IMS Fundamental Project

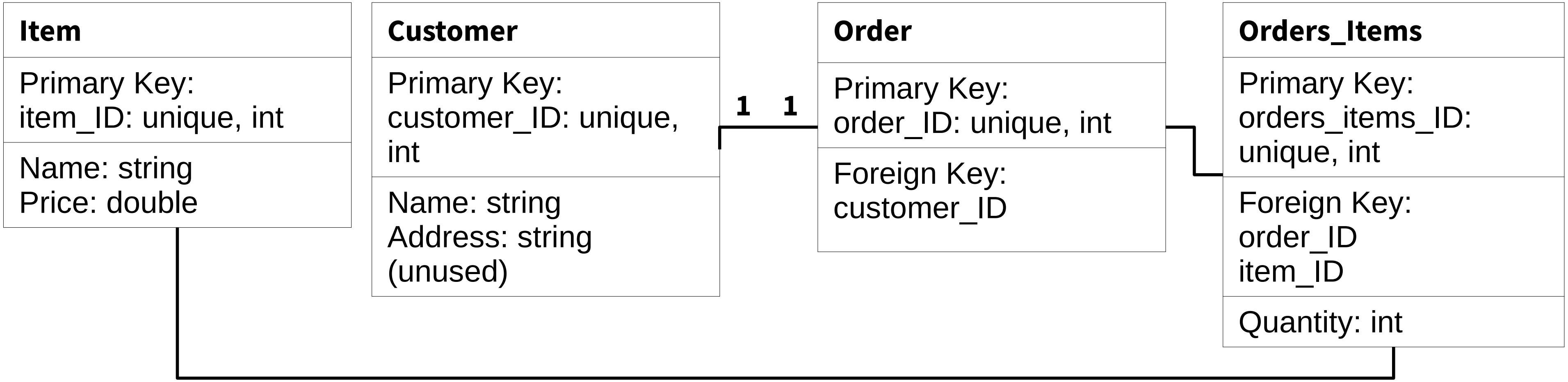
An Inventory Management System

Technologies Used

- MySQL for creating database schema
- Java for connecting to database and managing entries
- Git for version control to comply with feature-branch model
- JUnit and Mockito for testing
- Jira Kanban board for planning

Database Schema

Original Plan



Item

Primary Key:
item_ID: unique, int

Name: string
Price: double

Customer

Primary Key:
customer_ID: unique, int

Name: string
Address: string (unused)

1 1

Order

Primary Key:
order_ID: unique, int

Foreign Key:
customer_ID

Orders_Items

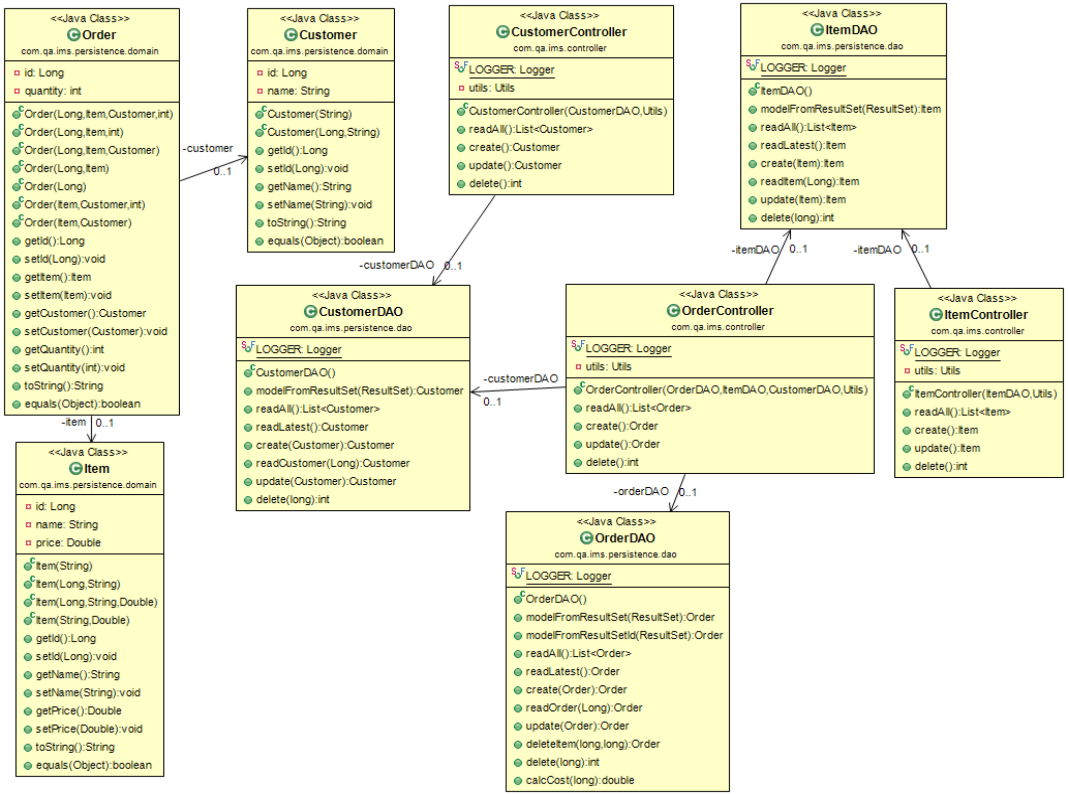
Primary Key:
orders_items_ID: unique, int

