1)).update(updated);

    }

    @Test
    public void deleteTest() {
        final long ID = 1L;

        Mockito.when(utils.getLong()).thenReturn(ID);
        Mockito.when(itemDAO.delete(ID)).thenReturn(1);

        assertEquals(1L, this.controller.delete());

        Mockito.verify(utils, Mockito.times(1)).getLong();
```

MOCKITO TEST CASE

```java
@Override
public Item create() {
    LOGGER.info("Please enter a name");
    String name = utils.getString();
    LOGGER.info("Please enter a cost");
    Double cost = utils.getDouble();
    LOGGER.info("Please enter a description");
    String description = utils.getString();
    Item item = itemDAO.create(new Item(name,cost,description));
    LOGGER.info("Item created");
    return item;
}
```

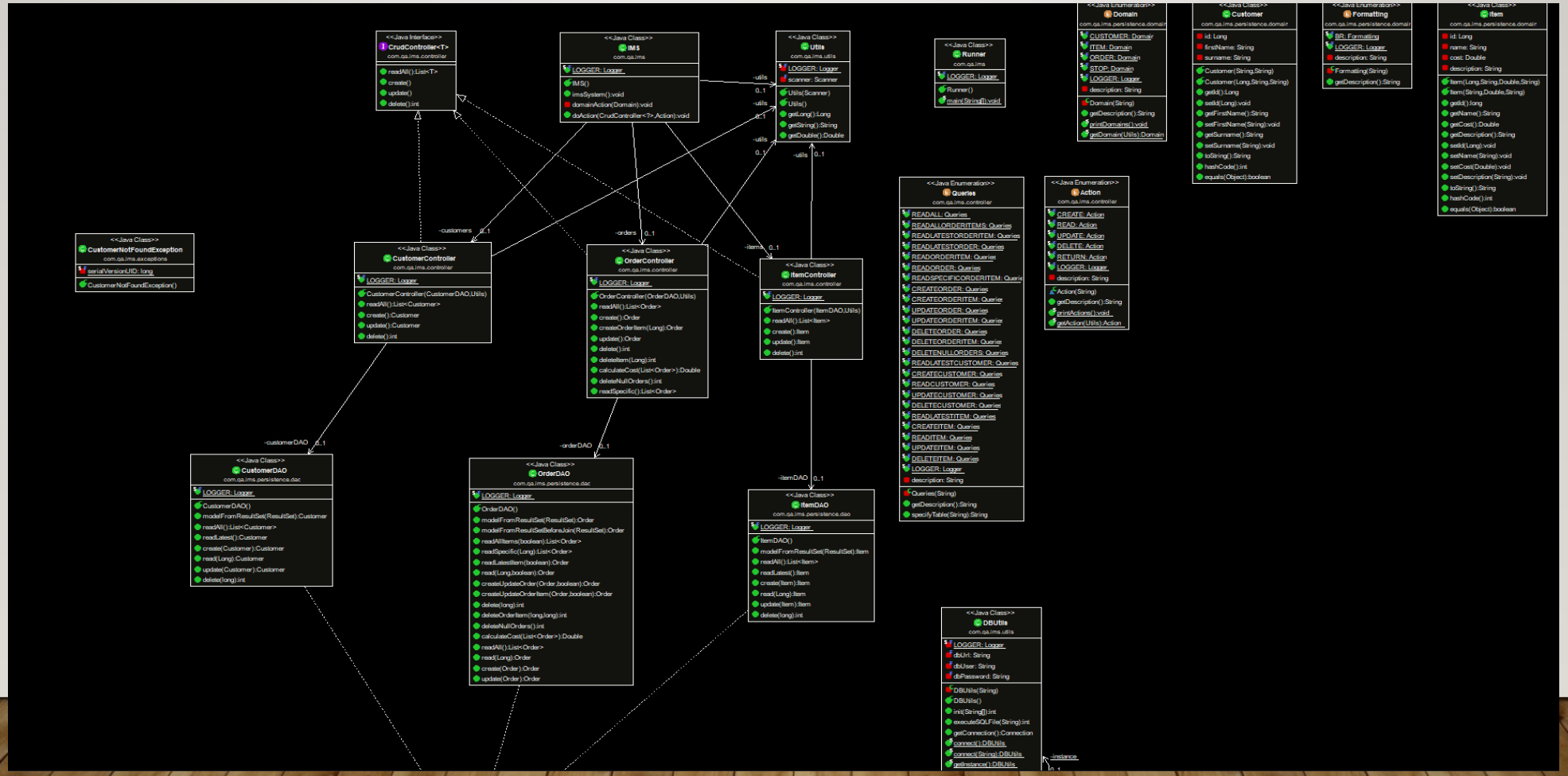
```java
@Test
public void createTest() {
