REFACTORING TAX INCOME CALCULATOR

OVERALL REPORT

VERSION <1.0>

Σύρπας Βασίλειος AM:4174
Μπουλώτης Παναγιώτης AM:4271

TABLE OF CONTENTS

INTRODUCTION

The main objective of this project was to refactor the legacy application and improve the GUI implementation, for a more user-friendly approach as well.

REFACTORED DESIGN

USE CASES

LoadTaxpayer

Use case ID	UC1
Actors	Taxpayer handler
Preconditions	A <tax registration number >_INFO.txt or <tax registration number >_INFO.xml file with the correct information must exist.
Main flow of events	1. The use case starts when the handler presses the button “Load Taxpayer” 2. The handler selects to load the information from a .txt or a .xml file by choosing the corresponding radio button. 3. The “File explorer” button is pressed 3.1. The handler browses their directories to select the INFO file. 3.2. The message “Opening: <file>” is displayed 3.3. The handler presses the “OK” button.
Alternative flow 1	The handler can select the type of INFO file they want to load and only type the tax registration number. After that, they can press OK to load the taxpayer.
Alternative flow 2	If the handler tries to load a tax registration number that is already loaded, a warning message is displayed.
Alternative flow 3	If the handler chooses a tax registration number that does not have 9 digits, a warning message is displayed.
Alternative flow 4	The handler presses the “Cancel” button and does not load a taxpayer.
Post conditions	The corresponding taxpayer has been loaded and their tax registration number is displayed on the taxpayer list.

SelectTaxpayer

Use case ID	UC2
Actors	Taxpayer handler
Preconditions	At least one taxpayer must be loaded on the list.
Main flow of events	1. The use case starts when the handler double clicks the tax registration number of the taxpayer they want to select
Alternative flow	If the list is empty and the handler clicks on it, a warning message is displayed.
Post conditions	The window with the taxpayer's information and available actions is displayed.

DeleteTaxpayer

Use case ID	UC3
Actors	Taxpayer handler
Preconditions	At least one taxpayer must be loaded on the list.
Main flow of events	1. The use case starts when the handler clicks the tax registration number of the taxpayer they want to delete 2. The handler presses the "Delete Taxpayer" button.
Alternative flow	-
Post conditions	The taxpayer is deleted, and the taxpayer's registration number is removed from the taxpayer list.

AddReceipt

Use case ID	UC4
Actors	Taxpayer handler
Preconditions	The taxpayer’s tax registration number must be selected from the list.
Main flow of events	1. The use case starts when the handler presses the “Add Receipt” button. 2. The program displays a new window with the information fields of a new receipt. 3. After completing the fields, the handler pressed the “OK” button.
Alternative flow 1	If the handler presses cancel when the new window is displayed, the window closes, and no new receipt is saved.
Alternative flow 2	If a field is not typed correctly according to each field format, a warning message is displayed.
Post conditions	The receipt is saved, and the receipt list adds its receipt Id.

EditReceipt

Use case ID	UC5
Actors	Taxpayer handler
Preconditions	The taxpayer's tax registration number must be selected from the list and at least one receipt must be added.
Main flow of events	1. The use case starts when the handler clicks on a receipt on the list 2. The handler presses the "Edit Receipt" button. 3. The program displays a new window with the information fields of the receipt. 4. After completing the fields, the handler pressed the "OK" button.
Alternative flow 1	If the handler presses cancel when the new window is displayed, the window closes, and the receipt is not edited.
Alternative flow 2	If a field is not typed correctly according to each field format, a warning message is displayed.
Alternative flow 3	If the receipt list is empty and the handler clicks on it, a warning message is displayed.
Post conditions	The receipt is edited.

DeleteReceipt

Use case ID	UC6
Actors	Taxpayer handler
Preconditions	The taxpayer's tax registration number must be selected from the list and at least one receipt must be added.
Main flow of events	1. The use case starts when the handler clicks on a receipt on the list 2. The handler presses the "Delete Receipt" button.
Alternative flow	If the receipt list is empty and the handler clicks on it, a warning message is displayed.
Post conditions	The receipt is deleted, and the receipt list removes its receipt Id.

