Name: Imogen Hay URN: 6579619 Username: ih00264

COM1028 Software Engineering

Coursework Assignment

20/05/2020

I.hay

1 Code and Tests

To output the results for each requirement please run App.java and to run the test cases use AppTest.java. During development I have been using GitLab for version control which you can see in my repository.

2 Diagrams

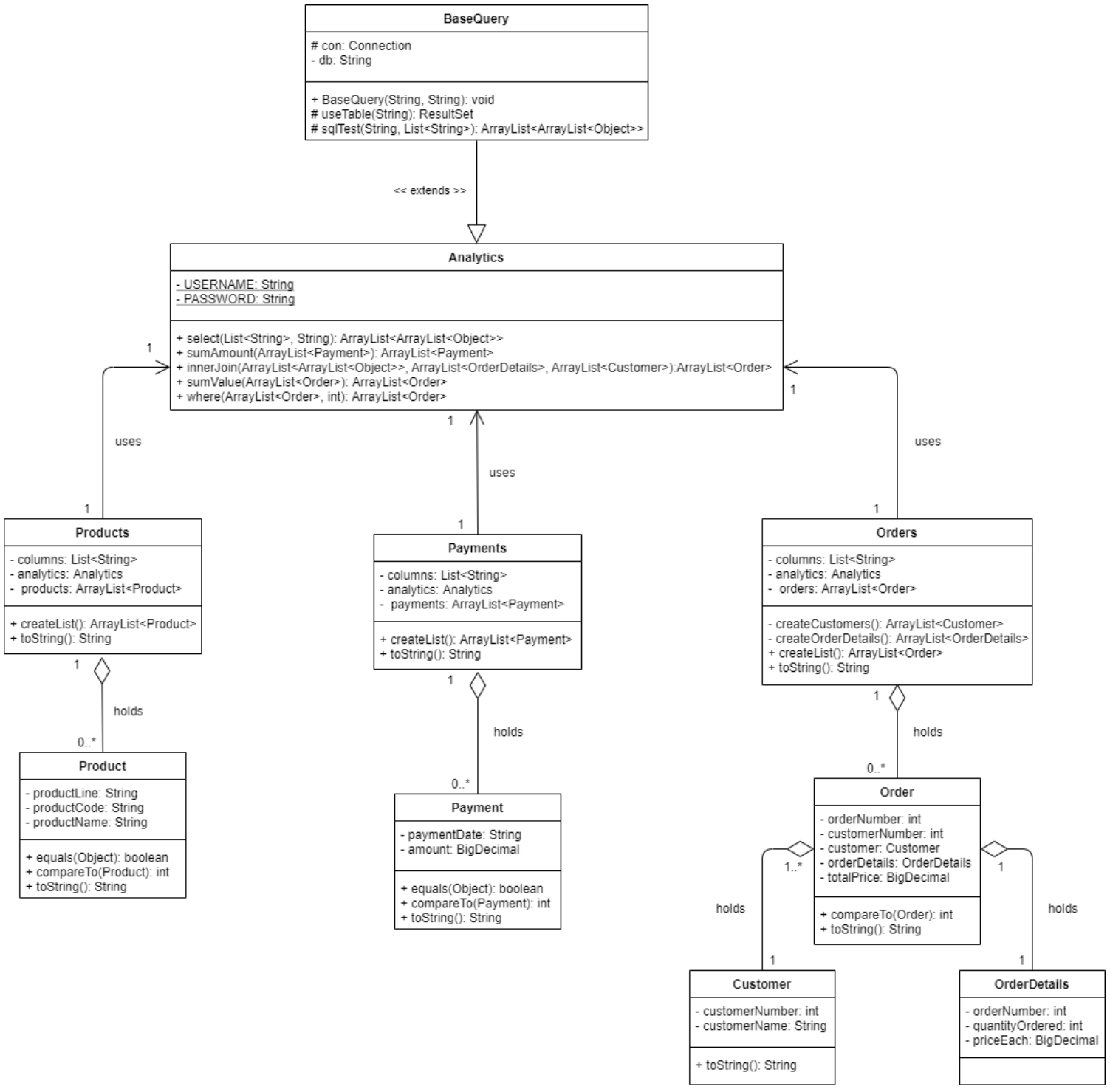
In the class diagram I have only included constructors that do more than set up a few internal data attributes. I have also not included the App class since it only contains a main method so is not relevant.

