SALES-COMISSION-APP

OVERALL REPORT

VERSION <1.0>

Anastasia Varoucha -4288

Katerina Papaxristopoulou-4148

–

TABLE OF CONTENTS

INTRODUCTION

Here is a list of tasks that we will deal with in this project, based on the packages:

data:

1.Remove Coat, Shirt, Skirt, Trouser classes.

2.1.Rename Agent Class.

2.2.Remove Duplicate Code in Agent Class.

2.3.Replace deprecated data structure.

3.1 Change name of FileAppender, FileAppenderTXT, FileAppenderXML.

3.2.Move classes that perform IO operations in the right packages.

4.Encapsulate the primitive data of FileAppender class.

5.Extract fileAppenderTXT , fileAppenderXML to a template method.

input:

1.Replace algorithm of TXTInput class with a simple and better one.

2.Extract TXTInput and XMLInput classes in a template method.

gui:

1.Refactor gui in a way that the user would be able to save a report in a name of his choice.

2.Rename InputWindow, SelectionWindow, ResultWindow class.

3.1.Improve conditional states of addReceiptButtonPressed().

3.2. Remove chains of method calls and reduce the size of SelectionWindowClass.

3.3. Remove if else statments from okButtonPressed().

output:

1.1. Rename Report, TxtReport, XMLReport classes

1.2.Extract algorithm of saveFile() in a template.

extensions:

1.1. Provide input to the application via an html file.

1.2.Store a report for a sales representative in an html file

REFACTORED DESIGN

USE CASES

insertFromHTML

Use case ID	UC1
Actors	The User.
Pre conditions	The user must upload a HTML file.
Main flow of events	1. The use case starts when the user upload the requested file. 2. The system displays the receipt data in the screen.
Alternative flow 1	1.The user uploads a wrong format file. 2.The system displays an error message.
Post conditions	-

insertFromTXT

Use case ID	UC2
Actors	The User.
Pre conditions	The user must upload a TXT file.
Main flow of events	1. The use case starts when the user upload the requested file. 2. The system displays the receipt data in the screen.
Alternative flow 1	1.The user uploads a wrong format file. 2.The system displays an error message.
Post conditions	-

insertFromXML

Use case ID	UC3
Actors	The User.
Pre conditions	The user must upload a XML file.
Main flow of events	1. The use case starts when the user upload the requested file. 2. The system displays the receipt data in the screen.
Alternative flow 1	1.The user uploads a wrong format file.

	2.The system displays an error message.
Post conditions	-

SelectAgent

Use case ID	UC4
Actors	The User.
Pre conditions	The user needs to press the representative of his choice on the screen.
Main flow of events	1. The use case starts when the user request a list of representatives to choose. 2. The system displays a list of representatives. 3. The user selects the representative of their choice. 4.The system displays a short report of the chosen represenive's data.
Post conditions	-

UpdateResults

Use case ID	UC5
Actors	The User.
Pre conditions	The user must have choose a representative.
Main flow of	1. The use case starts when the user demands the display of a general report for the representative of choice.

events	2. The system demands data clarifications from the user. 3. The user selects the prefered data such as: 3.1. Total sales value. 3.2. Sales for every product category. 3.3. Sales based on specific product of the user's choice. 3.4. The commission that he is entitled to based on the total sales.
Post conditions	-

CreateReport

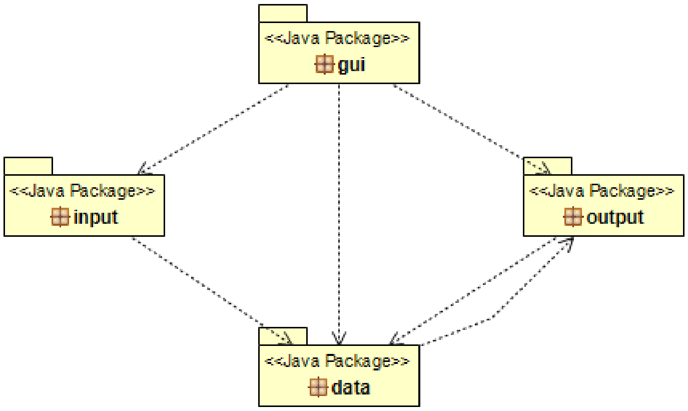
Use case ID	UC6
Actors	The User.
Pre conditions	The user must have choose a representative and must own a report full of the data of his choice.
Main flow of events	1. The use case starts when the user demands the storage of the general report. 2. The system demands the format clarfication. 3. The user selects the prefered format. 4. The system creates a report in the form of either TXT or XML.
Post conditions	