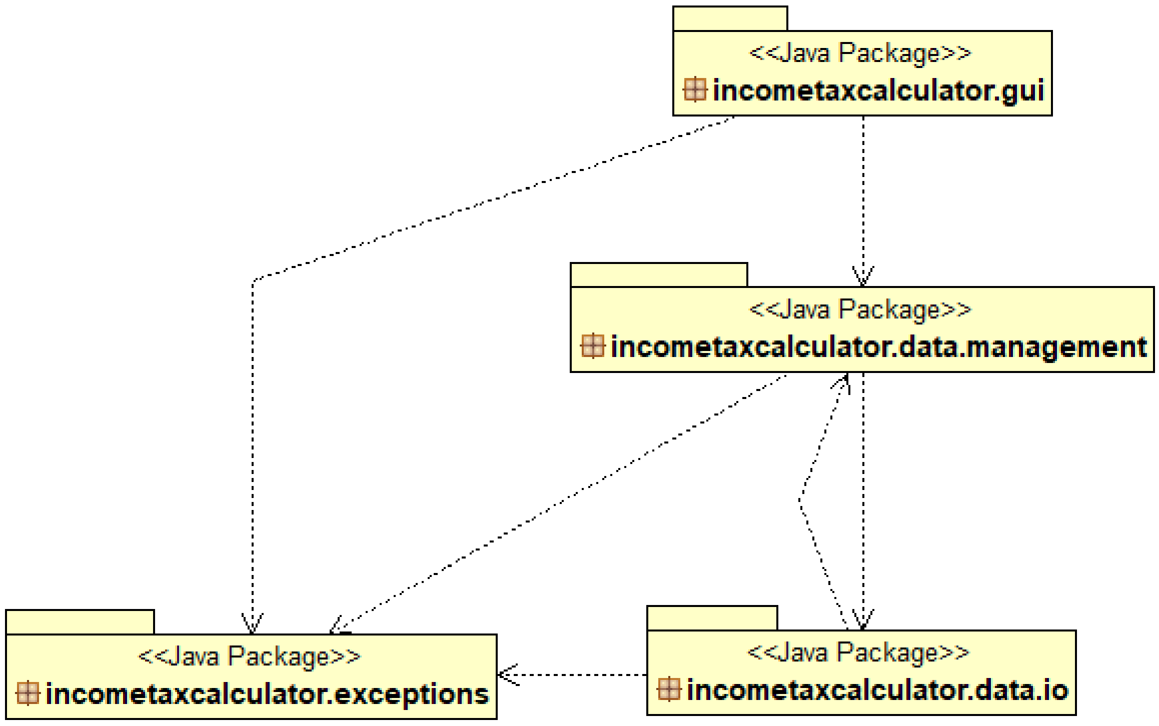
ViewReport

Use case ID	UC7
Actors	Taxpayer handler
Preconditions	The taxpayer's tax registration number must be selected from the list.
Main flow of events	1. The use case starts when the handler presses the "View Report" button.
Alternative flow	-
Post conditions	Two new windows are displayed. One with a pie-chart of the percentages of receipts and their kinds. The other about the total tax and tax increase/decrease of the taxpayer's income.

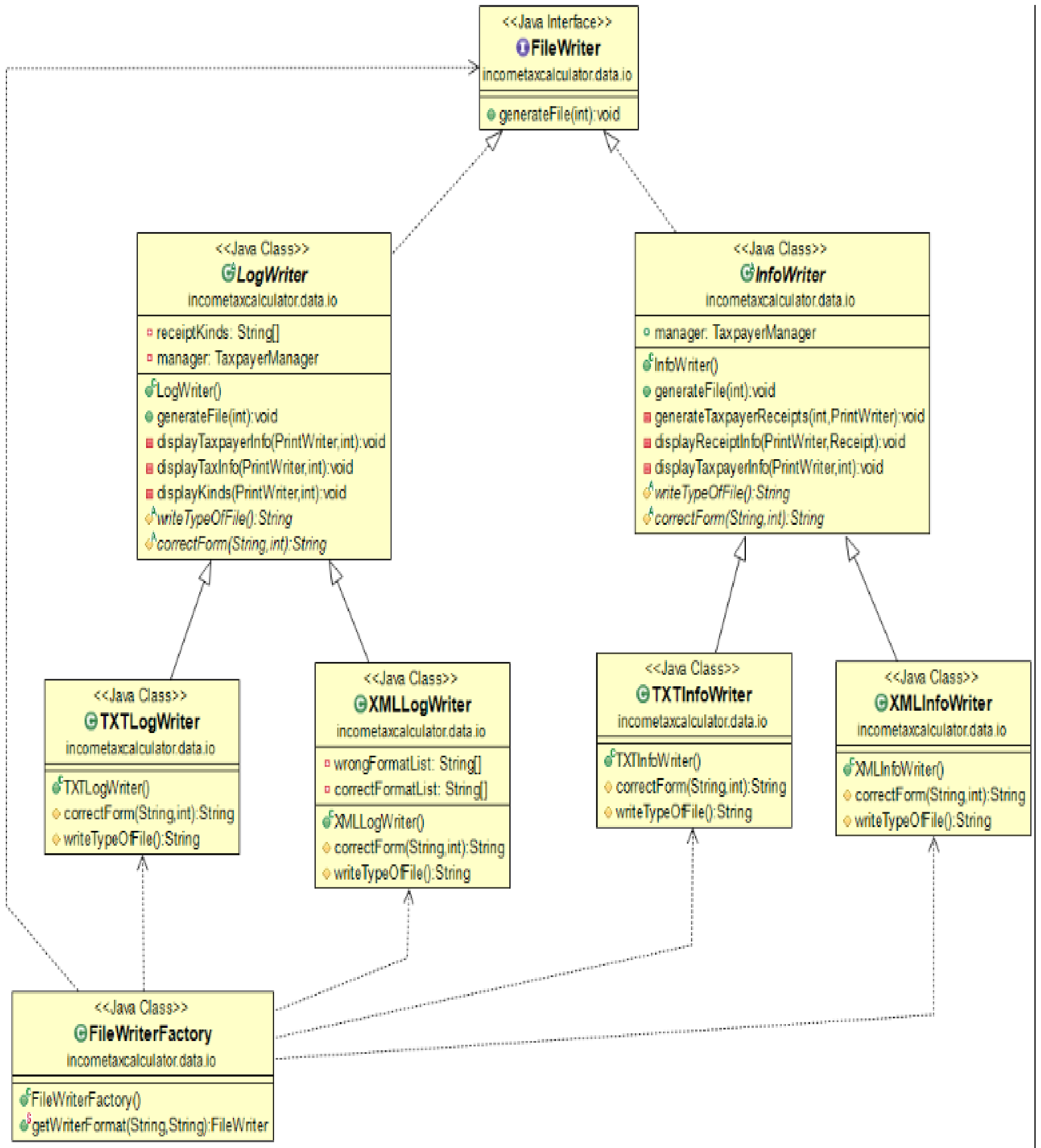
SaveData

Use case ID	UC8
Actors	Taxpayer handler
Preconditions	The taxpayer's tax registration number must be selected from the list.
Main flow of events	1. The use case starts when the handler presses the "Save Data" button. 2. The handler selects the type that the LOG file is saved to. 3. The handler presses the "OK" button
Alternative flow	The handler presses "Cancel", the window closes and no new file is saved.
Post conditions	A new <tax registration number>_LOG.txt or <tax registration number>_LOG.xml file is saved.