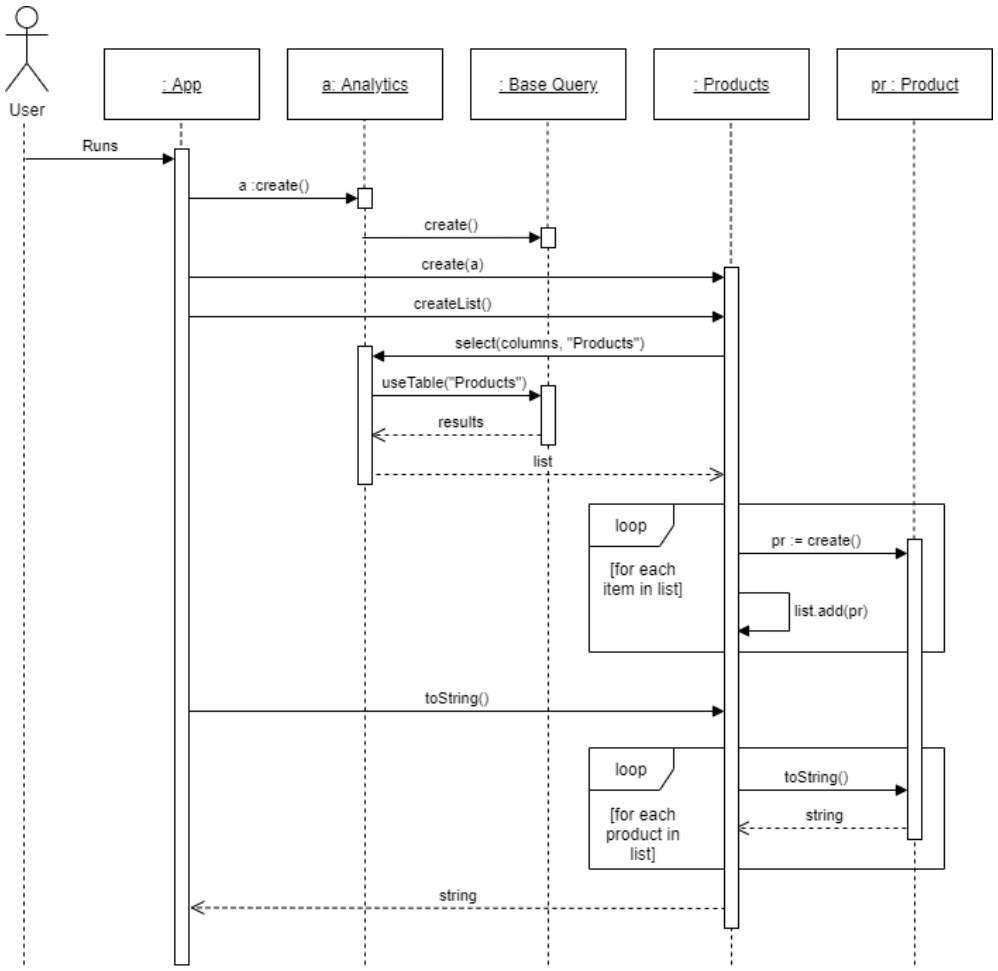
The Analytics class uses BaseQuery to connect to the database and get a ResultsSet for a specific table. I have a class for each table I will access in the database.

I have classes for each requirement which create a list containing all relevant records as objects E.g. 'Products' makes list of 'Product' objects, one for each row in the product table.
The Analytics class is used to perform the equivalent of SQL commands written in java such as select, where and sum which are all methods used to produce data relevant to the requirements.

