



Advance Software Engineering, F21AS

Heriot Watt University

Stage 1

**Written by Kamontorn Khamrun (H00144659),
Marcin Kopacz (H00176255) and
Stian Dalviken (H00024901)**

Table of Contents

1. Breakdown of responsibilities	3
1. Development Plan	3
2. Basic Application	3
3. Basic application implementation	3
4. Report	3
2. Status report.....	4
3. UML Class Diagram	5
4. Data Structures	6
5. The functionality of the program.....	6
a) How discounts are calculated?	6
b) Test descriptions	6
I. What errors in the input files do you check for?	6
II. Which errors did you catch with exceptions?	7
c) JUnit testing	8
I. DishItem class and DishOrder class	8
II. ManagerClass class.....	8
III. OrderCollection class	8
6. Development report.....	9
a) From the original Development Plan	9
1) Schedule:	9
2) Class Diagram:	10
b) Final Development Plan.....	11
c) Conclusion	12
Attachment 1	13

1. Breakdown of responsibilities

Team members:

Kamontorn Khamrun (H00144659)

Marcin Kopacz (H00176255)

Stian Dalviken (H00024901)

We were discussing problems on 7 meeting during Stage 1 of this project, It is hard to properly calculate individual contributions, as we were helping each other a lot. Contribution to project was even between group members. Below is an overview of responsibility and work, but in addition to this we also helped each other on almost all points.

1. Development Plan

- MS Project: Stian(60%), Marcin (20%), Kamontorn(20%)
- Input files and research iterative development: Kamontorn(100%)
- Class Diagram: Marcin(75%), Stian(25%)

2. Basic Application

- Set up Git: Marcin(50%),Stian(40%), Kamontorn(10%)
- Set up Dropbox: Stian(10%), Kamontorn(90%)
- JUnit preparation – tests for DishOrder, DishItem (Stian100%)

3. Basic application implementation

- 3 Basic Type Classes:
 - DishItem class: Stian(100%)
 - DishOrder: Marcin(100%)
 - Dish class: Kamontorn(100%)
- Collection Classes:
 - MenuSet class: Stian(100%)
 - OrderCollection class: Marcin(100%)
 - ManagerClass class: Marcin(10%), Kamontorn(90%)
 - Main class: Marcin(100%)
 - GrandTotalGUI- Kamontorn(80%),Marcin(20%)
- Basic features:
 - Menu by Category: Stian(100%)
 - Freq report, dishes not ordered with JUnit: Marcin(100%)
 - Table Summary and GUI with JUnit: Kamontorn(100%)
- Statistical Methods
 - Avg Spend per Category: Kamontorn(100%)
 - Best Selling Dish per Category: Marcin(100%)
 - Avg spend per person per table: Stian(100%)