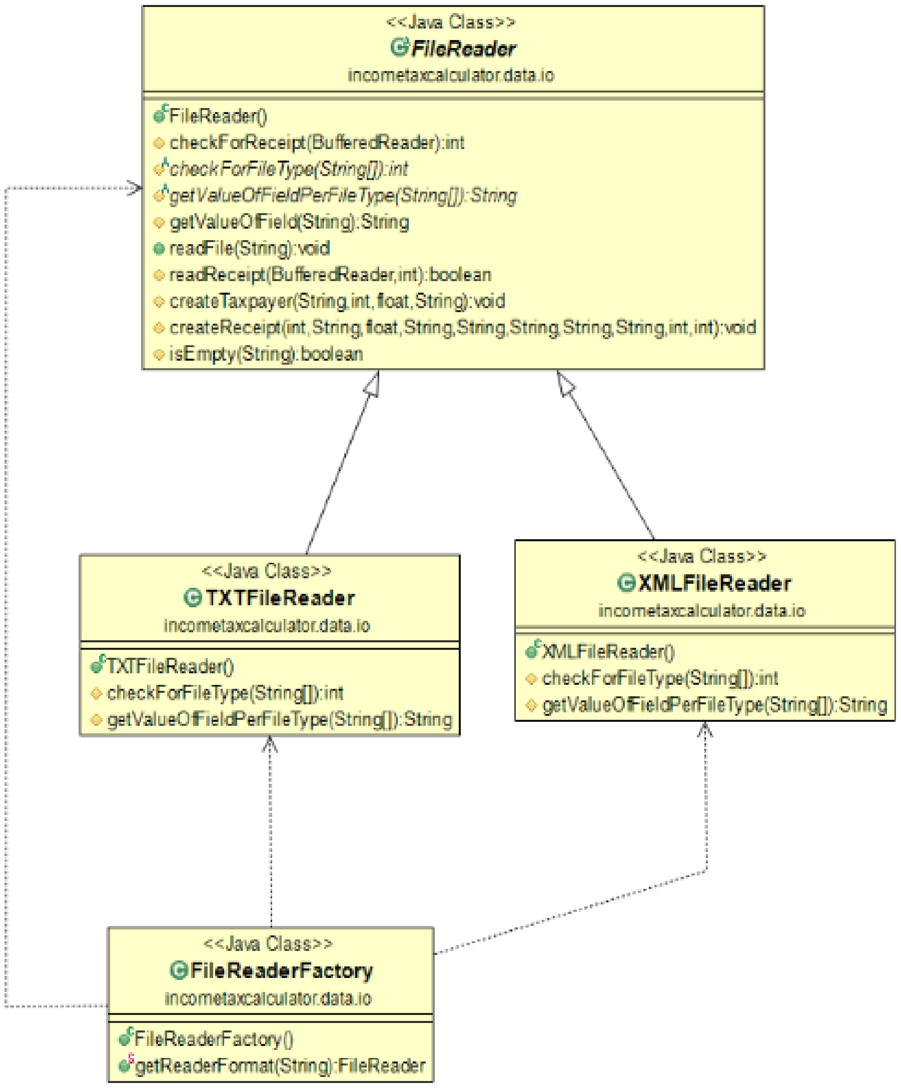
ARCHITECTURE


<<Java Package>>
incometaxcalculator.gui
<<Java Package>>
incometaxcalculator.data.management
<<Java Package>>
incometaxcalculator.exceptions
<<Java Package>>
incometaxcalculator.data.io