Foreign Key:
order_ID
item_ID

Quantity: int

IDs changed to long, address was not added to Customer table

UML Class Diagram



<<Java Class>>
Order
com.qa.ims.persistence.domain

- id: Long
- quantity: int

- Order(Long,Item,Customer,int)
- Order(Long,Item,int)
- Order(Long,Item,Customer)
- Order(Long,Item)
- Order(Long)
- Order(Item,Customer,int)
- Order(Item,Customer)
- getId():Long
- setId(Long):void
- getItem():Item
- setItem(Item):void
- getCustomer():Customer
- setCustomer(Customer):void
- getQuantity():int
- setQuantity(int):void
- toString():String
- equals(Object):boolean

<<Java Class>>
Customer
com.qa.ims.persistence.domain

- id: Long
- name: String

- Customer(String)
- Customer(Long,String)
- getId():Long
- setId(Long):void
- getName():String
- setName(String):void
- toString():String
- equals(Object):boolean

<<Java Class>>
CustomerController
com.qa.ims.controller

- LOGGER: Logger
- utils: Utils

- CustomerController(CustomerDAO,Utils)
- readAll():List<Customer>
- create():Customer
- update():Customer
- delete():int

<<Java Class>>
ItemDAO
com.qa.ims.persistence.dao

- LOGGER: Logger

- ItemDAO()
- modelFromResultSet(ResultSet):Item
- readAll():List<Item>
- readLatest():Item
- create(Item):Item
- readItem(Long):Item
- update(Item):Item
- delete(long):int

<<Java Class>>
CustomerDAO
com.qa.ims.persistence.dao

- LOGGER: Logger

- CustomerDAO()
- modelFromResultSet(ResultSet):Customer
- readAll():List<Customer>
- readLatest():Customer
- create(Customer):Customer
- readCustomer(Long):Customer
- update(Customer):Customer
- delete(long):int

<<Java Class>>
OrderController
com.qa.ims.controller

- LOGGER: Logger
- utils: Utils

- OrderController(OrderDAO,ItemDAO,CustomerDAO,Utils)
- readAll():List<Order>
- create():Order
- update():Order
- delete():int

<<Java Class>>
ItemController
com.qa.ims.controller

- LOGGER: Logger
- utils: Utils

- ItemController(ItemDAO,Utils)
- readAll():List<Item>
- create():Item
- update():Item
- delete():int

<<Java Class>>
Item
com.qa.ims.persistence.domain

- id: Long
- name: String
- price: Double

- Item(String)
- Item(Long,String)
- Item(Long,String,Double)
- Item(String,Double)
- getId():Long
- setId(Long):void
- getName():String
- setName(String):void
- getPrice():Double
- setPrice(Double):void
- toString():String
- equals(Object):boolean

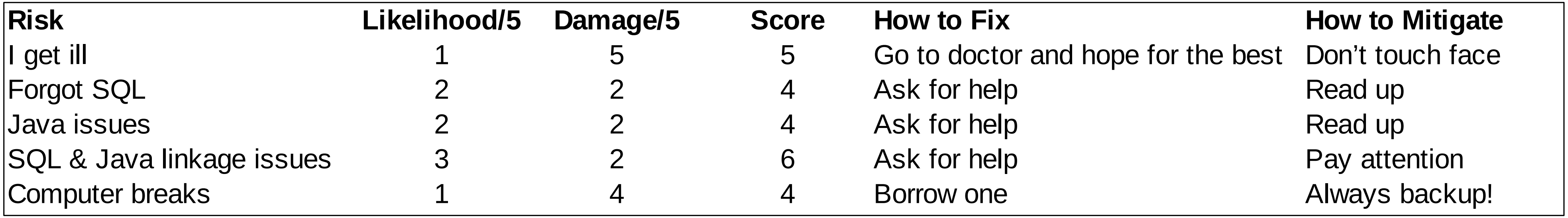
<<Java Class>>
OrderDAO
com.qa.ims.persistence.dao

- LOGGER: Logger

- OrderDAO()
- modelFromResultSet(ResultSet):Order
- modelFromResultSetId(ResultSet):Order
- readAll():List<Order>
- readLatest():Order
- create(Order):Order
- readOrder(Long):Order
- update(Order):Order
- deleteItem(long,long):Order
- delete(long):int
- calcCost(long):double