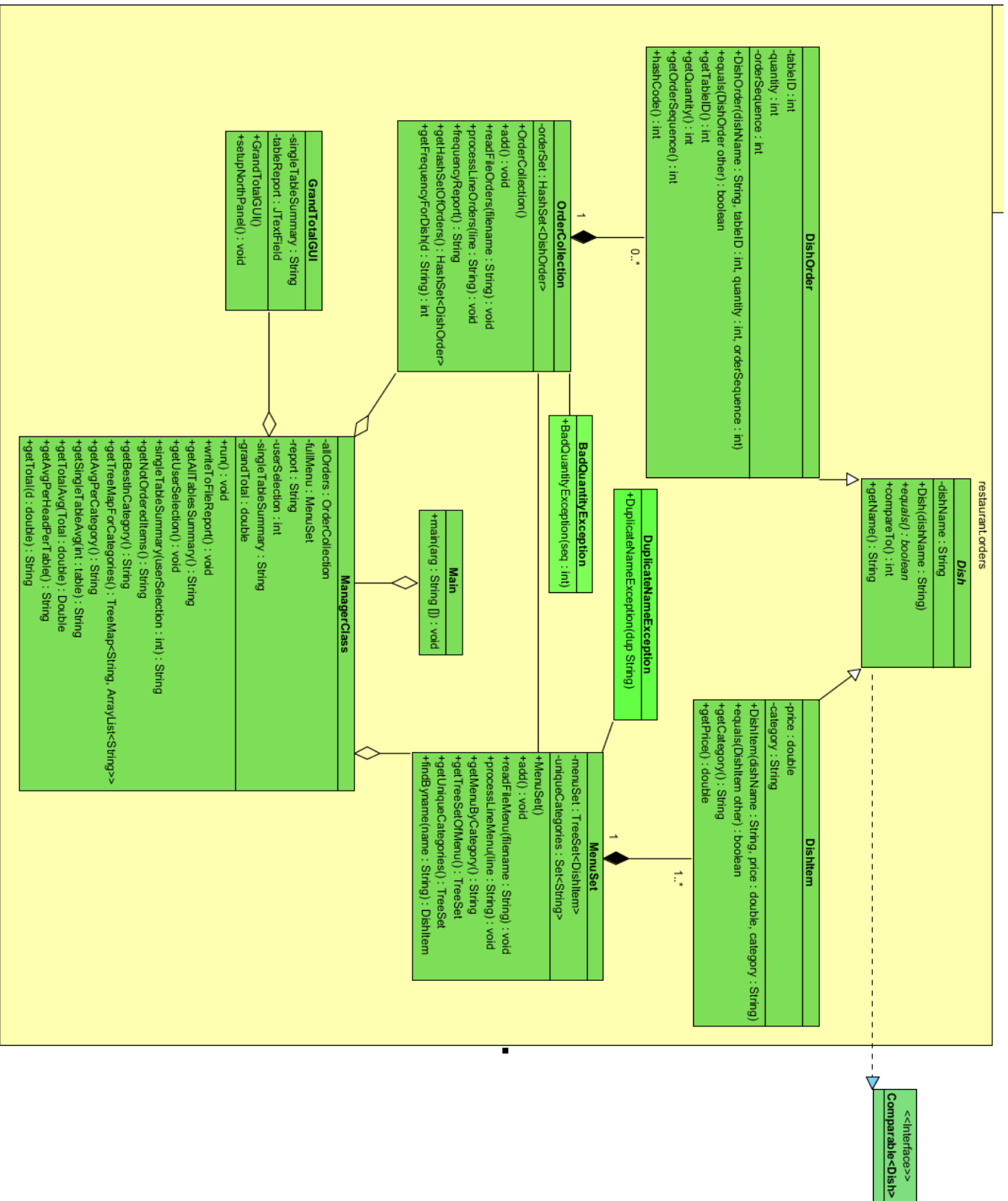
4. Report

Marcin (1,2,3,6a), Stian (4,5c, 6b), Kamontorn(5a,5b)

2. Status report

Our application fully meets all specifications specified in this assignment.

3. UML Class Diagram



4. Data Structures

We decided to use three different types of data structures, HashSet, TreeSet, and TreeMap. We used HashSet in the OrderCollection class, and it's used to make a HashSet of DishOrder objects. We have used TreeSet in the MenuSet class for two different uses, one for storing DishItem objects, and one for storing names of categories in a String variable. In the ManagerClass we are creating a TreeMap where categories are keys and dishes are values.

We wanted to get familiar with three different types of data structures in this stage. We might be changing the use of the different data structures in Stage 2, as we now have a better understanding on how they work and what they should be used for. But since this is quite a small program, the speed of a HashSet compared to a TreeSet is not really noticeable. If this had been a bigger program on the other hand, switching out the TreeSet in favour for a HashSet, where items does not need to be sorted, could've been beneficial with regards to the speed.

Using TreeSet in the MenuSet class is also beneficial for the getMenuByCategory() method, that alphabetically orders the items.

5. The functionality of the program

a) How discounts are calculated?