- Package data.io

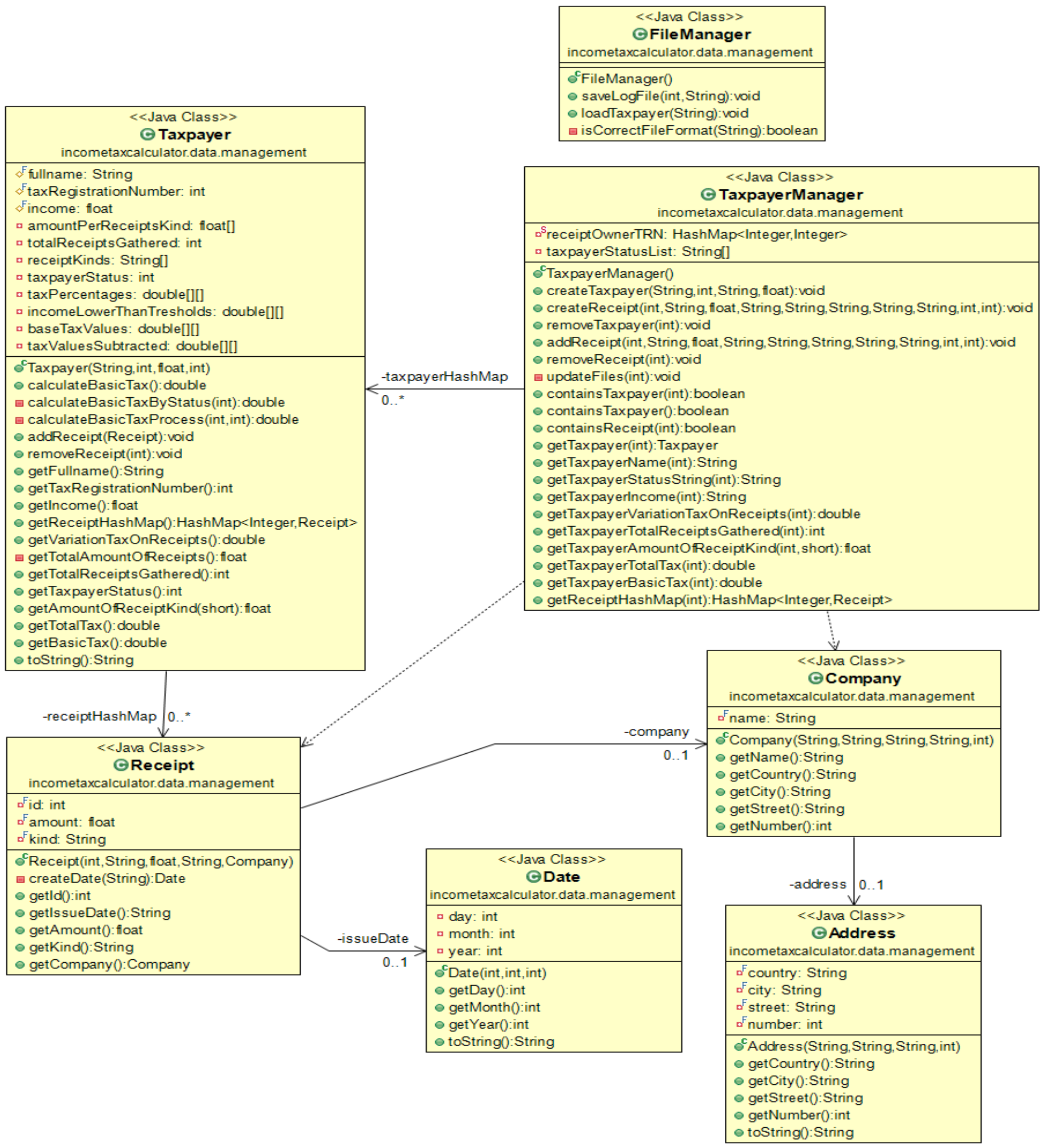


<<Java Interface>>
FileWriter
incometaxcalculator.data.io
generateFile(int):void
<<Java Class>>
LogWriter
incometaxcalculator.data.io
receiptKinds: String[]
manager: TaxpayerManager
LogWriter()
generateFile(int):void
displayTaxpayerInfo(PrintWriter,int):void
displayTaxInfo(PrintWriter,int):void
displayKinds(PrintWriter,int):void
writeTypeOfFile():String
correctForm(String,int):String
<<Java Class>>
InfoWriter
incometaxcalculator.data.io
manager: TaxpayerManager
InfoWriter()
generateFile(int):void
generateTaxpayerReceipts(int,PrintWriter):void
displayReceiptInfo(PrintWriter,Receipt):void
displayTaxpayerInfo(PrintWriter,int):void
writeTypeOfFile():String
correctForm(String,int):String
<<Java Class>>
TXTLogWriter
incometaxcalculator.data.io
TXTLogWriter()
correctForm(String,int):String
writeTypeOfFile():String
<<Java Class>>
XMLLogWriter
incometaxcalculator.data.io
wrongFormatList: String[]
correctFormatList: String[]
XMLLogWriter()
correctForm(String,int):String
writeTypeOfFile():String
<<Java Class>>
TXTInfoWriter
incometaxcalculator.data.io
TXTInfoWriter()
correctForm(String,int):String
