IMS Project SoftwareMarch16

☆ | Personal | Private | AP Invite

Things To Do

Doing

Done

- I have created an empty instance on gcp
- Actually create database tables
- created Customers and items tables with correct primary key
- risk assessment and ERD's
- created SQL statements in java for the customers table allowing CRUD operations
- done the same for items so I can call on CRUD operations from java when running my IMS
- More challenging part of creating an orders table firstly on sql to make sure i have the right syntax
- left out important parts of the orders table so need to add it in and also research constraints
- I now have CRUD statements for customers and items and am ready to implement the same for orders in java
- create, read and delete an order in java as these seem more straight forward than update which will require some more thinking and examples
- I have now got the right syntax and for UPDATE on an order and have applied it to java

during the process i still am not too comfortable about implementing an orderline as im struggling to apply it to java

I have concerns that my order wont be fully functioning hence i need to find a way of compensating for the absence of an orderline

created a very basic ims that can CRUD the three tables using scanner for input

created constraints so order deletes if customer or item gets removed

as a user i want to be asked for an input until i decide to quit

i dont want special characters in the name of a customer or item or numbers to keep it simple

all crud statements should check if the relevant ID exists.

for the orders when updating i believe the order id and product id should be the two inputs as far as ID goes as i want to limit the risk of messing up the wrong customers order like in real life

if an item already exists i dont want duplicates they can just update quantity instead

apply the duplicates rule to orders where a customer can exist >1 but the item must be different if updating or creating. If updating the same item just increase the quantity

I now have the necessary error messages across all tables - items and customers for invalid name and insufficient stock a successful order update will replenish the stock of the item which is no longer required as well as decreasing that of the new item

have a package dedicated to the database connections and the actual SQL statements

have a package for the actual calling of the methods which is also responsible for catching and printing errors

in order to use static analysis provided by Sonarqube I built the project first using Jenkins which called on maven as well (see screenshots in presentation)

My Jar and Fat Jar are correct as are the configuration calls so that my "build now" is successful first time"

have a package for the tables and what defines a customer, item and order

Sonarqube suggested a try and finally for ResultSet to be closed after it's finished to prevent leaks

As part of the sonarqube suggestions and in attempt to increase coverage I had too many void methods which i struggled to test

the ims shouldnt have all the code and the methods so i need to split them into the relevant packages

So i changed the type of all methods to return a string rather than just Sys.Out("created") etc

have all ResultSet calls closed in a try-finally block to try and comply with sonarqube implement more mockito tests to try and mock what would happen when invalid inputs are put in for the 3 tables in terms of CRUD

increase the test coverage and get rid of the red lines on sonarqube which still say not covered by tests

bugs decreased on sonarqube aswell as code smells but do not know what to do to make them even lower

upload all necessary docs to git hub including a git ignore and fully complete README

have the necessary screenshots from the build in jenkins aswell as separate sreenshots of the packaging and Fat JAR

create screenshots and presentation for Monday 27/04

Rehearse the presentation and have screen shots of the special features as the demo will not be long enough to show those