    // RESOURCES
    final String itemName = "Mouse", itemDesc = "Clicker";
    final Double itemCost = 8.0;
    final Item created = new Item(itemName, itemCost, itemDesc);

    // RULES
    Mockito.when(utils.getString()).thenReturn(itemName, itemDesc);
    Mockito.when(utils.getDouble()).thenReturn(itemCost);
    Mockito.when(itemDAO.create(created)).thenReturn(created);

    // ACTIONS
    final Item result = controller.create();

    // ASSERTIONS
    assertEquals(created, result);
    Mockito.verify(utils, Mockito.times(2)).getString();
    Mockito.verify(utils, Mockito.times(1)).getDouble();
    Mockito.verify(itemDAO, Mockito.times(1)).create(created);
}
```

DEMONSTRATION

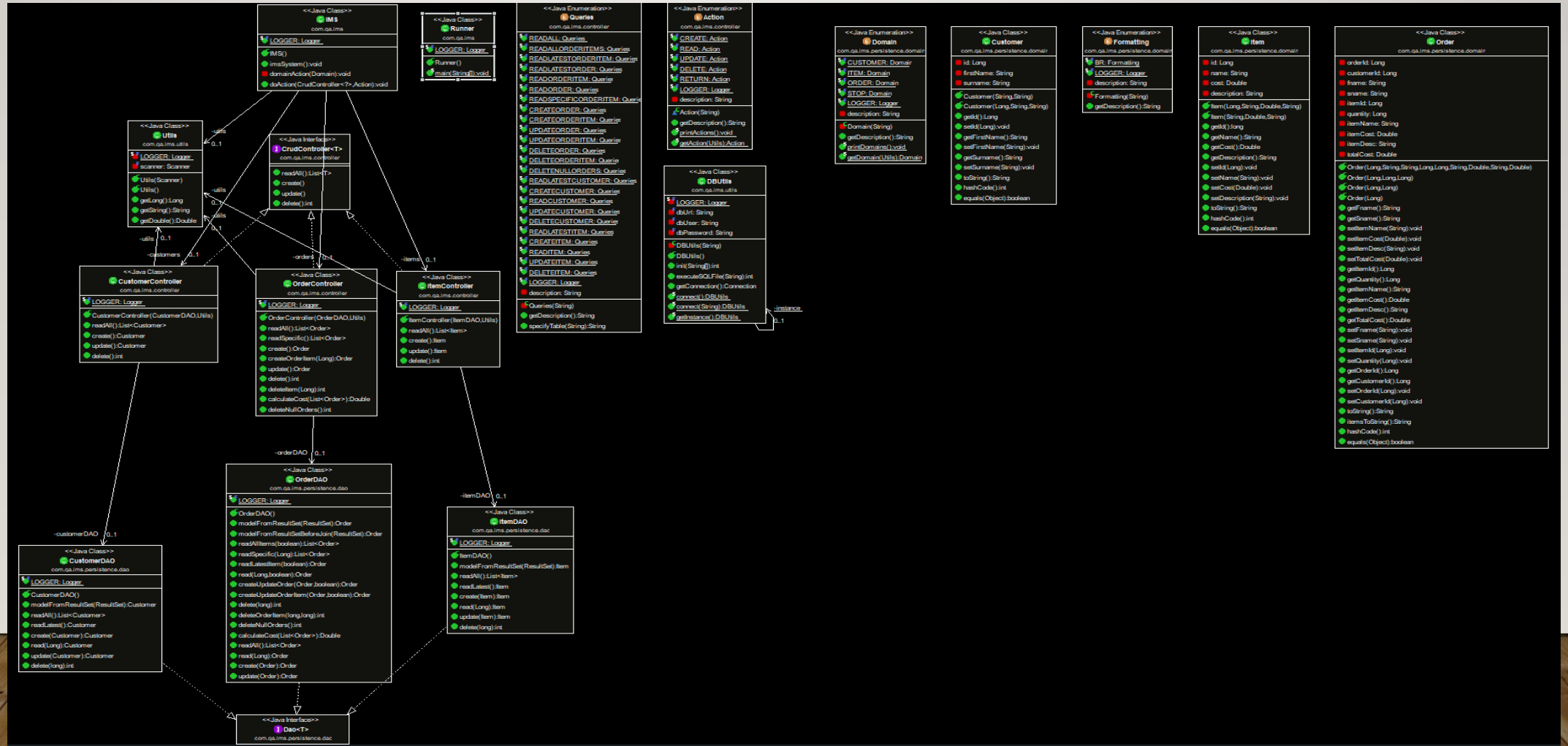
- A variety of user stories will now be demonstrated through the GitBash console.