writeTypeOfFile():String
<<Java Class>>
XMLInfoWriter
incometaxcalculator.data.io
XMLInfoWriter()
correctForm(String,int):String
writeTypeOfFile():String
<<Java Class>>
FileWriterFactory
incometaxcalculator.data.io
FileWriterFactory()
getWriterFormat(String,String):FileWriter



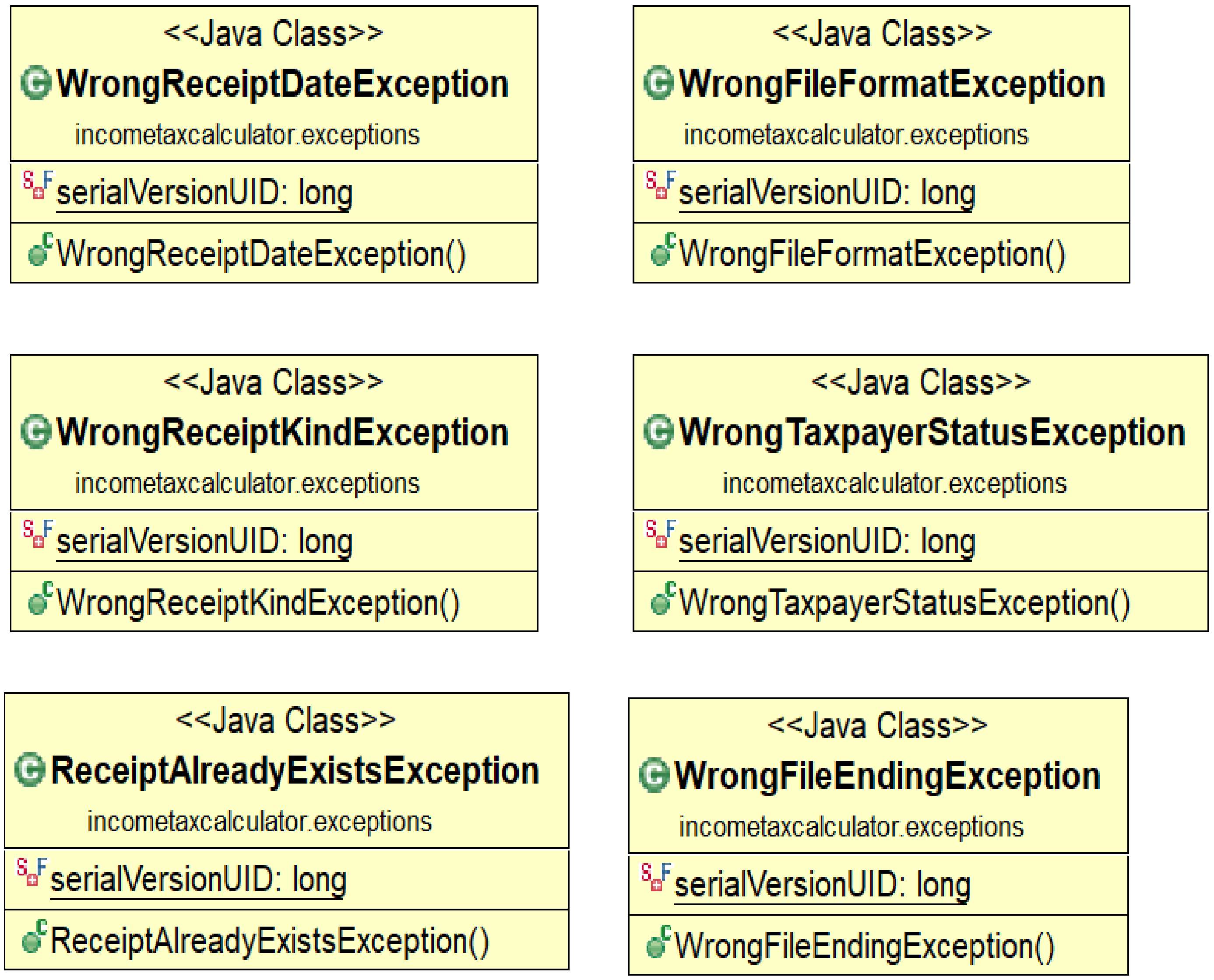
<<Java Class>>
FileReader
incometaxcalculator.data.io
FileReader()
checkForReceipt(BufferedReader):int
checkForFileType(String[]):int
getValueOfFieldPerFileType(String[]):String
getValueOfField(String):String
readFile(String):void
readReceipt(BufferedReader,int):boolean
createTaxpayer(String,int,float,String):void
createReceipt(int,String,float,String,String,String,String,String,int,int):void
isEmpty(String):boolean
<<Java Class>>
TXTFileReader
incometaxcalculator.data.io
TXTFileReader()
checkForFileType(String[]):int
getValueOfFieldPerFileType(String[]):String
<<Java Class>>
XMLFileReader
incometaxcalculator.data.io
XMLFileReader()
checkForFileType(String[]):int
getValueOfFieldPerFileType(String[]):String
<<Java Class>>
FileReaderFactory
incometaxcalculator.data.io
FileReaderFactory()
getReaderFormat(String):FileReader

