REFACTORING CODE OF SALES COMMISSIONS APPLICATION

OVERALL REPORT

TABLE OF CONTENTS

INTRODUCTION

The goal of this project was to refactor an existing app to be more readable and more extendable. Multiple classes were rewritten and architecture was rethough. The app was also made to support HTML loading/reporting.

REFACTORED DESIGN

USE CASES

Use case ID	LoadFile
Actors	The user
Pre conditions	None
Main flow of events	1. The use case starts when the user picks the option to load a file from the system, the file can be a. TXT b. XML c. HTML 2. The file gets parsed 3. A list of the names of the representatives appears on the right hand side of the window
Alternative flow 1	1. If the system cannot parse the selected file an error message is displayed
Post conditions	The system has the data of the file loaded in memory

Use case ID	SelectRepresentative
Actors	The user
Pre conditions	A file has been loaded
Main flow of events	1. The use case starts when the user picks a representative from the list 2. The user presses the OK button
Post conditions	None

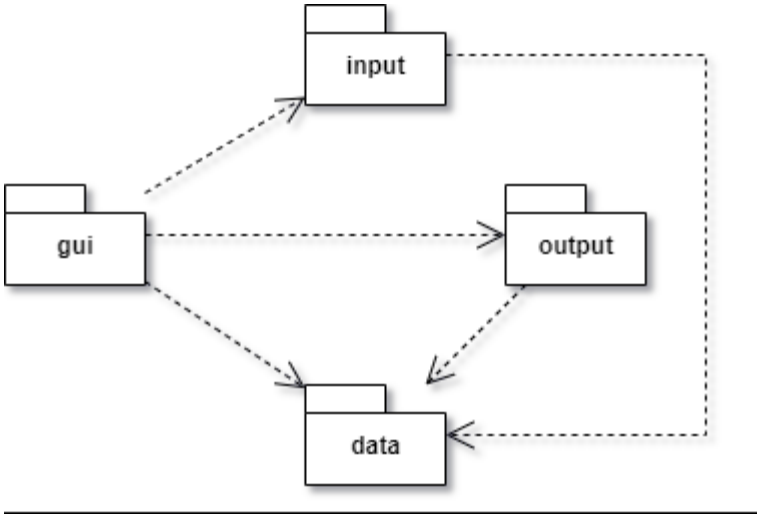
Use case ID	addReceipt
Actors	The user
Pre conditions	None
Main flow of events	1. The use case starts when the user presses the “Add New Receipt” button. 2. A list of text boxes the user is required to complete is shown. 3. The user fills in the info he wants 4. User presses add button
Alt. Flow 1	If the user clicks the “Add New Receipt” button again the fields disappear
Post conditions	The counter of added receipts increments by 1.

Use case ID	renewDetails
Actors	Time
Pre conditions	User adds a new Receipt
Main flow of events	1. The use case starts when addReceipt UC has ended. 2. The system adds the info the user has previously written in a new receipt and appends it to the loaded file.
Post conditions	The loaded file has been changed.

Use case ID	ShowRepresentativeStats
Actors	The user
Pre conditions	A file has been loaded and the user has picked a representative from the list
Main flow of events	1. The use case starts when the user starts selecting which stats to show 2. The user presses OK
Post conditions	The window the user was on closes and a new one with the stats opens

Use case ID	ReportToFile
Actors	The user
Pre conditions	A file has been loaded and the user has picked a representative from the list and he has chosen the stats he wants to see
Main flow of events	1. The use case starts when the user presses one of the report buttons the options being a. Report to TXT b. Report to XML c. Report to HTML 2. The user picks where he wants the file to be saved
Post conditions	A file with the name <AFM>_SALES has been created at the user's desired location