~~IPO-13~~	As a customer, I want to view all orders in the system, so that I can see the information abo...
~~IPO-15~~	As a customer, I want to delete an order from the system, so I can cancel orders if I choose...
~~IPO-14~~	As a customer, I want to update an order, so that I can change the order if I require more it...
~~IPO-16~~	As a customer, I want to add an item to an order, so that I can place items I want onto the ...
~~IPO-18~~	As a customer, I want to delete an item from an order, so that I can remove items from my ...
~~IPO-17~~	As a customer, I want to calculate the cost of an order, so that I can see the total cost
~~IPO-12~~	As a customer, I want to create an order in the system, so that I can order items

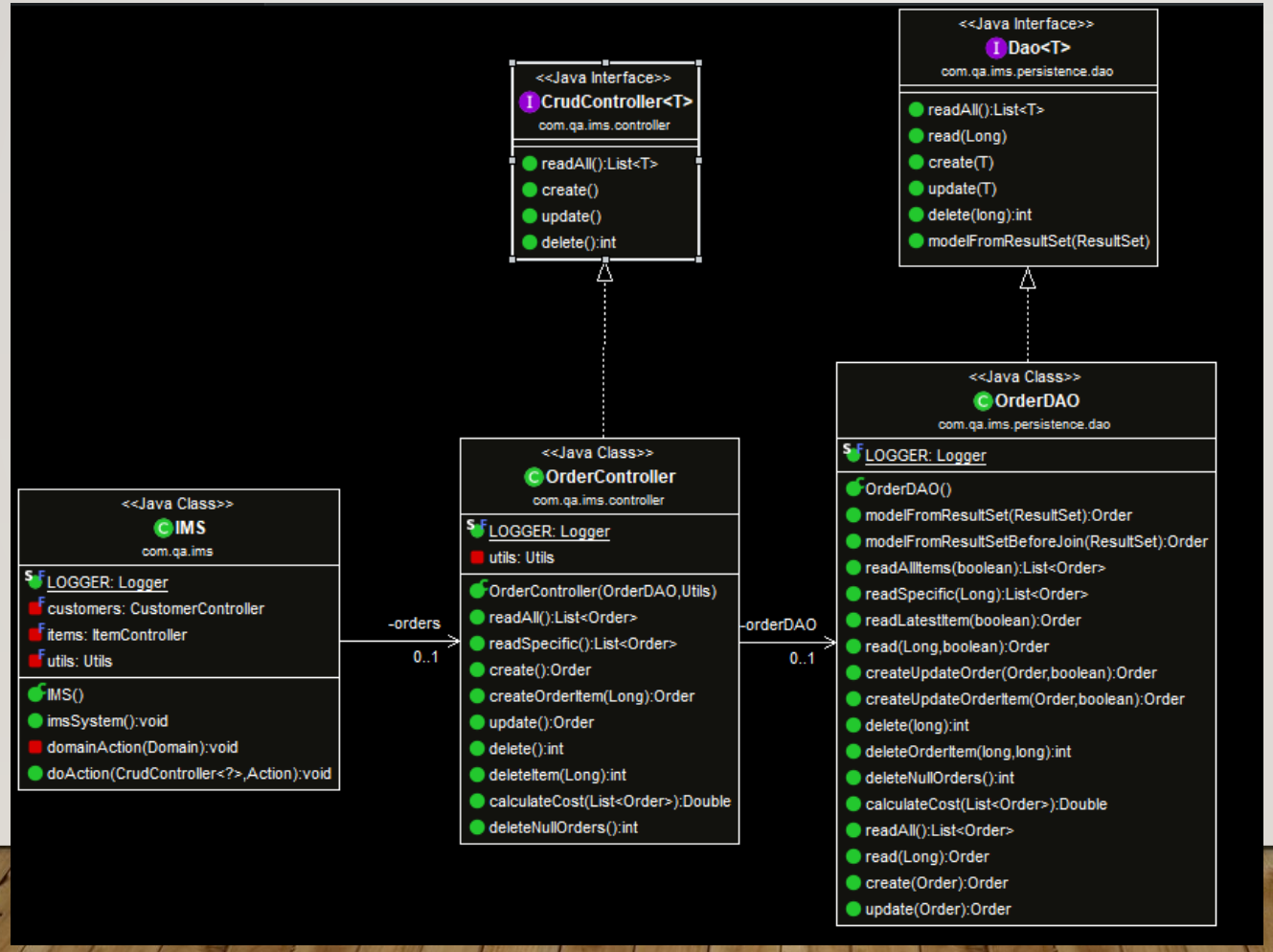
UML INITIAL


CrudController<T>
com.qa.ims.controller
readAll():List<T>
create()
update()
delete():int
IMS
com.qa.ims
LOGGER: Logger
IMS()
imsSystem():void
domainAction(Domain):void
doAction(CrudController<?>,Action):void
Utils
com.qa.ims.utils
LOGGER: Logger
scanner: Scanner
Utils(Scanner)
Utils()
getLong():Long
getString():String
getDouble():Double
Runner
com.qa.ims
LOGGER: Logger
Runner()
main(String[]):void
Domain
com.qa.ims.persistence.domain
CUSTOMER: Domain
ITEM: Domain
ORDER: Domain
STOP: Domain
LOGGER: Logger
description: String
Domain(String)
getDescription():String
printDomains():void
getDomain(Utils):Domain
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):void
getSurname():String
setSurname(String):void
toString():String
hashCode():int
equals(Object):boolean
Formatting
com.qa.ims.persistence.domain
BR: Formatting
LOGGER: Logger
description: String
Formatting(String)
getDescription():String
Item
com.qa.ims.persistence.domain
id: Long
name: String
cost: Double
description: String
Item(Long,String,Double,String)
Item(String,Double,String)
getId():long
getName():String
getCost():Double
getDescription():String
setId(Long):void
setName(String):void
setCost(Double):void
setDescription(String):void
toString():String
hashCode():int
equals(Object):boolean
CustomerNotFoundException
com.qa.ims.exceptions
serialVersionUID: long
CustomerNotFoundException()
CustomerController
com.qa.ims.controller
LOGGER: Logger
CustomerController(CustomerDAO,Utils)
readAll():List<Customer>
create():Customer
update():Customer
delete():int
OrderController
com.qa.ims.controller
LOGGER: Logger
OrderController(OrderDAO,Utils)
readAll():List<Order>
create():Order
createOrderItem(Long):Order
update():Order
delete():int
deleteItem(Long):int
calculateCost(List<Order>):Double
deleteNullOrders():int
readSpecific():List<Order>
ItemController
com.qa.ims.controller
LOGGER: Logger
ItemController(ItemDAO,Utils)
readAll():List<Item>
create():Item
update():Item