- Package data.management

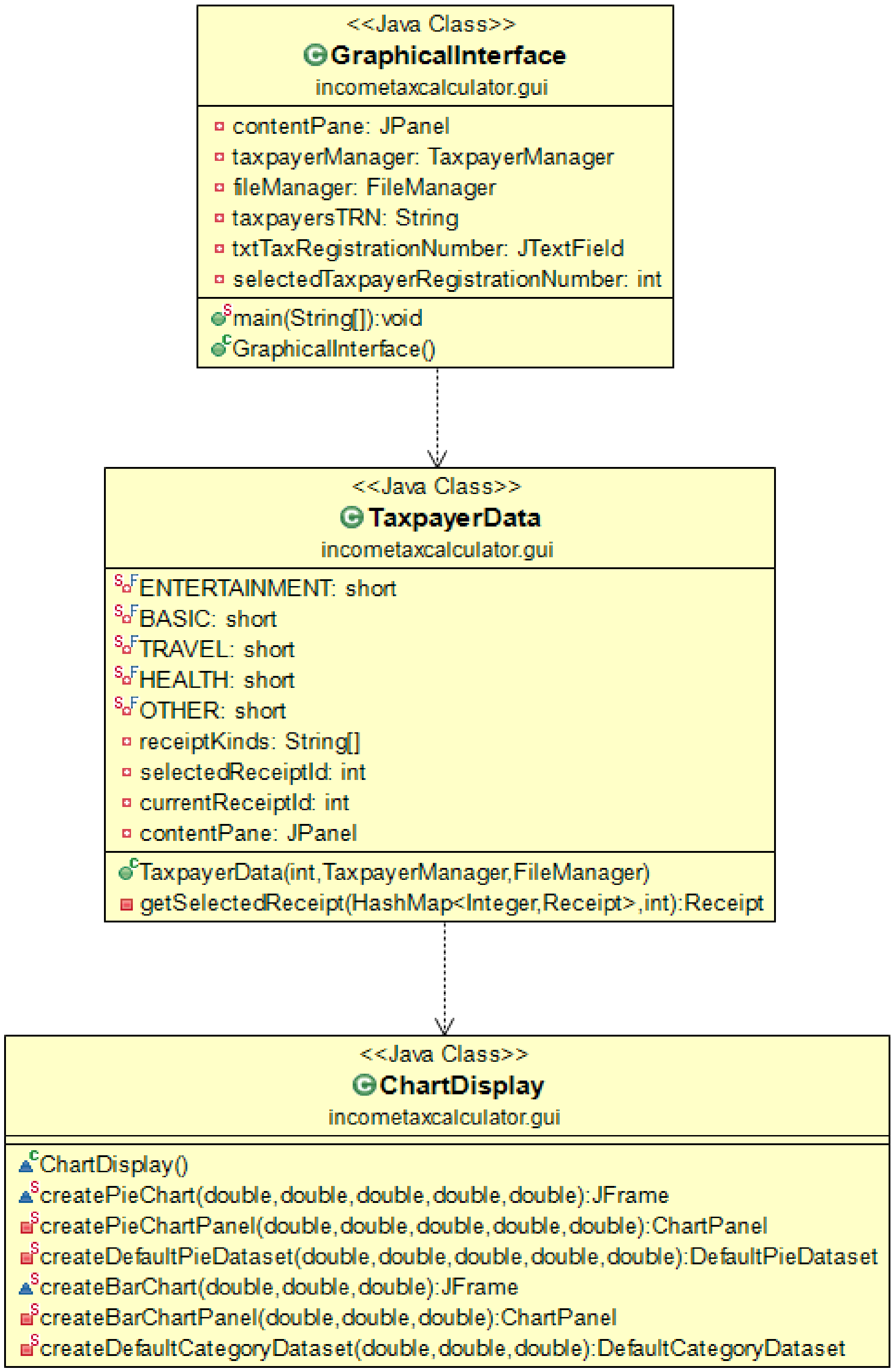


<<Java Class>>
FileManager
incometaxcalculator.data.management
FileManager()
saveLogFile(int,String):void
loadTaxpayer(String):void
isCorrectFileFormat(String):boolean
<<Java Class>>
Taxpayer
incometaxcalculator.data.management
fullname: String
taxRegistrationNumber: int
income: float
amountPerReceiptsKind: float[]
totalReceiptsGathered: int
receiptKinds: String[]
taxpayerStatus: int
taxPercentages: double[][]
incomeLowerThanTresholds: double[][]
baseTaxValues: double[][]
taxValuesSubtracted: double[][]
Taxpayer(String,int,float,int)
calculateBasicTax():double
calculateBasicTaxByStatus(int):double
calculateBasicTaxProcess(int,int):double
addReceipt(Receipt):void
removeReceipt(int):void
getFullname():String
getTaxRegistrationNumber():int
getIncome():float
getReceiptHashMap():HashMap<Integer,Receipt>
getVariationTaxOnReceipts():double
getTotalAmountOfReceipts():float
getTotalReceiptsGathered():int
getTaxpayerStatus():int
getAmountOfReceiptKind(short):float
getTotalTax():double
getBasicTax():double
toString():String
<<Java Class>>
TaxpayerManager
incometaxcalculator.data.management
receiptOwnerTRN: HashMap<Integer,Integer>
taxpayerStatusList: String[]
TaxpayerManager()
createTaxpayer(String,int,String,float):void
createReceipt(int,String,float,String,String,String,String,String,int,int):void
removeTaxpayer(int):void
addReceipt(int,String,float,String,String,String,String,String,int,int):void
removeReceipt(int):void
updateFiles(int):void
containsTaxpayer(int):boolean
containsTaxpayer():boolean
containsReceipt(int):boolean
getTaxpayer(int):Taxpayer
getTaxpayerName(int):String
getTaxpayerStatusString(int):String
getTaxpayerIncome(int):String
getTaxpayerVariationTaxOnReceipts(int):double
getTaxpayerTotalReceiptsGathered(int):int
getTaxpayerAmountOfReceiptKind(int,short):float
getTaxpayerTotalTax(int):double
getTaxpayerBasicTax(int):double
getReceiptHashMap(int):HashMap<Integer,Receipt>
-taxpayerHashMap
0..*
-receiptHashMap
0..*
<<Java Class>>
Company
incometaxcalculator.data.management
name: String
Company(String,String,String,String,int)
getName():String
getCountry():String
getCity():String
getStreet():String
getNumber():int
-company
0..1
<<Java Class>>
Receipt
incometaxcalculator.data.management
id: int
amount: float
kind: String
Receipt(int,String,float,String,Company)
createDate(String):Date
getId():int
getIssueDate():String
getAmount():float
getKind():String
getCompany():Company
-issueDate
0..1
<<Java Class>>
Date
incometaxcalculator.data.management
day: int
month: int
year: int
Date(int,int,int)
getDay():int
getMonth():int
getYear():int
toString():String
-address
0..1
<<Java Class>>
Address
incometaxcalculator.data.management
country: String
city: String
street: String
number: int
Address(String,String,String,int)
getCountry():String
getCity():String
getStreet():String
getNumber():int
toString():String