Discounts are calculated based on the amount of the input parameter "Total", and is done as follows:

- For each table, if the total of bill is more than £150, but less than £300, a 5% discount will be applied to the bill
- For each table, if the total of bill is equal or more than £300, a 10% discount will be applied to the bill

b) Test descriptions

1. What errors in the input files do you check for?

- Check if the input file contains correct format or not. For example, it will ignore lines in an error with NumberFormatException while it is trying to convert a String to an Integer
- Check if the input file contains enough items before adding to them to the collection. For example, if there are one or two missing items in the input file, ArrayIndexOutOfBoundsException will be caught
- If a quantity of the dish of the table is more than 10, it will throw the exception BadQuantityException
- If the menu input file contains a duplicate of a dish name, it will ignore the line and show a message "This dish name is already in list".
- There are also some error-checking to validate parameters in the constructor of Dish, DishOrder and DishItem

II. Which errors did you catch with exceptions?

Several exceptions are used in the program as listed below:

- 1) `IllegalArgumentException`, an unchecked exception, is used in constructors such as `Dish`, `DishOrder` and `DishItem` to ensure that parameters are valid for creating an object. The valid parameters are specified below:
 - a) `DishName` cannot be left blank in `Dish`
 - b) A table ID has to be more than 0 and a quantity has to be more than 0 in `DishOrder`
 - c) Price has to be more than 0 and a category cannot be left blank in `DishItem`

Otherwise an `IllegalArgumentException` will be caught in the `ManagerClass`' run-method, with a message for each parameter and "Details not added".

- 2) `NumberFormatException`, an unchecked exception, is used for catching the number format error in the input. For example, while it is trying to convert a `String` to an `Integer`. This exception is used in the `MenuSet` Class at `ProcessLineMenu()` and in the `OrderCollection` Class at `ProcessLineOrders()`
- 3) `ArrayIndexOutOfBoundsException`, an unchecked exception, is used for missing items in the input file during `ProcessLine` method and trying to get a value in the array which its index does not exist. This exception is used in the `MenuSet` Class at `processLineMenu()` and in the `OrderCollection` Class at `processLineOrders()`
- 4) `BadQuantityException`, an unchecked exception, is our own exception class. It will throw when quantity is more than 10 in one dish order. This exception is used in the `OrderCollection` Class at `add()` and will get caught in `processLineOrders()`
- 5) `DuplicateNameException`, a checked exception, is our own exception class. It will throw when there is a duplicate dish name. This exception is used in the `MenuSet` Class in the `add()` method, and will be caught in `processLineMenu()`
- 6) `FileNotFoundException`, a checked exception, is thrown if the file is not found

c) JUnit testing

We used JUnit to test different types of methods, and one constructor, in 4 different classes. In the code you will see that they are all very well commented, but a short description on what has been done, follows:

I. DishItem class and DishOrder class

In these two classes, the equals methods were tested. We tested to see if the methods returned false when all values in the comparing object were unequal, and then one test method each to check one unequal value, one by one. We also tested to see if they returned true if the object were actually equal, as well. What actually happened in the DishOrder class was that we found a mistake in the code. The mistake made it so that all test methods returned false, and the reason being that not every single value in the equals method was checked. We then corrected the code based on this.

II. ManagerClass class

In this class, the getTotal method was tested. The getTotal method is used to calculate the discount, and the discounted Total, with an input variable of type double. If total is more than 150, but less than 300 a 5% discount applies. If total is equal to or more than 300, a 10% discount applies. Otherwise there are no discounts. The logic in the method is being tested to see if the discounts are being correctly added, as well as being right mathematically, and not added when the input value is less than 150.

III. OrderCollection class

In this class we have a test to see if the BadQuantityException is getting thrown for when the quantity is more than 10. Quantity is the number of dishes in one order.