Risk Assesment

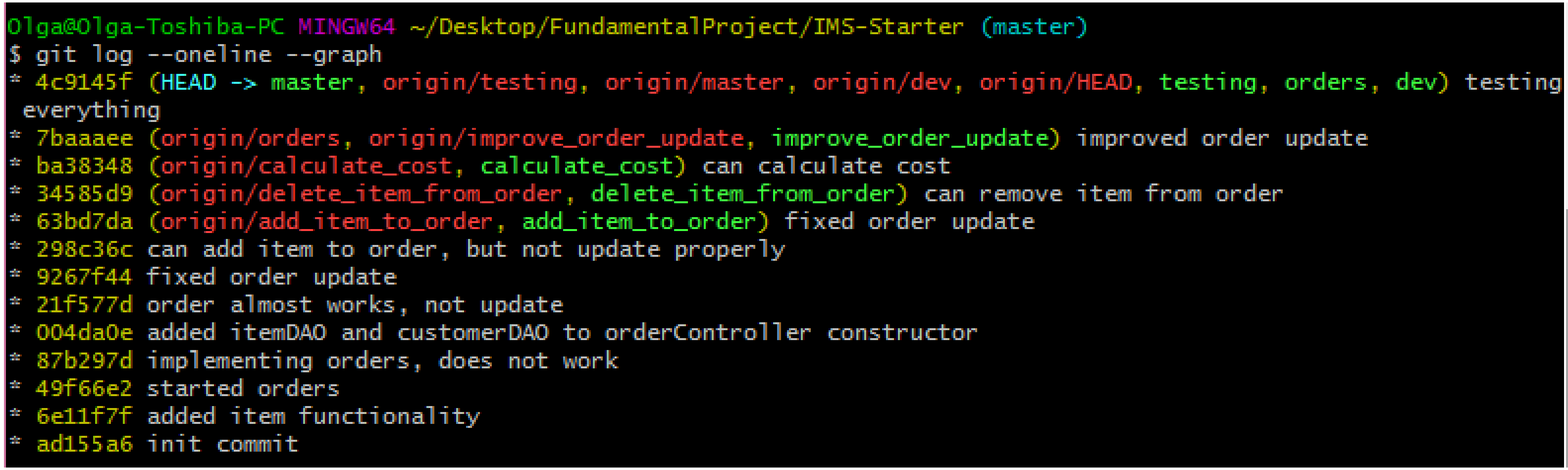
- Score calculated as Likelihood * Damage



Risk	Likelihood/5	Damage/5	Score	How to Fix	How to Mitigate
I get ill	1	5	5	Go to doctor and hope for the best	Don’t touch face
Forgot SQL	2	2	4	Ask for help	Read up
Java issues	2	2	4	Ask for help	Read up
SQL & Java linkage issues	3	2	6	Ask for help	Pay attention
Computer breaks	1	4	4	Borrow one	Always backup!

Version Control

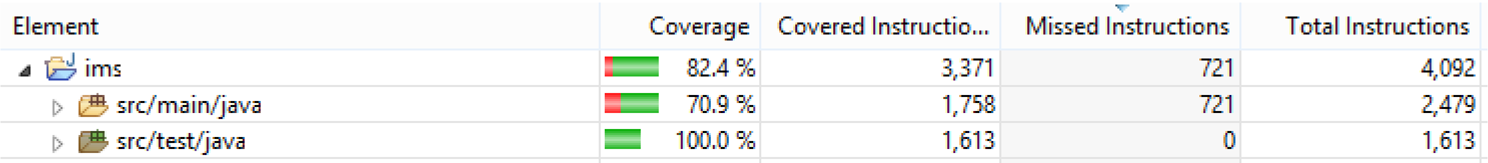
- Git was used for version control
- Used the feature branch model



```
Olga@Olga-Toshiba-PC MINGW64 ~/Desktop/FundamentalProject/IMS-Starter (master)
$ git log --oneline --graph
* 4c9145f (HEAD -> master, origin/testing, origin/master, origin/dev, origin/HEAD, testing, orders, dev) testing
 everything
* 7baaaee (origin/orders, origin/improve_order_update, improve_order_update) improved order update
* ba38348 (origin/calculate_cost, calculate_cost) can calculate cost
* 34585d9 (origin/delete_item_from_order, delete_item_from_order) can remove item from order
* 63bd7da (origin/add_item_to_order, add_item_to_order) fixed order update
* 298c36c can add item to order, but not update properly
* 9267f44 fixed order update
* 21f577d order almost works, not update
* 004da0e added itemDAO and customerDAO to orderController constructor
* 87b297d implementing orders, does not work
* 49f66e2 started orders
* 6e11f7f added item functionality
* ad155a6 init commit
```

Testing

- Testing was done using Mockito and JUnit
- DAO objects (customer, item and order) and controllers (customer, item and order)
- Tested for failures where possible
- Achieved an overall test coverage of 82.4%



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
ims	82.4 %	3,371	721	4,092
src/main/java	70.9 %	1,758	721	2,479
src/test/java	100.0 %	1,613	0	1,613

Sprint Review

Completed	**Left Behind**
CRUD Functionality	Password verification before connecting to database
Cost of order calculation	Address field in customers table
Deleting and adding items to order	Connecting to GCP instance

Sprint Retrospective

What went well:

- Has CRUD functionality
- Relatively easy to use

Room for improvement:

- Currently no way to verify if the password entered is wrong until the user tries to use the database
- User interface could be improved
- More testing could be done

Conclusion

Very educational project. Learned:

- The usefulness of planning
- How to do proper testing (not just smoke testing)
- How to use Java to affect a database