writeReceiptToFile

Use case ID	UC7
Actors	The User.
Pre conditions	The user must have choose a representative.
Main flow of events	1. The use case starts when the user request to add a receipt in one of the representatives. 2. The system adds the requested receipt.
Alternative flow of events	1.The user request to add a receipt in one of the representatives and the format of the info is wrong. 2.The systems throws an error in the screen.
Post Conditions	-

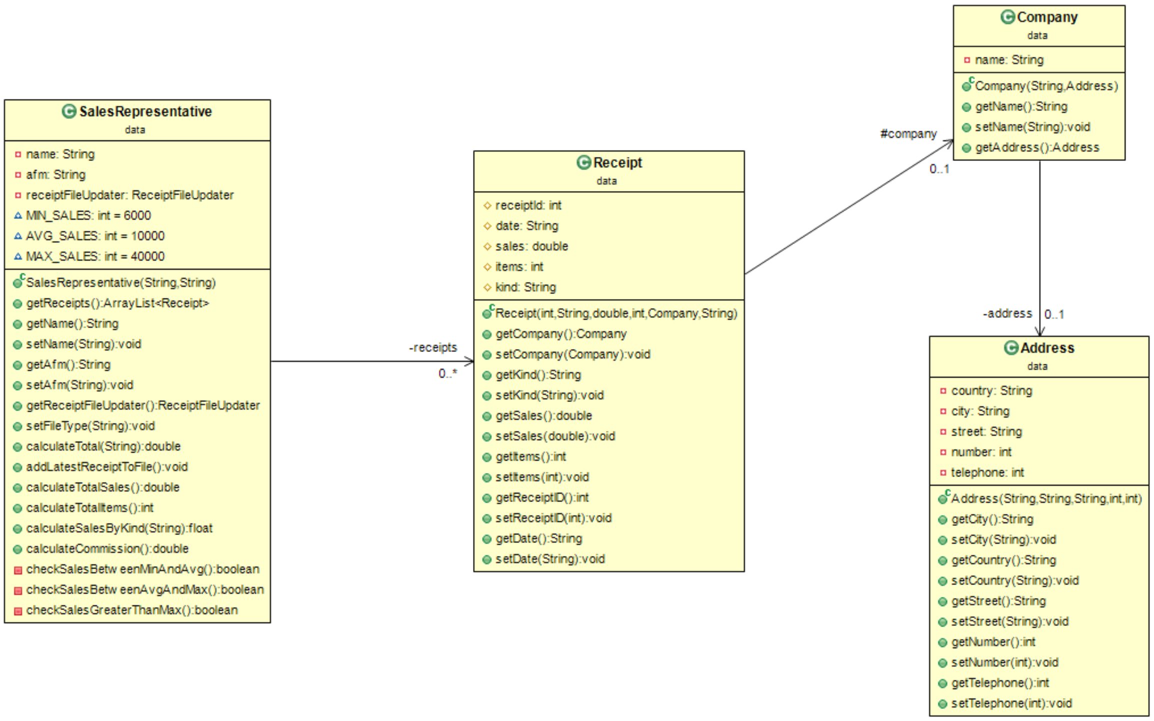
ARCHITECTURE

- **UML package diagram**

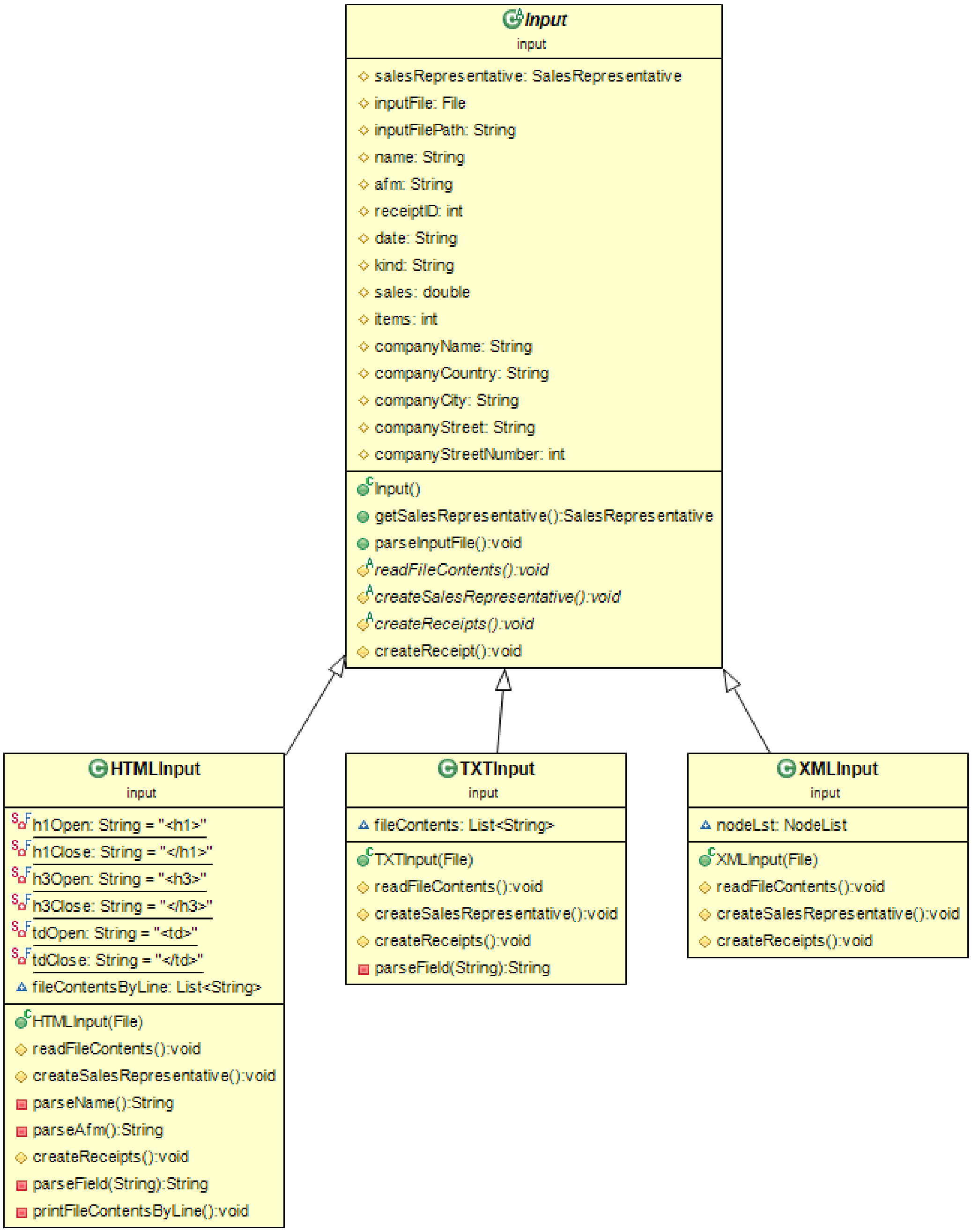


DETAILED DESIGN

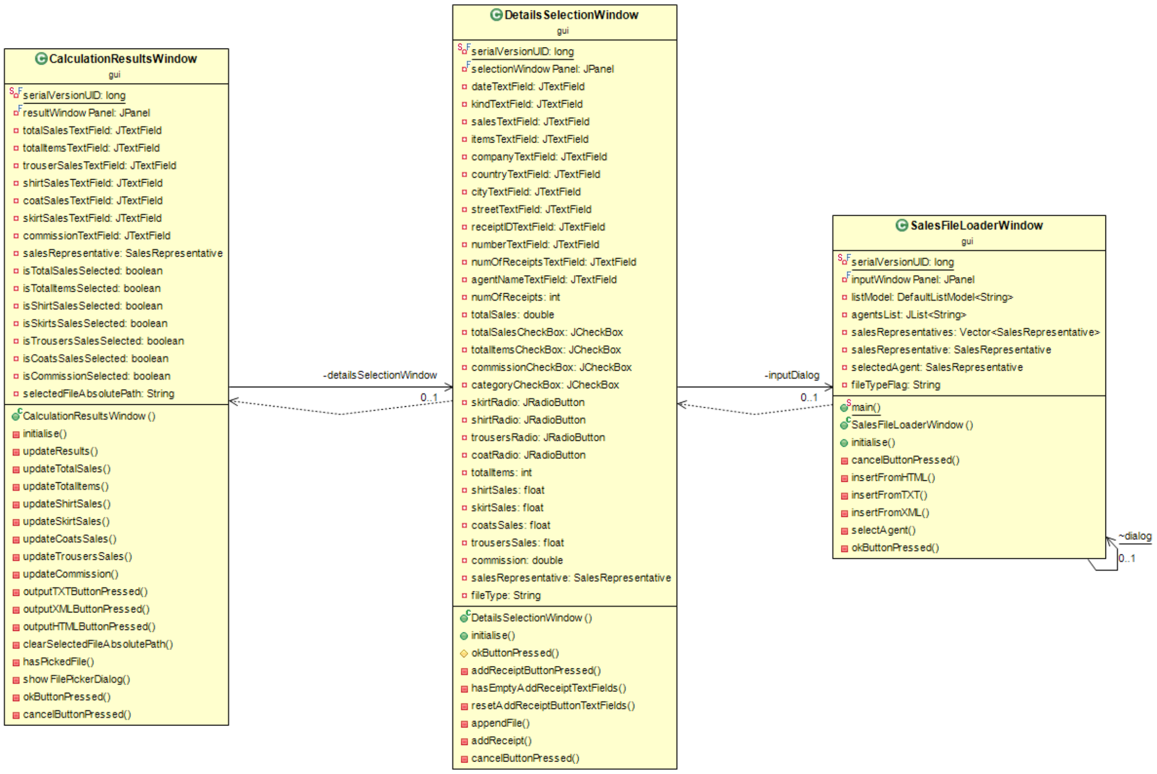
data:



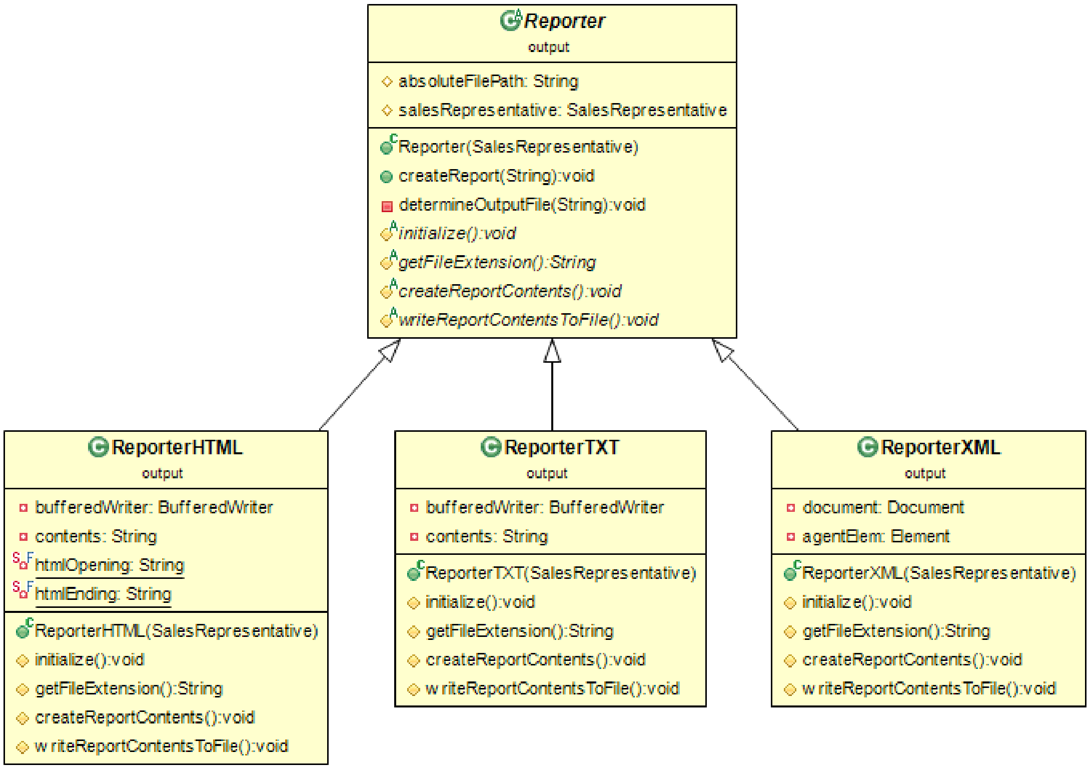
input:

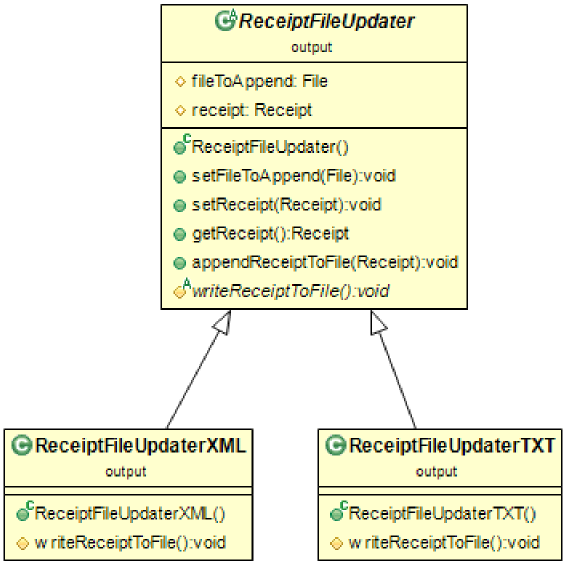

Input
input
salesRepresentative: SalesRepresentative
inputFile: File
inputFilePath: String
name: String
afm: String
receiptID: int
date: String
kind: String
sales: double
items: int
companyName: String
companyCountry: String
companyCity: String
companyStreet: String
companyStreetNumber: int
Input()
getSalesRepresentative():SalesRepresentative
parseInputFile():void
readFileContents():void
createSalesRepresentative():void
createReceipts():void
createReceipt():void
HTMLInput
input
h1Open: String = "<h1>"
h1Close: String = "</h1>"
h3Open: String = "<h3>"
h3Close: String = "</h3>"
tdOpen: String = "<td>"
tdClose: String = "</td>"
fileContentsByLine: List<String>
HTMLInput(File)
readFileContents():void
createSalesRepresentative():void
parseName():String
parseAfm():String
createReceipts():void
parseField(String):String
printFileContentsByLine():void
TXTInput
input
fileContents: List<String>
TXTInput(File)
readFileContents():void
createSalesRepresentative():void
createReceipts():void
parseField(String):String
XMLInput
input
nodeLst: NodeList
XMLInput(File)
readFileContents():void
createSalesRepresentative():void
createReceipts():void

gui:


CalculationResultsWindow
gui
serialVersionUID: long
resultWindowPanel: JPanel
totalSalesTextField: JTextField
totalItemsTextField: JTextField
trouserSalesTextField: JTextField
shirtSalesTextField: JTextField
coatSalesTextField: JTextField
skirtSalesTextField: JTextField
commissionTextField: JTextField
salesRepresentative: SalesRepresentative
isTotalSalesSelected: boolean
isTotalItemsSelected: boolean
isShirtSalesSelected: boolean
isSkirtsSalesSelected: boolean
isTrousersSalesSelected: boolean
isCoatsSalesSelected: boolean
isCommissionSelected: boolean
selectedFileAbsolutePath: String
CalculationResultsWindow()
initialise()
updateResults()
updateTotalSales()
updateTotalItems()
updateShirtSales()
updateSkirtSales()
updateCoatsSales()
updateTrousersSales()
updateCommission()
outputTXTButtonPressed()
outputXMLButtonPressed()
outputHTMLButtonPressed()
clearSelectedFileAbsolutePath()
hasPickedFile()
showFilePickerDialog()
okButtonPressed()
cancelButtonPressed()
-detailsSelectionWindow
0..1
DetailsSelectionWindow
gui
serialVersionUID: long
selectionWindowPanel: JPanel
dateTextField: JTextField
kindTextField: JTextField
salesTextField: JTextField
itemsTextField: JTextField
companyTextField: JTextField
countryTextField: JTextField
cityTextField: JTextField
streetTextField: JTextField
receiptIDTextField: JTextField
numberTextField: JTextField
numOfReceiptsTextField: JTextField
agentNameTextField: JTextField
numOfReceipts: int
totalSales: double
totalSalesCheckBox: JCheckBox
totalItemsCheckBox: JCheckBox
commissionCheckBox: JCheckBox
categoryCheckBox: JCheckBox
skirtRadio: JRadioButton
shirtRadio: JRadioButton
trousersRadio: JRadioButton
coatRadio: JRadioButton
totalItems: int
shirtSales: float
skirtSales: float
coatsSales: float
trousersSales: float
commission: double
salesRepresentative: SalesRepresentative
fileType: String
DetailsSelectionWindow()
initialise()
okButtonPressed()
addReceiptButtonPressed()
hasEmptyAddReceiptTextFields()
resetAddReceiptButtonTextFields()
appendFile()
addReceipt()
cancelButtonPressed()
-inputDialog
0..1
SalesFileLoaderWindow
gui
serialVersionUID: long
inputWindowPanel: JPanel
listModel: DefaultListModel<String>
agentsList: JList<String>
salesRepresentatives: Vector<SalesRepresentative>
salesRepresentative: SalesRepresentative
selectedAgent: SalesRepresentative
fileTypeFlag: String
main()
SalesFileLoaderWindow()
initialise()
cancelButtonPressed()
insertFromHTML()
insertFromTXT()
insertFromXML()
selectAgent()
okButtonPressed()
~dialog
0..1