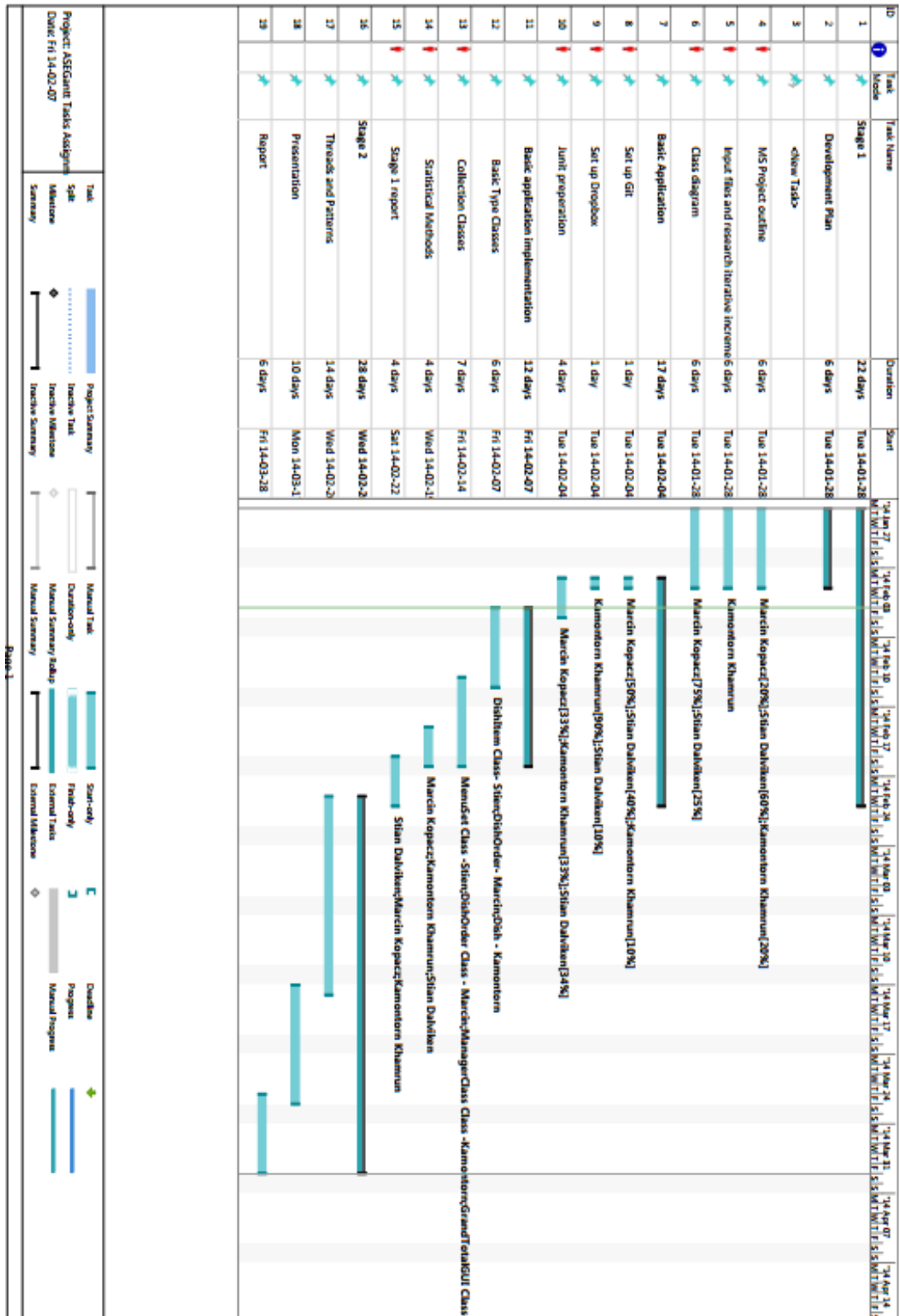
getFrequencyForDish method is being tested to see if it returns the correct amount of frequency for a given dish. There is also a test to check that the method returns 0 when a dish isn't found.

The frequencyReport in this class is also tested. It's checking to see if the method produces the correct String.