delete():int
Queries
com.qa.ims.controller
READALL: Queries
READALLORDERITEMS: Queries
READLATESTORDERITEM: Queries
READLATESTORDER: Queries
READORDERITEM: Queries
READORDER: Queries
READSPECIFICORDERITEM: Queries
CREATEORDER: Queries
CREATEORDERITEM: Queries
UPDATEORDER: Queries
UPDATEORDERITEM: Queries
DELETEORDER: Queries
DELETEORDERITEM: Queries
DELETENULLORDERS: Queries
READLATESTCUSTOMER: Queries
CREATECUSTOMER: Queries
READCUSTOMER: Queries
UPDATECUSTOMER: Queries
DELETECUSTOMER: Queries
READLATESTITEM: Queries
CREATEITEM: Queries
READITEM: Queries
UPDATEITEM: Queries
DELETEITEM: Queries
LOGGER: Logger
description: String
Queries(String)
getDescription():String
specifyTable(String):String
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
CustomerDAO
com.qa.ims.persistence.dao
LOGGER: Logger
CustomerDAO()
modelFromResultSet(ResultSet):Customer
readAll():List<Customer>
readLatest():Customer
create(Customer):Customer
read(Long):Customer
update(Customer):Customer
delete(long):int
OrderDAO
com.qa.ims.persistence.dao
LOGGER: Logger
OrderDAO()
modelFromResultSet(ResultSet):Order
modelFromResultSetBeforeJoin(ResultSet):Order
readAllItems(boolean):List<Order>
readSpecific(Long):List<Order>
readLatestItem(boolean):Order
read(Long,boolean):Order
createUpdateOrder(Order,boolean):Order
createUpdateOrderItem(Order,boolean):Order
delete(long):int
deleteOrderItem(long,long):int
deleteNullOrders():int
calculateCost(List<Order>):Double
readAll():List<Order>
read(Long):Order
create(Order):Order
update(Order):Order
ItemDAO
com.qa.ims.persistence.dao
LOGGER: Logger
ItemDAO()
modelFromResultSet(ResultSet):Item
readAll():List<Item>
readLatest():Item
create(Item):Item
read(Long):Item
update(Item):Item
delete(long):int
DBUtils
com.qa.ims.utils
LOGGER: Logger
dbUrl: String
dbUser: String
dbPassword: String
DBUtils(String)
DBUtils()
init(String[]):int
executeSQLFile(String):int
getConnection():Connection
connect():DBUtils
connect(String):DBUtils
getInstance():DBUtils
-customers
-orders
-items
-utils
-customerDAO
-orderDAO
-itemDAO
0..1

UML REVISED


IMS
com.qa.ims
LOGGER: Logger
IMS()
imsSystem():void
domainAction(Domain):void
doAction(CrudController<?>,Action):void
Runner
com.qa.ims
LOGGER: Logger
Runner()
main(String[]):void
Queries
com.qa.ims.controller
READALL: Queries
READALLORDERITEMS: Queries
READLATESTORDERITEM: Queries
READLATESTORDER: Queries
READORDERITEM: Queries
READORDER: Queries
READSPECIFICORDERITEM: Queries
CREATEORDER: Queries
CREATEORDERITEM: Queries
UPDATEORDER: Queries
UPDATEORDERITEM: Queries
DELETEORDER: Queries
DELETEORDERITEM: Queries
DELETENULLORDERS: Queries
READLATESTCUSTOMER: Queries
CREATECUSTOMER: Queries
READCUSTOMER: Queries
UPDATECUSTOMER: Queries