output:


Reporter
output
absoluteFilePath: String
salesRepresentative: SalesRepresentative
Reporter(SalesRepresentative)
createReport(String):void
determineOutputFile(String):void
initialize():void
getFileExtension():String
createReportContents():void
writeReportContentsToFile():void
ReporterHTML
output
bufferedWriter: BufferedWriter
contents: String
htmlOpening: String
htmlEnding: String
ReporterHTML(SalesRepresentative)
initialize():void
getFileExtension():String
createReportContents():void
writeReportContentsToFile():void
ReporterTXT
output
bufferedWriter: BufferedWriter
contents: String
ReporterTXT(SalesRepresentative)
initialize():void
getFileExtension():String
createReportContents():void
writeReportContentsToFile():void
ReporterXML
output
document: Document
agentElem: Element
ReporterXML(SalesRepresentative)
initialize():void
getFileExtension():String
createReportContents():void
writeReportContentsToFile():void



CLASSES RESPONSIBILITIES AND COLLABORATIONS (CRC CARDS)

data:

Class Name: Address	
Responsibilities	**Collaborations**
• Address class is resposible for: • Initializing **country, city, street, number, telephone fields.** • Making **getters** and **setters.**	• **Company()** class in data package • **DetailsSelectionWindow()** class in gui package. • **Input()** class in Input package.

Class Name: Company	
Responsibilities	**Collaborations**
• Company class is resposible for: • Initializing **name, address fields.** • Making **getters** and **setters.**	• **Company()** class in data package • **DetailsSelectionWindow()** class in gui package. • **Input()** class in Input package.

Class Name: Receipt

Responsibilities	**Collaborations**
- Address class is resposible for: - Initializing **receiptId,date, sales, items, company, kind fields.** - Making **getters** and **setters.**	- **SalesRepresentative()** class in data package - **DetailsSelectionWindow()** class in gui package. - **Input()** class in Input package. - **ReceiptFileUpdater()** class in Output package.

Class Name: SalesRepresentative	
Responsibilities	**Collaborations**
- Address class is resposible for: - Initializing **name,afm fields.** - Making **getters** and **setters.** - Having methods that calculate **comission,sales by kind, total items, total sales, adds latest receipt to file.**	- **CalculationResultWindow()** class in gui package. - **DetailSelectionWindow()** class in gui package. - **SalesFileLoaderWindow()** class in gui package. - **Input()** class in input package. - **TXTInput()** class in input package. - **XMLInput()** class in input package.

	• **HTMLInput()** class in input package. • **Reporter()** class in output package. • **ReporterTXT()** class in output package. • **ReporterXML()** class in output package. • **Reporter()HTML** class in output package.

gui:

Class Name: SalesFileLoaderWindow	
Responsibilities	**Collaborations**
• SalesFileLoaderWindow class is resposible for: • The window that appears when the project runs. • The buttons that appear in the screen and their operations.	• **DetailsSelectionWindow()** class in gui package. • **SalesFileLoaderWindow()** class in gui package.