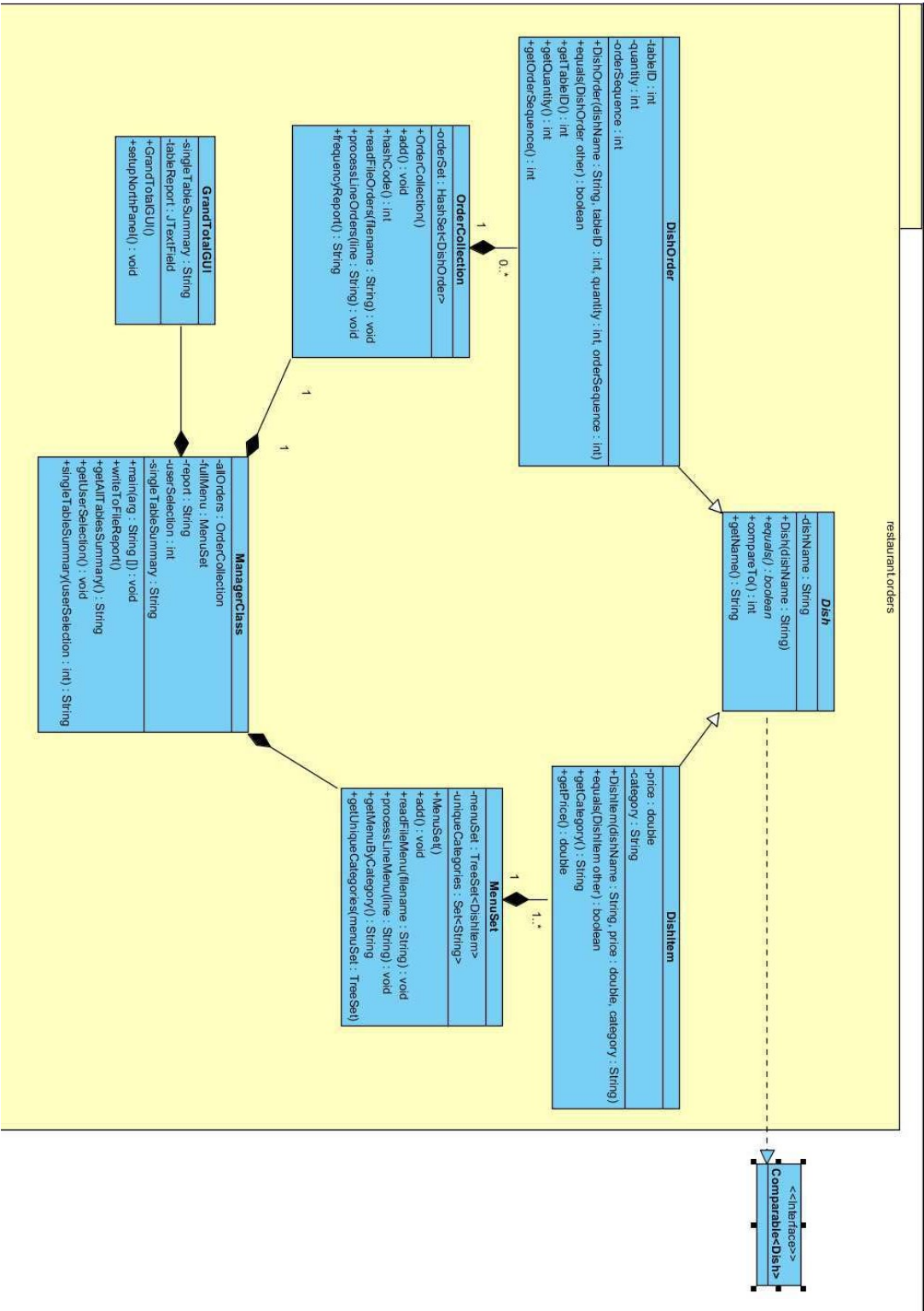
6. Development report

a) From the original Development Plan

1) Schedule:



2) Class Diagram:



b) Final Development Plan

Development schedule:



We had group meetings every Tuesday, as shown in the Gantt diagram above, and in addition to that, we also met on some Fridays as well.

Schedule and iterative process were close to plan. We had enough discipline to make meeting in given times. Some requirements like “other statistical methods” has not been taken in consideration in initial development plan. We added them during the implementation period, after carefully deciding what to include.

Because we didn't read carefully enough the requirements, we missed part about exceptions. As soon we notice that we added two exceptions `DuplicateNameException` (checked) and `BadQuantityException` (unchecked) to meet requirements.

Some methods like `getAllTablesSummary()`, `frequencyReport` were too big and because of that we divided them into smaller methods to increase code reusability.

We also added get methods to access to collections of orders and menu to follow data hiding rules and to remove public variables.

The communication between the group members was done using a group in WhatsApp (a mobile phone app used for text messages), Dropbox to exchange documents, and online storage of code with version control using Git on Bitbucket.org. In Attachment 1, in this document, you can see screenshots of our commits from Bitbucket.org, using git.

c) Conclusion

We managed to fully meet the requirements for this stage. We tried following our development plan as much as possible. Our schedule was very close to the plan, but with regards to the design of the code, it could've been done better and more efficient if we had more time, flexibility and if the functional requirements had been better specified.