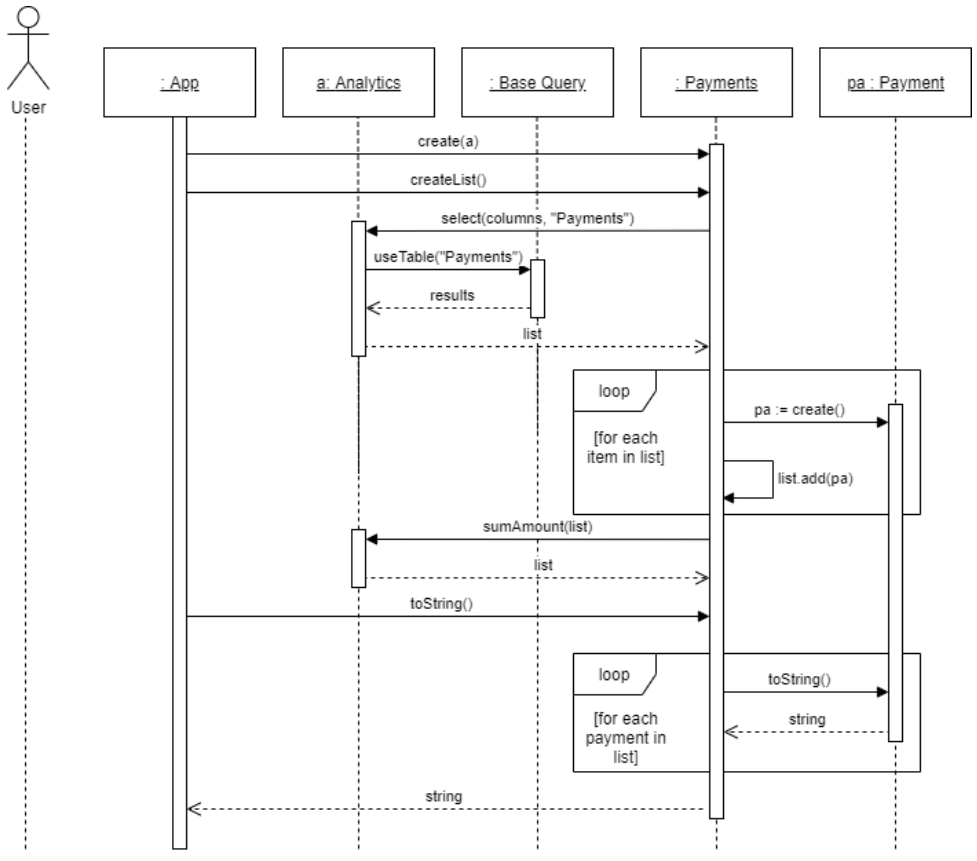
I designed my project this way to keep a balanced workload and short communication chains between objects by not creating one class for too many functions.