- <u>Package exceptions</u>

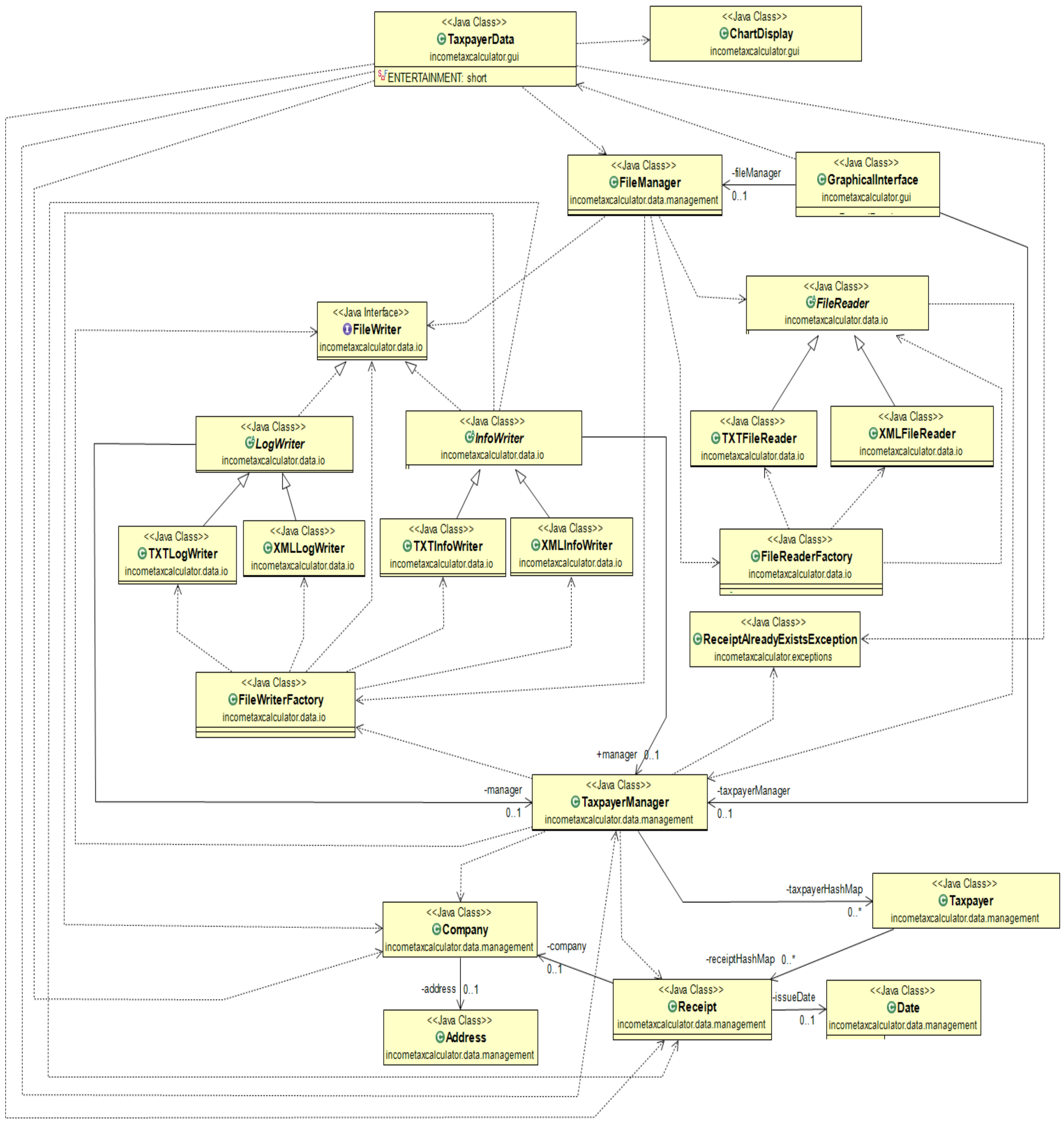

<<Java Class>>
WrongReceiptDateException
incometaxcalculator.exceptions
serialVersionUID: long
WrongReceiptDateException()
<<Java Class>>
WrongFileFormatException
incometaxcalculator.exceptions
serialVersionUID: long
WrongFileFormatException()
<<Java Class>>
WrongReceiptKindException
incometaxcalculator.exceptions
serialVersionUID: long
WrongReceiptKindException()
<<Java Class>>
WrongTaxpayerStatusException
incometaxcalculator.exceptions
serialVersionUID: long
WrongTaxpayerStatusException()
<<Java Class>>
ReceiptAlreadyExistsException
incometaxcalculator.exceptions
serialVersionUID: long
ReceiptAlreadyExistsException()
<<Java Class>>
WrongFileEndingException
incometaxcalculator.exceptions
serialVersionUID: long
WrongFileEndingException()

- Package gui



<<Java Class>>
GraphicalInterface
incometaxcalculator.gui
contentPane: JPanel
taxpayerManager: TaxpayerManager
fileManager: FileManager
taxpayersTRN: String
txtTaxRegistrationNumber: JTextField
selectedTaxpayerRegistrationNumber: int
main(String[]):void
GraphicalInterface()
<<Java Class>>
TaxpayerData
incometaxcalculator.gui
ENTERTAINMENT: short
BASIC: short
TRAVEL: short
HEALTH: short
OTHER: short
receiptKinds: String[]
selectedReceiptId: int
currentReceiptId: int
contentPane: JPanel
TaxpayerData(int,TaxpayerManager,FileManager)
getSelectedReceipt(HashMap<Integer,Receipt>,int):Receipt
<<Java Class>>
ChartDisplay
incometaxcalculator.gui
ChartDisplay()
createPieChart(double,double,double,double,double):JFrame
createPieChartPanel(double,double,double,double,double):ChartPanel
createDefaultPieDataset(double,double,double,double,double):DefaultPieDataset
createBarChart(double,double,double):JFrame
createBarChartPanel(double,double,double):ChartPanel
createDefaultCategoryDataset(double,double,double):DefaultCategoryDataset

- <u>All classes</u>



<<Java Class>> TaxpayerData incometaxcalculator.gui
ENTERTAINMENT: short
<<Java Class>> ChartDisplay incometaxcalculator.gui
<<Java Class>> FileManager incometaxcalculator.data.management
-fileManager 0..1
<<Java Class>> GraphicalInterface incometaxcalculator.gui
<<Java Class>> FileReader incometaxcalculator.data.io
<<Java Interface>> FileWriter incometaxcalculator.data.io
<<Java Class>> LogWriter incometaxcalculator.data.io
<<Java Class>> InfoWriter incometaxcalculator.data.io
<<Java Class>> TXTFileReader incometaxcalculator.data.io
<<Java Class>> XMLFileReader incometaxcalculator.data.io
<<Java Class>> TXTLogWriter incometaxcalculator.data.io
<<Java Class>> XMLLogWriter incometaxcalculator.data.io
<<Java Class>> TXTInfoWriter incometaxcalculator.data.io
<<Java Class>> XMLInfoWriter incometaxcalculator.data.io
<<Java Class>> FileReaderFactory incometaxcalculator.data.io
<<Java Class>> ReceiptAlreadyExistsException incometaxcalculator.exceptions
<<Java Class>> FileWriterFactory incometaxcalculator.data.io
+manager 0..1
-manager 0..1
<<Java Class>> TaxpayerManager incometaxcalculator.data.management
-taxpayerManager 0..1
-taxpayerHashMap 0..*
<<Java Class>> Taxpayer incometaxcalculator.data.management
<<Java Class>> Company incometaxcalculator.data.management
-company 0..1
-receiptHashMap 0..*
-address 0..1
<<Java Class>> Address incometaxcalculator.data.management
<<Java Class>> Receipt incometaxcalculator.data.management
-issueDate 0..1
<<Java Class>> Date incometaxcalculator.data.management

GUI changes

The GUI is now more user friendly, due to some basic changes that were made:

- When loading a taxpayer, the user can now load the taxpayer’s data directly from the corresponding INFO file by the new file chooser button. There also is a log field that displays the file chooser actions.

- The “Select Taxpayer” button is now removed, because the taxpayer can be selected directly from the list just by double clicking at a tax registration number. Furthermore the instruction for selecting the taxpayer is displayed by hovering test on the list.

- There is no need to type the ta registration number so the taxpayer is deleted anymore, the user selects them from the list and only needs to press the “Delete Taxpayer”.

- The “Add Receipt” window now has some completed fields, the receipt id, and the drop-down menu for the receipt kinds. Moreover, the user can now see the correct format needed via hovering text for each field.

- There is a new button, the “Edit Receipt” if there is need to fix something that was wrongly typed when adding the receipt.

- The “Delete Receipt” button now works in the same way that “Delete Taxpayer” does.

Refactoring

- There are no longer any subclasses of the Taxpayer class, due to them having the same algorithm but with different values for each case. So now the calculateBasicTax() method is more simplified as a part of the Taxpayer class, with the use of arrays and for loops. This method is also extracted to two methods for the line reduction.

- The TaxpayerManager class is now simplified by extracting some file-oriented methods to a new class, FileManager, and the implementation of factory pattern.