Code re-usage could've also been done better. As we were working on separate isolated classes and methods, it was hard to predict any code that more than one method would make use of, so that some of the code has been duplicated. If we had done this again, we would've broken down the methods in detail together before starting the coding part of the project. Methods/class design changes was very time consuming, as we would have to update the development plan every time we made a change, and also communicate the changes to the other group members. So it was sometimes hard to track other group members' changes. As each group member made the code separately, we did not check each other's code after it had been made. Each group member had their own brick to build, but we all lacked a proper overview of the whole building.































We also tried to make use of a planned iterative development process, but we all had to complete our deliverables to be able to see any changes to the project.
























Even though there have been a couple of obstacles along the way, this stage of the assignment has gone well, and we are all satisfied with the result.

Attachment 1

Screenshots of commits using Git as version control, taken from Bitbucket.org:

Author	Commit	Message	Date
 Marcin Kopacz	7de6201	small changes	2 days ago
 Chyzwar	585f4b9	rm menuSet	2 days ago
 Chyzwar	b06e697	comments	2 days ago
 Kamontorn Kha...	89a06bc	Added comment to GrandTotalGUI and updated manager class	3 days ago
 Chyzwar	495b6ec	removed public variables	3 days ago
 Chyzwar	f9d8f65	test for orderCollection	3 days ago
 Kamontorn Kha...	f56b5d5	added comments about exception	4 days ago
 Kamontorn Kha...	d0f16e0	add comment	4 days ago
 Kamontorn Kha...	b1655ad	Catch exception in managerclass	4 days ago
 Kamontorn Kha...	24d7be5	added exception to constructors	4 days ago
 Kamontorn Kha...	18e9e8f	TestManagerClass updated	4 days ago
 Kamontorn Kha...	af204dc	Change getTotal to return ArrayList<Double> for easier Junit test case	4 days ago
 Kamontorn Kha...	25e09b5	modify constructor in ManagerClass	4 days ago
 Chyzwar	e0de35e	added exceptions	5 days ago
 Kamontorn Kha...	bbb6ba0	Added java doc style comments	2014-02-21
 Chyzwar	8dcb8f6	menuSet test 0.6	2014-02-19
 Chyzwar	9ea6bb5	menuSet test starting	2014-02-19
 Stian Dalviken	3e8a8b9	Added methods calculating average spend per head per table	2014-02-17
 Stian Dalviken	13b3796	RestaurantReport.txt removed form project	2014-02-17
 Stian Dalviken	7c3b3b3	Added restaurantReport.txt to be ignored by git	2014-02-17
 Kamontorn Kha...	c0db63f	updated getAvgPerCategory	2014-02-17
 Kamontorn Kha...	db83af2	Done with getAvgPerCategory() !	2014-02-17
 Chyzwar	b22baa2	comment and details	2014-02-17
 Chyzwar	c14d7a7	getBestInCategory() done	2014-02-16
 Chyzwar	1563a5e	getBestInCategory() almost	2014-02-16
 Chyzwar	5e7545c	getuniqueCategories	2014-02-16
 Chyzwar	aecab6c	getNotOrderedItems	2014-02-16
 Stian Dalviken	48c2032	Finished the getMenuByCategory method, and commented it	2014-02-15
 Chyzwar	2aad2f7	not ordered items	2014-02-14
 Chyzwar	035bd87	small fix	2014-02-14

