L-Università ta' Malta
Faculty of Information &
Communication Technology

Assignment Report

Manwel Bugeja

December 27, 2019

Contents

0.1 Task 1

0.1.1 The Design

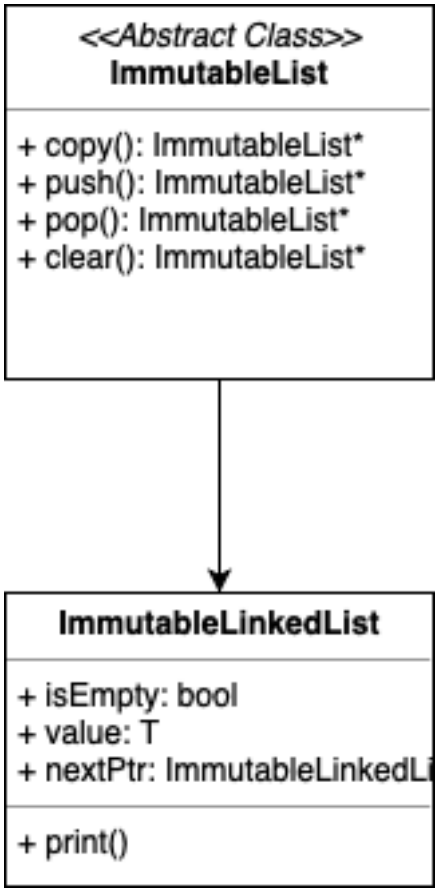

Figure 1: Task 1 UML Diagram

In this section a pure abstract class was created called 'ImmutableList' containing the required functions. Another class was created called 'ImmutableLinkedList' which inherited from the first class. This was done to implement the methods of the abstract class for a structure called a linked list. The class has 2 constructors and 3 variables.

The constructors

The key to this design is the fact that there are actually two constructors but one of them is inaccessible by the user. The hidden constructor has the ability to create nodes with custom values for the variables isHead,

value and nextPtr whose use is explained later. On the other hand the public constructor is only able to set the value of a head node. The public constructor is implementer using the private, passing isHead as true and nextPtr as nullptr and the value containing what is passed by the user.

isHead

This variable is a boolean type which indicates if the current node is the head of a list or not. This is needed to distinguish the head from other nodes of the list. isHead enables the possibility to have an empty list as the value head node is completely ignored by the api.

value

This variable contains the value of the node which is the same type as that indicated at the creation of the immutable list.

nextPtr

This variable contains a pointer towards the next node of the immutable list.

0.1.2 Testing

Initial testing

To test the immutability of a list, a list was created with some initial values. Then a new list was created using the old list and the push method. A third list was created using the first one and the push method with a different value passed as a parameter. All three lists were printed as should.

Concurrent testing

The list was also tested concurrently. A function was created that receives a list and a value to append. Two threads were created and the function was run in each thread with a different value passed. After the threads were joined, the list that was passed was printed to show that it was unchanged.

0.2 Task 2

0.2.1 The Design

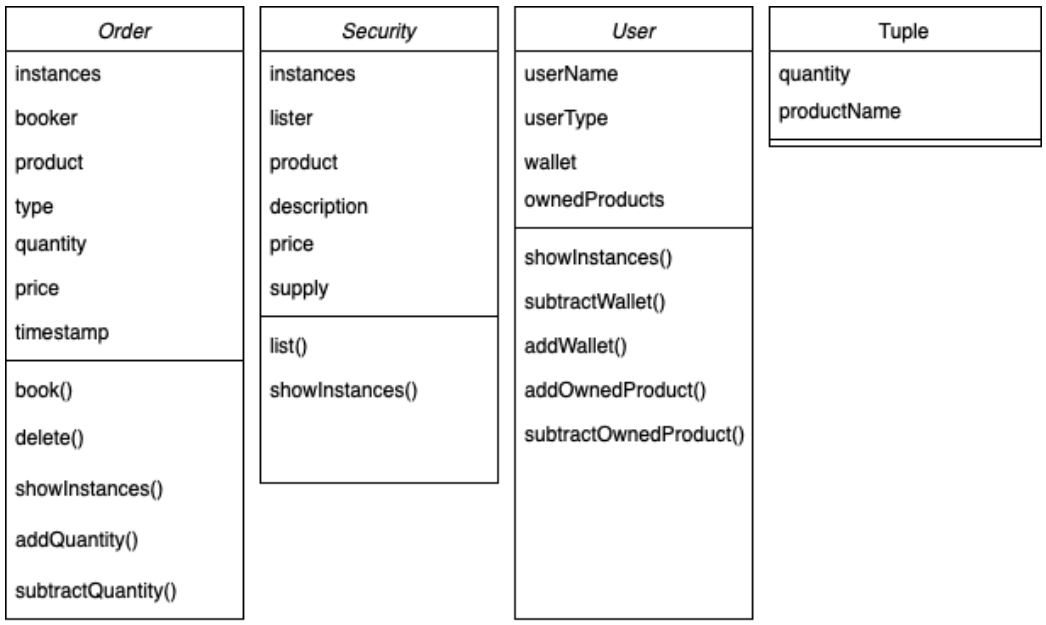

Figure 2: Task 2 UML Diagram

For this task, three classes: Order, Security and User were created and each contained an ArrayList to keep track of all objects created. Due to the fact that in this design, buy orders and sell orders are identical except for their type (buy or sell), a String attribute 'type' was created and inheritance was not used. This was also the case for listers and traders in the class 'User'.
The matching engine simply compares all sell orders with buy orders, checks when the product matches, and if the amount of the buy is less than or equal to the amount of the sell, it completes the trade and adjust the required fields.