- As explained before, there are now some Factory classes, FileWriterFactory and FileReaderFactory, that are implemented for simplifying the different file formats. These classes are now used by the updateFiles(), saveLogFile() and loadTaxpayer() methods accordingly.

- The common parts of XMLFileReader and TXTFileReader were moved to the abstract class FileReader and each of them is now responsible for the difference in file formatting.

- The FileWriter class has been stripped of its methods, and is now an interface with the abstract method generateFile(). This method is implemented by the InfoWriter and LogWriter classes, simplified by extracting. Accordingly with the FileReader case, these classes use their xml and txt classes for the difference in file formatting. In both instances, the template method pattern is now implemented.

CLASSES RESPONSIBILITIES AND COLLABORATIONS (CRC CARDS)

Interface Name: FileWriter	
Responsibilities	**Collaborations**
▪ Ensures the classes that implement it will generate a file properly.	▪ LogWriter&Infowriter

Class Name: LogWriter	
Responsibilities	**Collaborations**
▪ Creates a "LOG" file and fills it with the correct lines.	▪ FileWriter, TXTLogWritter & XMLLogWriter, TaxpayerManager.

Class Name: TXTLogWriter	
Responsibilities	**Collaborations**
▪ Creates the ".txt" ending for the file that LogWriter shall make.	▪ LogWriter.
▪ Takes a text and returns the correct format of it for the txt file e.g "Name" to "Name: "	▪ LogWriter.

Class Name: XMLLogWriter	
Responsibilities	**Collaborations**
▪ Creates the ".xml" ending for the file that LogWriter shall make. ▪ Takes a text and returns the correct format of it for the xml file e.g "Name" to "<Name> "and "</Name>"	▪ LogWriter. ▪ LogWriter

Class Name: InfoWriter	
Responsibilities	**Collaborations**
▪ Creates a "INFO" type file and fills it with the correct lines.	▪ FileWriter, TXTInfoWritter and XMLInfoWriter, TaxpayerManager, Company, Receipt.

Class Name: TXTInfoWriter	
Responsibilities	**Collaborations**
▪ Creates the ".txt" ending for the file that LogWriter shall make. ▪ Takes a text and returns the correct format of it for the txt file e.g "Name" to "Name: ".	▪ InfoWriter. ▪ InfoWriter.

Class Name: XMLInfoWriter	
Responsibilities	**Collaborations**
▪ Creates the ".xml" ending for the file that LogWriter shall make. ▪ Takes a word and returns the correct format of it for the xml file, e.g., "Name" to "<Name> "or "</Name>".	▪ InfoWriter. ▪ InfoWriter.

Class Name: FileWriterFactory	
Responsibilities	**Collaborations**
▪ According to the inputs of the file format (txt or xml) and of the io type (info or log), it returns an object (TXTLogWriter, XMLLogWriter, TXTInfoWriter or XMLINfoWriter).	▪ TXTLogWriter, XMLLogWriter, TXTInfoWriter, XMLLInfoWriter.

Class Name: FileReader	
Responsibilities	**Collaborations**
▪ Reads a .txt or .xml file	▪ FileWriter, TXTFileReader, XMLFileReader & TaxpayerManager.

Class Name: Address	
Responsibilities	**Collaborations**
▪ Stores the address information of a company for each receipt.	▪ -

Class Name: Company	
Responsibilities	**Collaborations**
▪ Stores the information of a company for each receipt.	▪ Address.

Class Name: Date	
Responsibilities	**Collaborations**
▪ Stores the issue date information for each receipt.	▪ -

Class Name: FileManager	
Responsibilities	**Collaborations**
▪ Generates a “LOG” file. ▪ Loads a taxpayer given their tax registration number.	▪ FileWriter &FileWriterFactory. ▪ FileReader &FileReaderFactory.

Class Name: Receipt	
Responsibilities	**Collaborations**
▪ Creates an issue date.	▪ Date.
▪ Stores all receipt information.	▪ Company & Date.

Class Name: Taxpayer	
Responsibilities	**Collaborations**
▪ Stores the taxpayer information.	▪ -
▪ Calculates the basic tax.	▪ -
▪ Adds and removes receipts	▪ Receipt
▪ Provides a list of all taxpayer’s receipts.	▪ Receipt
▪ Computes variation tax on receipts.	▪ Receipt

Class Name: TaxpayerManager	
Responsibilities	**Collaborations**
▪ Creates a taxpayer.	▪ Taxpayer.
▪ Removes a taxpayer.	▪ Taxpayer & Receipt.
▪ Creates a receipt.	▪ Company & Receipt.
▪ Adds a receipt on the receipt list	▪ Company, Receipt, FileWriter & FileWriterFactory.
▪ Removes a receipt	▪ Receipt, FileWriter & FileWriterFactory
▪ Displays taxpayer’s status as a text.	▪ Taxpayer.

Class Name: GraphicalInterface	
Responsibilities	**Collaborations**
▪ Executes the project's GUI. ▪ Loads, selects, and removes a taxpayer.	▪ - ▪ FileManager, TaxpayerManager & TaxpayerData.

Class Name: TaxpayerData	
Responsibilities	**Collaborations**
▪ Displays the taxpayer's information.	▪ Taxpayer, TaxpayerManager, Company, Receipt & FileManager.
▪ Adds a new receipt on the receipt list.	▪ Receipt & Company.
▪ Edits a receipt from the receipt list.	▪ Receipt & Company.
▪ Deletes a receipt from the receipt list.	▪ Receipt & Company.
▪ Displays the taxpayer's report.	▪ ChartDisplay, Taxpayer & TaxpayerManager.
▪ Saves a LOG file with all taxpayer's data.	▪ FileManager, Taxpayer & TaxpayerManager

Class Name: ChartDisplay	
Responsibilities	**Collaborations**
▪ Displays a bar chart with taxpayer's income and variation tax increase/decrease. ▪ Displays a pie chart with percentages of each kind of receipt.	▪ -. ▪ -