DELETECUSTOMER: Queries
READLATESTITEM: Queries
CREATEITEM: Queries
READITEM: Queries
UPDATEITEM: Queries
DELETEITEM: Queries
LOGGER: Logger
description: String
Queries(String)
getDescription():String
specifyTable(String):String
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
Domain
com.qa.ims.persistence.domain
CUSTOMER: Domain
ITEM: Domain
ORDER: Domain
STOP: Domain
LOGGER: Logger
description: String
Domain(String)
getDescription():String
printDomains():void
getDomain(Utils):Domain
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):void
getSurname():String
setSurname(String):void
toString():String
hashCode():int
equals(Object):boolean
Formatting
com.qa.ims.persistence.domain
BR: Formatting
LOGGER: Logger
description: String
Formatting(String)
getDescription():String
Item
com.qa.ims.persistence.domain
id: Long
name: String
cost: Double
description: String
Item(Long,String,Double,String)
Item(String,Double,String)
getId():long
getName():String
getCost():Double
getDescription():String
setId(Long):void
setName(String):void
setCost(Double):void
setDescription(String):void
toString():Stirng
hashCode():int
equals(Object):boolean
Order
com.qa.ims.persistence.domain
orderId: Long
customerId: Long
fname: String
sname: String
itemId: Long
quantity: Long
itemName: String
itemCost: Double
itemDesc: String
totalCost: Double
Order(Long,String,String,Long,Long,String,Double,String,Double)
Order(Long,Long,Long)
Order(Long,Long)
Order(Long)
getFname():String
getSname():String
setItemName(String):void
setItemCost(Double):void
setItemDesc(String):void
setTotalCost(Double):void
getItemId():Long
getQuantity():Long
getItemName():String
getItemCost():Double
getItemDesc():String
getTotalCost():Double
setFname(String):void
setSname(String):void
setItemId(Long):void
setQuantity(Long):void
getOrderId():Long
getCustomerId():Long
setOrderId(Long):void
setCustomerId(Long):void
toString():String
itemsToString():String
hashCode():int
equals(Object):boolean
Utils
com.qa.ims.utils
LOGGER: Logger
scanner: Scanner
Utils(Scanner)
Utils()
getLong():Long
getString():String
getDouble():Double
CrudController<T>
com.qa.ims.controller
readAll():List<T>
create()
update()
delete():int
DBUtils
com.qa.ims.utils
LOGGER: Logger
dbUrl: String
dbUser: String
dbPassword: String
DBUtils(String)
DBUtils()
init(String[]):int
executeSQLFile(String):int
getConnection():Connection
connect():DBUtils
connect(String):DBUtils
getInstance():DBUtils
-instance
CustomerController
com.qa.ims.controller
LOGGER: Logger
CustomerController(CustomerDAO,Utils)
readAll():List<Customer>
create():Customer
update():Customer
delete():int
OrderController
com.qa.ims.controller
LOGGER: Logger
OrderController(OrderDAO,Utils)
readAll():List<Order>
readSpecific():List<Order>