Name: Imogen Hay **URN:** 6579619 **Username:** ih00264

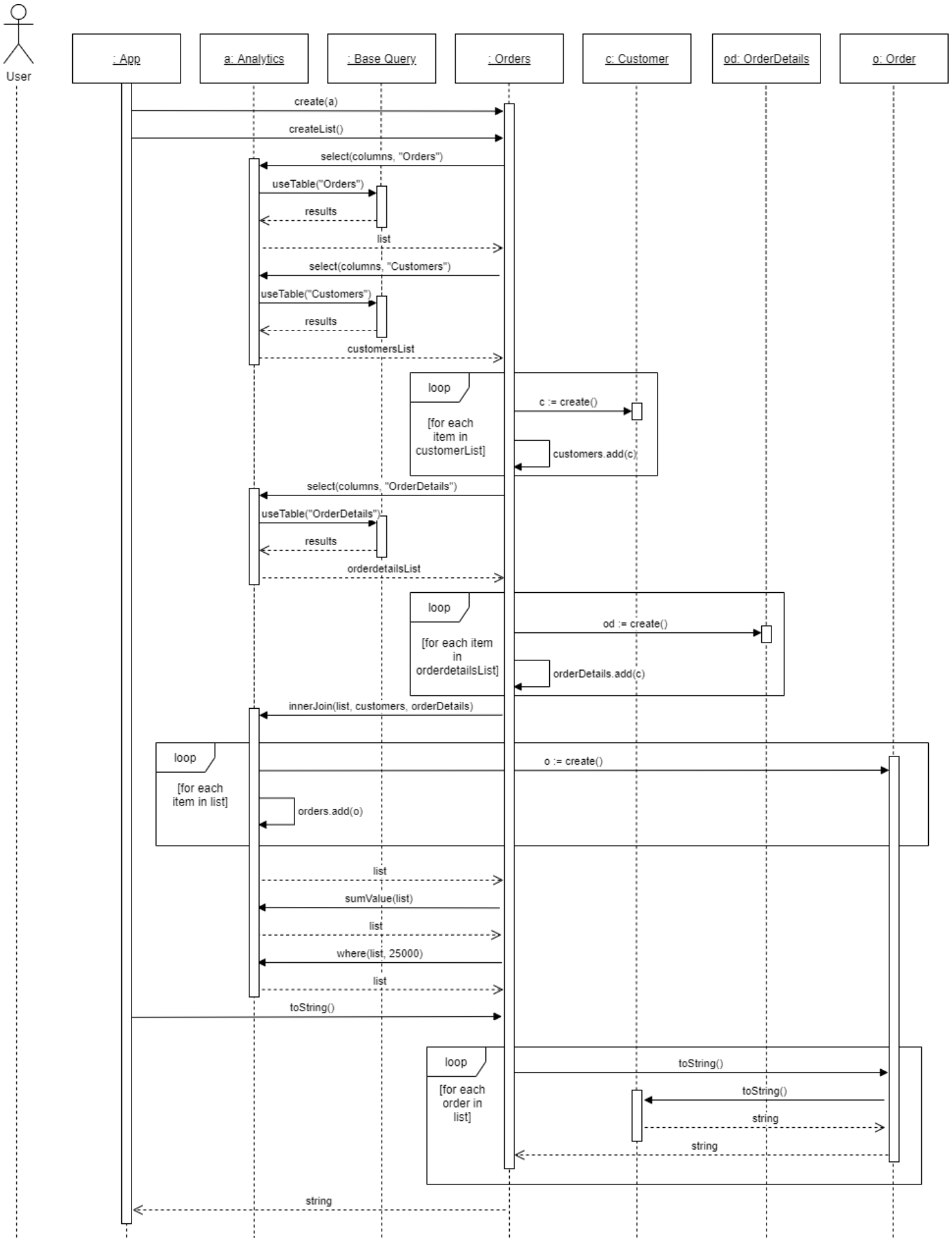




Sequence Diagram for Requirement 1



Sequence Diagram for Requirement 2



Sequence Diagram for Requirement 3

The sequence diagrams show how during analysis I divided the system into manageable modules using divide and conquer. Each sequence diagram follows on from the one before. Requirement 1 and 2 only needed access to one table so use less interactions whereas requirement 3 requires an inner join between 3 tables. The Orders class creates lists of objects for each table then uses the Analysis class to join them into one list. I created objects for each record to make my program more adaptable since values can be easily changed and retrieved using mutator and accessor methods.

3 Tests Case Explanation

During testing my strategy was to create a test class for each requirement including multiple test cases to make sure the relevant classes and output worked as they should. To determine which tests passed or failed I used assertTrue and assertEquals so I first tested that they worked accurately.

To test the output for each requirement I used the SQL statements:

```
1: SELECT productLine, productCode, productName FROM products ORDER BY productLine

2: SELECT paymentDate, SUM(amount) FROM payments GROUP BY paymentDate ORDER BY
paymentDate

3: SELECT customers.customerName, orders.orderNumber, SUM(orderdetails.quantityOrdered *
orderdetails.priceEach) FROM orders INNER JOIN customers ON
customers.customerNumber=orders.customerNumber INNER JOIN orderdetails ON
orderdetails.orderNumber=orders.orderNumber GROUP BY orders.orderNumber HAVING
SUM(orderdetails.quantityOrdered * orderdetails.priceEach) > 25000 ORDER BY
upper(customers.customerName)
```

Then converted the ResultSet into a list of objects which I could compare with my programs output. Since I was creating objects within this test case, I also needed to test creation of objects. I then checked whether the objects for each table could be constructed correctly and that its accessor methods returned the appropriate output.

To increase my programs robustness, I checked for SQL exceptions wherever the database was being accessed since problems connecting to a database could be a common problem for many users. My program also works if no data is found for a query which produces an empty list instead of a null list. This means that other functions using the list still work and a null pointer exception does not need to be used.

4 Word Count

Diagrams: 257
Test Case Explanation: 243
Total: 500