Class Name: DetailSelectionWindow	
Responsibilities	**Collaborations**
• DetailSelectionWindow class is resposible for: • The window that appears when the user **presses the insert button** of any format. • The buttons that appear in the screen and their operations.	• **CalculationResultWindow** class in gui package. • **DetailsSelectionWindow()** class in gui package. • **SalesFileLoaderWindow()** class in gui package.

Class Name: CalculationResultWindow	
Responsibilities	**Collaborations**
• CalculationResultWindow class is resposible for: • The window that appears when the users selects a representative and he is requested to select the specific details about him**.** • The buttons that appear in the screen and their operations.	• **CalculationResultWindow** class in gui package. • **DetailsSelectionWindow()** class in gui package.

input:

Class Name: Input	
Responsibilities	**Collaborations**
• Input classis the basic class for a template pattern of HTMLInput, TXTInput, XMLInput is resposible for: • Making methods for the **common algorithm** of the template classes. • Making methods for the parts of the code that **are not common** in template classes.	• **TXTInput()** class in input package. • **XMLInput()** class in input package. • **HTMLInput()** class in input package.

Class Name: HTMLInput	
Responsibilities	**Collaborations**
• HTMLInput class is class that extends Input and its resposible for: • **Parsing data** from a html file and show the details of the representative on the screen.	• **SalesFileLoaderWindow()** class in gui package.

Class Name: TXTInput	
Responsibilities	**Collaborations**
• TXTInput class is class that extends Input and its resposible for: • **Parsing data** from a txt file and show the details of the representative on the screen.	• **SalesFileLoaderWindow()** class in gui package.

Class Name: XMLInput	
Responsibilities	**Collaborations**
• XMLInput class is class that extends Input and its resposible for: • **Parsing data** from a xml file and show the details of the representative on the screen.	• **SalesFileLoaderWindow()** class in gui package.

output:

Class Name: ReceiptFileUpdater

Responsibilities	**Collaborations**
- ReceiptFileUpdater class is the basic class for a template pattern of ReceiptFileUpaterTXT, ReceiptFileUpaterXML resposible for: - Writing the **Receipt contents** to the file.	- **SalesRepresentative()** class in data package.

Class Name: ReceiptFileUpdaterTXT	
Responsibilities	**Collaborations**
- ReceiptFileUpaterTXT class is class that extends ReceiptFileUpater and its resposible for: - Writing the **Receipt contents** to the TXT file.	- **SalesFileLoaderWindow()** class in gui package.

Class Name: ReceiptFileUpdaterXML	
Responsibilities	**Collaborations**
- ReceiptFileUpaterXML class is class that extends ReceiptFileUpater and its resposible for: - Writing the **Receipt contents**	- **SalesFileLoaderWindow()** class in gui package.

to the XML file.	

Class Name: Reporter	
Responsibilities	**Collaborations**
• Reporter class is the basic class for a template pattern of HTMLInput, TXTInput, XMLInput is resposible for: • Making methods for the **common algorithm** of the template classes. • Making methods for the parts of the code that **are not common** in template classes.	• **ReporterTXT()** class in output package. • **ReporterXML()** class in output package. • **ReporterInput()** class in output package.

Class Name: ReporterHTML	
Responsibilities	**Collaborations**
• ReporterHTML class is class that extends Reporter and its resposible for: • Exprorting the requested data to an html file report .	• **CalculationResultWindow** class in gui package.

Class Name: ReporterTXT	
Responsibilities	**Collaborations**
• ReporterTXT class is class that extends Reporter and its resposible for: • Exprorting the requested data to an txt file report.	• **CalculationResultWindow** class in gui package.

Class Name: ReporterXML	
Responsibilities	**Collaborations**
• ReporterXML class is class that extends Reporter and its resposible for: • Exprorting the requested data to an xml file report.	• **CalculationResultWindow** class in gui package.

Note : **Overall, many names of variables, methods and classes have been changed in the project.Many new methods that helped the code get more clear have been added and many getters and setters have been erased by parsing the requested data through objects.**

We also added some test inside the project for the Input and Reporter.