ARCHITECTURE



DETAILED DESIGN

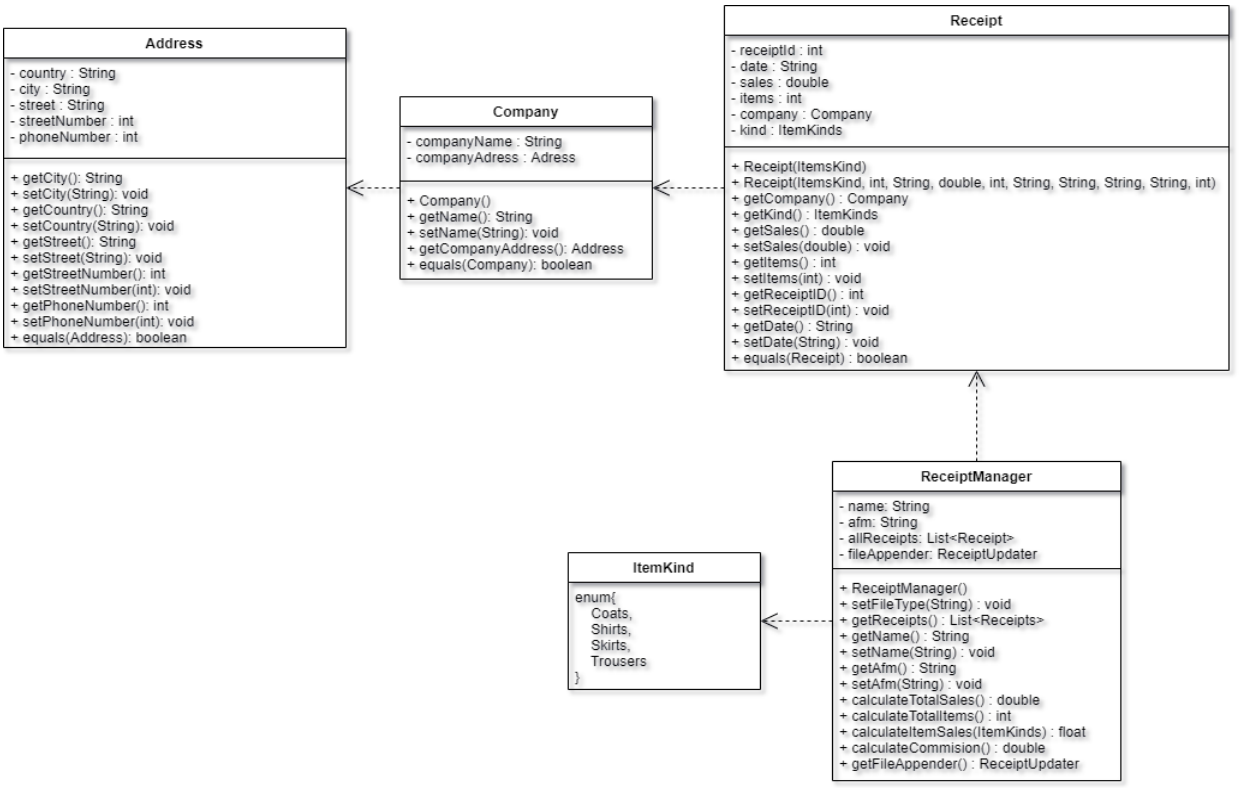


Figure 1:data Package

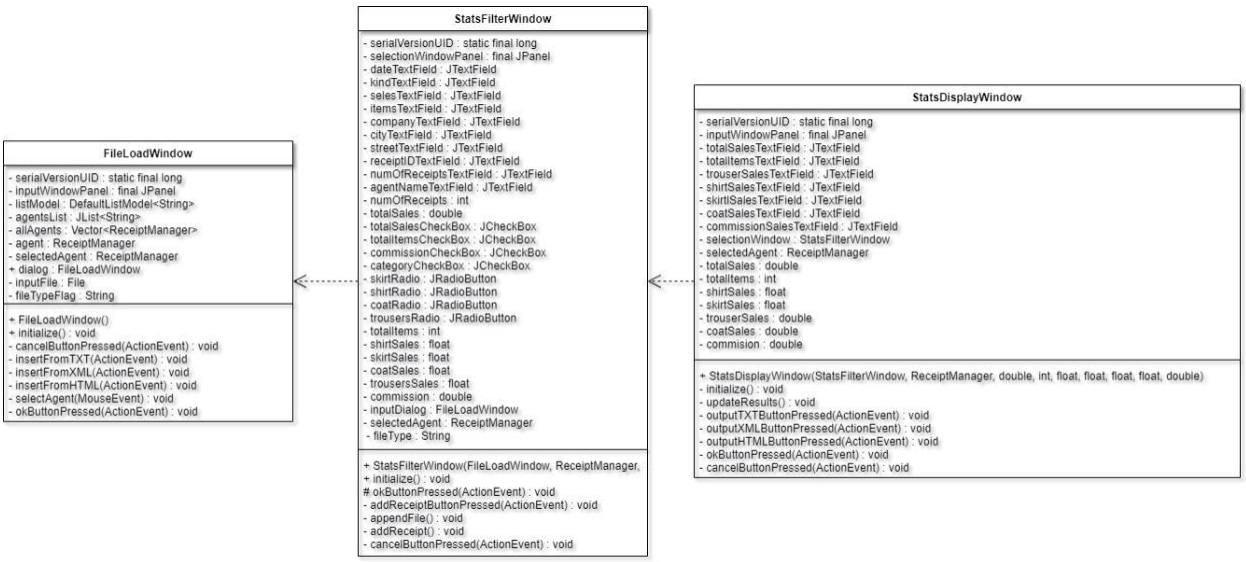


Figure 2:gui Package

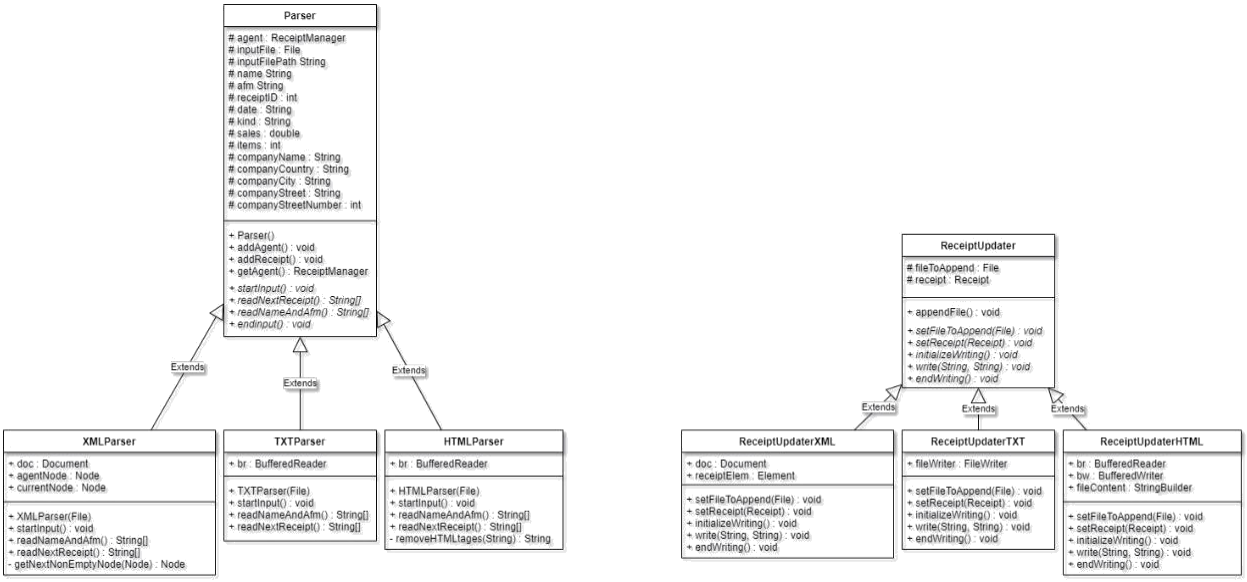


Figure 3:input Package

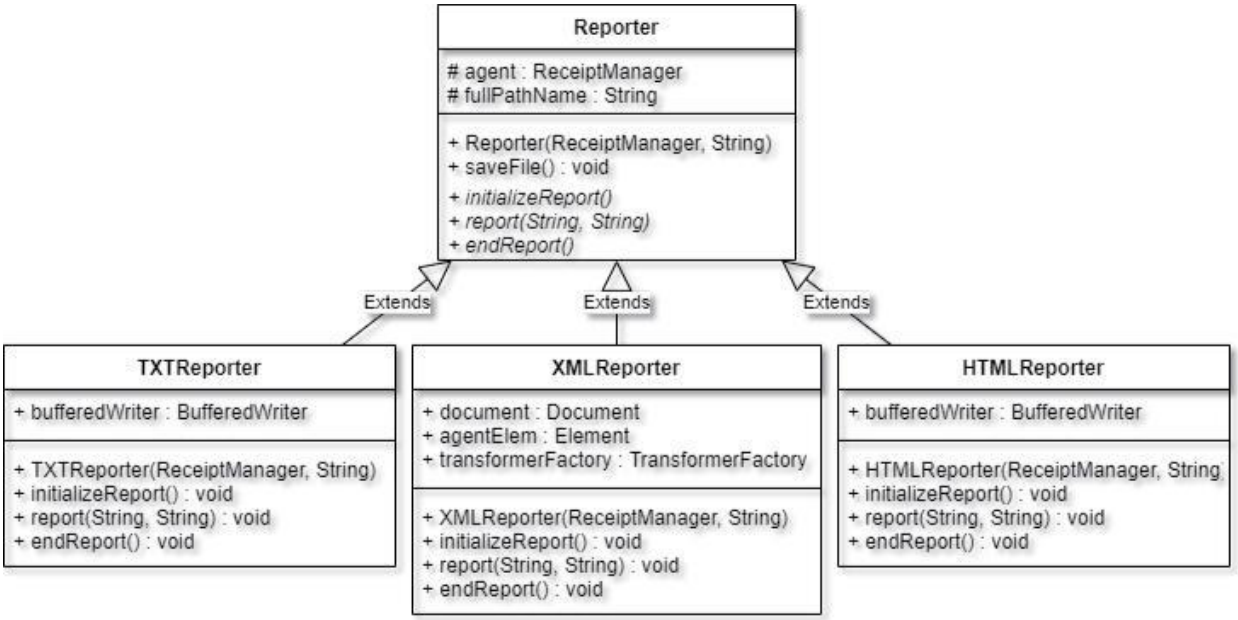


Figure 4:output Package

A common problem is the old design was functions that implemented the same algorithm not being constructed properly. To fix this we implemented the common steps of the algorithm in the mother class and each class that extended it implemented only the file specific steps of that algorithm. Another common problem was bad naming of classes which we fixed. There were other small problems like using too many primitives instead of the defined object classes and classes being in the wrong package which were also fixed.

CLASSES RESPONSIBILITIES AND COLLABORATIONS (CRC CARDS)

Class Name: Address	
Responsibilities	**Collaborations**
This class is a class that collects the primitive data types of an address into a single class It provides no functionality apart from storage	Is used by Company

Class Name: Company	
Responsibilities	**Collaborations**
This class is a class that collects the primitive and non primitive data types of a company into a single class It provides no functionality apart from storage	Is used by Receipt

Class Name: Receipt	
Responsibilities	**Collaborations**