Author	Commit	Message	Date
 Chyzwar	74857a6	added get methods for collections	2014-02-14
 Kamontorn Kha...	558f3cd	NumberFormatException and detect if there is no order in Table	2014-02-13
 Kamontorn Kha...	1f6af5b	Done with GUI!	2014-02-13
 Kamontorn Kha...	b9e871e	getAllTablesSummary() done! removed String singleTableSummary to be local variable in the	2014-02-13
 Kamontorn Kha...	0c74768	singleTableSummary Done! created new method getTotal to calculate total with discount.	2014-02-13
 Kamontorn Kha...	5764dc6	change singleTableSummary to display in alphabetical order	2014-02-13
 Kamontorn Kha...	58f649d	Input more orders (Table 4)	2014-02-13
 Kamontorn Kha...	e2b284e	Change menuSet to public for singleTableSummary. Update managerClass	2014-02-13
 Chyzwar	25d554b	little	2014-02-13
 Chyzwar	ea1e36a	array list fix	2014-02-13
 Chyzwar	745cd80	clean import	2014-02-13
 Chyzwar	53cce91	freqreport method	2014-02-13
 Stian Dalviken	7cfe2ac	Deleted the not working test class for Dish	2014-02-13
 Chyzwar	db3527c	finally it works	2014-02-13
 Chyzwar	8088253	Merge branch 'master' of https://bitbucket.org/AdvSoftEng/assignment-1-stage-1	2014-02-13
 Chyzwar	871472b	added more to manager class	2014-02-13
 Chyzwar	b12acbf	manager class update, input files	2014-02-13
 Stian Dalviken	53887a4	Test class for Dish added, but cannot seem to create an object of Dish as it's abstract	2014-02-12
 Stian Dalviken	14ffa77	Fixed the equals method to check for TableID and Quantity equality	2014-02-11
 Stian Dalviken	d9d39aa	Added test class for DishOrder	2014-02-11
 Stian Dalviken	ce45512	Specified the package import only to the class being used	2014-02-11
 Stian Dalviken	f8da968	Merging branch to master	2014-02-11
 Stian Dalviken	d0645b5	Added more test methods to test the equals method properly, and added javadoc comments	2014-02-11
 Kamontorn Kha...	6e472be	manager class/ writing to file	2014-02-11
 Stian Dalviken	1c5a949	Fixed the test method	2014-02-11
 Chyzwar	20cd9cb	Merge branch 'master' of https://bitbucket.org/AdvSoftEng/assignment-1-stage-1	2014-02-11
 Stian Dalviken	de8df0c	Merging branches	2014-02-11
 Stian Dalviken	652d7be	Added test package, and test class for DishItem	2014-02-11
 Chyzwar	08e5b2f	hashcode details	2014-02-11
 Chyzwar	f3cc0dc	hashcode moved to dishOrder	2014-02-11

Author	Commit	Message	Date
 Chyzwar	25c6e3a	file reading in order collection	2014-02-11
 Chyzwar	37917f5	small fix in menuSet	2014-02-11
 Chyzwar	4b5b848	input file upload	2014-02-11
 Chyzwar	a3cb704	file reading in menuSet	2014-02-11
 Chyzwar	09dd22f	manager class fix	2014-02-11
 Chyzwar	7fbe760	small update main class	2014-02-08
 Chyzwar	090a6e0	constructors made	2014-02-08
 Chyzwar	78ac811	Manager class and fix	2014-02-08
 Chyzwar	30222d9	Merge branch 'master' of https://bitbucket.org/AdvSoftEng/assignment-1-stage-1	2014-02-08
 Kamontorn Kha...	4d1f864	Create JAVA doc comment	2014-02-08
 Chyzwar	6b2df17	small stuff	2014-02-07
 Chyzwar	2bef11b	Small tweaks in DishOrder	2014-02-07
 Stian Dalviken	0b08964	Some changes to the DishItem class, and added javadoc style comments to the same class	2014-02-07
 Stian Dalviken	8aa57f7	DishItem class done	2014-02-07
 Kamontorn Kha...	84bace6	modify Dish class, create constructor. comment out compareTo() : boolean method because	2014-02-07
 Marcin Kopacz	022e1ea	class DishItem final and some small changes to Dish class	2014-02-06
 Marcin Kopacz	c8b6466	class DishItem updated	2014-02-06
 Stian Dalviken	db2c184	Added .DS_Store to be ignored by git. These are hidden mac os files	2014-02-06
 Stian Dalviken	49a7e3d	Removed the bin folder with the class files from the repo, as it's not needed	2014-02-05
 Stian Dalviken	85c8f66	Added .gitignore file to the project, and removed the class files from the repo, as it will	2014-02-05
 Stian Dalviken	0af8174	Quickly started on the dishItem constructor	2014-02-04
 Stian Dalviken	a530eec	Fixed an error in the build.xml file. Linking to file that does not exist	2014-02-04
 Marcin Kopacz	989f9a7	basic java classes	2014-02-04