create():Order
createOrderItem(Long):Order
update():Order
delete():int
deleteItem(Long):int
calculateCost(List<Order>):Double
deleteNullOrders():int
ItemController
com.qa.ims.controller
LOGGER: Logger
ItemController(ItemDAO,Utils)
readAll():List<Item>
create():Item
update():Item
delete():int
OrderDAO
com.qa.ims.persistence.dao
LOGGER: Logger
OrderDAO()
modelFromResultSet(ResultSet):Order
modelFromResultSetBeforeJoin(ResultSet):Order
readAllItems(boolean):List<Order>
readSpecific(Long):List<Order>
readLatestItem(boolean):Order
read(Long,boolean):Order
createUpdateOrder(Order,boolean):Order
createUpdateOrderItem(Order,boolean):Order
delete(long):int
deleteOrderItem(long,long):int
deleteNullOrders():int
calculateCost(List<Order>):Double
readAll():List<Order>
read(Long):Order
create(Order):Order
update(Order):Order
CustomerDAO
com.qa.ims.persistence.dao
LOGGER: Logger
CustomerDAO()
modelFromResultSet(ResultSet):Customer
readAll():List<Customer>
readLatest():Customer
create(Customer):Customer
read(Long):Customer
update(Customer):Customer
delete(long):int
ItemDAO
com.qa.ims.persistence.dao
LOGGER: Logger
ItemDAO()
modelFromResultSet(ResultSet):Item
readAll():List<Item>
readLatest():Item
create(Item):Item
read(Long):Item
update(Item):Item
delete(long):int
Dao<T>
com.qa.ims.persistence.dao
-utils
-customers
-orders
-items
-orderDAO
-itemDAO
-customerDAO
0..1

UML TEST CASE



SPRINT REVIEW

Completed Epics:

- Customers
- Items
- Orders
- Order-Items
- Testing

To be completed:

- - Code Admin

As a developer, I want to revise the format of the output to make it more readable, so the user can navigate it more easily

As a developer, I want to output understandable error messages when attempting to access a row that is not present within a table, for a better user experience

As a developer, I want to add more functionality throughout the program to increase efficiency, for a better user experience and a more optimised program

SPRINT RETROSPECTIVE

What I did well:

- Jira – well documented and thought out
- GitHub – made consistent commits (smart commits inc.)
- Code – efficient, well ordered, good naming conventions
- Project Documentation – well detailed, covered all aspects.

What I could have done better:

- Jira – better thought out sprints
- GitHub – Smart commits from the start.
- Code – Better formatted output. More readable error messages (especially SQL errors)

CONCLUSION

- I feel that I completed the project to a good standard
- There are future implementations that could be added, which would increase the efficiency of the application and the user experience.
- I would like to further my knowledge and abilities to use Jira and GitHub properly.

THANKS FOR WATCHING