This class is a class that collects the primitive and non primitive data types of a receipt into a single class It provides no functionality apart from storage	Is used by ReceiptManager

Class Name: ItemKinds	
Responsibilities	**Collaborations**
A sum type class to hold the kinds of items that can be contained in a receipt	Is used by all classes

Class Name: ReceiptManager	
Responsibilities	**Collaborations**
This class responsibility is to handle receipt objects, it calculates stats and info bases on the receipts and hold the ReceiptUpdater class responsible for the updating	Is used by all classes

Class Name: FileLoadWindow	
Responsibilities	**Collaborations**
This class responsibility is to present the user with a window to load files from the file system and provide a list of representatives of the receipts that have been loaded to see their stats	Is used by StatsFilterWindow

Class Name: StatsFilterWindow	
Responsibilities	**Collaborations**
This class responsibility is to present the user with a window to select what stats he wants to see about the selected representative and add receipts	Is used by StatsDisplayWindow and FileLoadWindow

Class Name: StatsDisplayWindow	
Responsibilities	**Collaborations**
This class responsibility is to present the user with a window that show the stats that he selected in the StatsFilterWindow and also provide him with the option to export these stats to a file	Is used by StatsDisplayWindow

Class Name: Parser	
Responsibilities	**Collaborations**
This class responsibility is provide the basic algorithmic steps for loading a file as well as the functionality to add a receipt from the file loaded	Is extended by HTMLParser, TXTParser, XMLParser

Class Name: HTMLParser, TXTParser, XMLParser	
Responsibilities	**Collaborations**
These classes' responsibility is to load a file by providing the field data to Parser. Each class handles a different file type	Extend Parser, are used by FileLoadWindow

Class Name: ReceiptUpdater	
Responsibilities	**Collaborations**
This class responsibility is provide the basic algorithmic steps for appending a receipt to the loaded file	Is extended by ReceiptUpdaterTXT, ReceiptUpdaterXML, ReceiptUpdaterHTML

Class Name: ReceiptUpdaterTXT, ReceiptUpdaterXML, ReceiptUpdaterHTML	
Responsibilities	**Collaborations**
These classes' responsibility is to append receipts to the loaded file. Each class handles a different file type	Are used by StatsFilterWindow

Class Name: Reporter	
Responsibilities	**Collaborations**
This class responsibility is to write the stats the user has selected to a selected file	Is extended by TXTReporter, XMLReporter, HTMLReporter

Class Name: TXTReporter, XMLReporter, HTMLReporter	
Responsibilities	**Collaborations**
These classes' responsibility is to report to a file. Each class handles a different file type	Are used by StatsDisplayWindow