0.2.2 Testing

Listing 1: Task 2 sample output

```
0. quit
1. exchange
```

```
2. register
3. list
4. book
5. cancel order
6. show all

2
registering
Enter name
Manwel
Please select type:
1. lister
2. trader
1
Enter wallet:
200000
Registering Manwel as lister
Registered lister
0. quit
1. exchange
2. register
3. list
4. book
5. cancel order
6. show all

2
registering
Enter name
Christian
Please select type:
1. lister
2. trader
2
Enter wallet:
200000
Registering Christian as trader
Registered trader
0. quit
1. exchange
2. register
```

```
3. list
4. book
5. cancel order
6. show all

3
listing
Enter user: Manwel
Enter product: building
Enter description: buildings in mxlokk
Enter price: 100
Enter supply: 5
0. quit
1. exchange
2. register
3. list
4. book
5. cancel order
6. show all

4
booking
Enter user: Christian
Enter product: building
Select a type:
1. Buy
2. Sell
1
Enter quantity: 3
Enter price: 1000
0. quit
1. exchange
2. register
3. list
4. book
5. cancel order
6. show all

6
Users:
Name: Manwel
```

```
Type: lister
Wallet: 200000.0

Name: Christian
Type: trader
Wallet: 200000.0


Securities:
Lister: Manwel
Product: building
Description: buildings in mxlokk
Price: 100.0
Supply: 5


Orders:
Booker: Manwel
Product: building
Type: sell
Quantity: 5
Price: 100.0
Timestamp: 1577462020210

Booker: Christian
Product: building
Type: buy
Quantity: 3
Price: 1000.0
Timestamp: 1577462045590


0. quit
1. exchange
2. register
3. list
4. book
5. cancel order
6. show all

1
```

```
exchanging
0. quit
1. exchange
2. register
3. list
4. book
5. cancel order
6. show all

6
Users:
Name: Manwel
Type: lister
Wallet: 203000.0

Name: Christian
Type: trader
Wallet: 197000.0


Securities:
Lister: Manwel
Product: building
Description: buildings in mxlokk
Price: 100.0
Supply: 5


Orders:
Booker: Manwel
Product: building
Type: sell
Quantity: 2
Price: 100.0
Timestamp: 1577462020210


0. quit
1. exchange
2. register
3. list
```

```
4. book
5. cancel order
6. show all
```

0.2.3 Limitations

For a large amount of orders, it would have been better if they were stored in a separate ArrayLists, because as it is, the matching engine has to go through all orders twice. This is very inefficient as opposed to comparing the list of sell orders with the list of buy orders.

0.2.4 Polymorphism

To implement the log, a class can be created with a new list structure with only an add method (and no remove). Every time 'book()' or 'list()' or 'removeOrder()' is called, the action is added to this new list. Tampering this list can be avoided with proper encapsulation.

0.3 Task 3

0.3.1 The Design

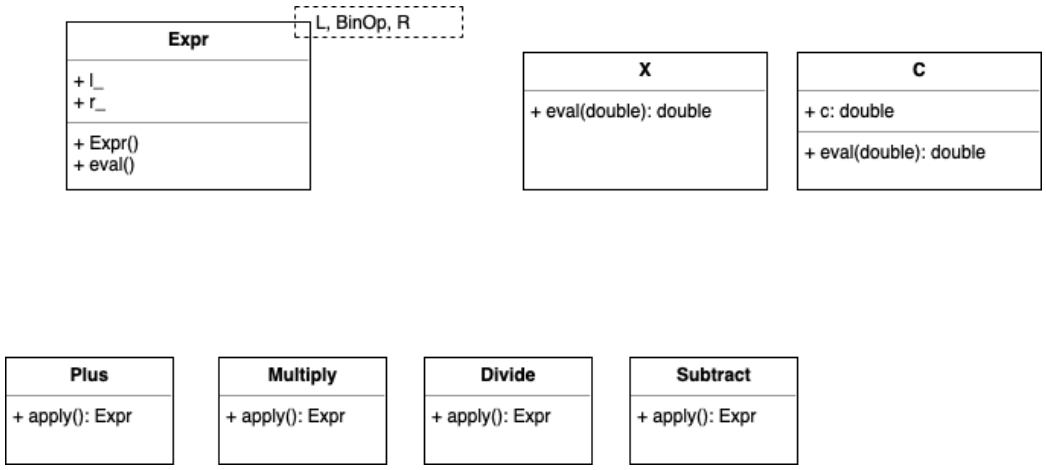

Figure 3: Task 3 UML Diagram

For this task, a template class 'Expr' was created. The purpose of this was to represent the AST and unwind it as should. Template classes for the

operators were also created which server a similar purpose. The operators were also overloaded using templated functions.
For the intergration function, a template function was created that received 'Expr' along with the other requirements as parameters. The function implemented the trapeziod rule.

0.3.2 Testing

For the testing, an expression that used all operators was used and the output was checked. As for the integration part, an expression was passed and checked against the correct answer.