# INCOME TAX CALCULATOR

## OVERALL REPORT

VERSION <1.0>

**Πηνελόπη Ελευθεριάδη 3221**

**Κωνσταντίνος Γεωργίου 4333**

# TABLE OF CONTENTS

## INTRODUCTION

Ο στόχος της εργασίας αυτής είναι η διαμόρφωση μιας υπάρχουσας εφαρμογής, η οποία υπολογίζει την φορολογία των πολιτών της πολιτείας της Minnesota. Η εφαρμογή δέχεται txt ή xml αρχεία που περιέχουν πληροφορίες για τους πολίτες και υπολογίζει την φορολογία τους. Επίσης, παρέχεται η δυνατότητα εμφάνισης των δεδομένων μέσω διαγραμμάτων σε μορφή bar/pie charts.

## REFACTORED DESIGN

### USE CASES

Specify the use cases of the application.

| | |
|---|---|
| **Use case ID** | 1. Load Taxpayer |
| **Actors** | User |
| **Pre conditions** | The TIN (Tax Identification Number) should match an existing taxpayer. |
| **Main flow of events** | 1. The use case starts when the user opens the application.<br>2. The user selects the “Load Taxpayer” button<br>3. The use case shows a pop up window that asks the user to insert the tax registration number.<br>3.1. The user fills in the registration number and selects the type of file(txt or xml) that the information are.<br>3.2. The user clicks “OK” and the use case closes the pop up window and adds the registration number in the “Tax Registration Number” box in the main window. |
| **Alternative flow 1** | If the user doesn't insert a valid registration number, a pop up window will appear to notify him that this file doesn't exist. |
| **Alternative** | If the user inserts a number with less or more than 9 digits or insert letters or |

| | |
|---|---|
| **flow 2** | character the use case will inform him that the registration number must contain only 9 digits. |
| **Post conditions** | The file for the taxpayer is loaded. |

| | |
|---|---|
| **Use case ID** | 2.Delete Taxpayer |
| **Actors** | User |
| **Pre conditions** | |
| **Main flow of events** | 1.The use case starts when the user clicks the Delete Taxpayer button<br>2. The use case shows a pop up window that asks the user to insert the tax registration number.<br>3.The use case deletes the taxpayer with this registration number, |

| | |
|---|---|
| **Use case ID** | 3.Select taxpayer |
| **Actors** | User |
| **Pre conditions** | A taxpayer should be loaded |
| **Main flow of events** | 1. The use case starts when the user clicks the “Select Taxpayer” button.<br>2. The use case shows a pop up window that asks the user to insert the tax registration number that the user wants to be displayed.<br>1. The user fills in the registration number and clicks “OK”<br>2. The use case shows a pop up window that contain the information of the taxpayer. The information are the name, TRN, status, income and receipts. |

| | |
|---|---|
| **Alternative flow 1** | If the user inserts a registration number that isn't loaded, a pop up window will appear to notify him. |

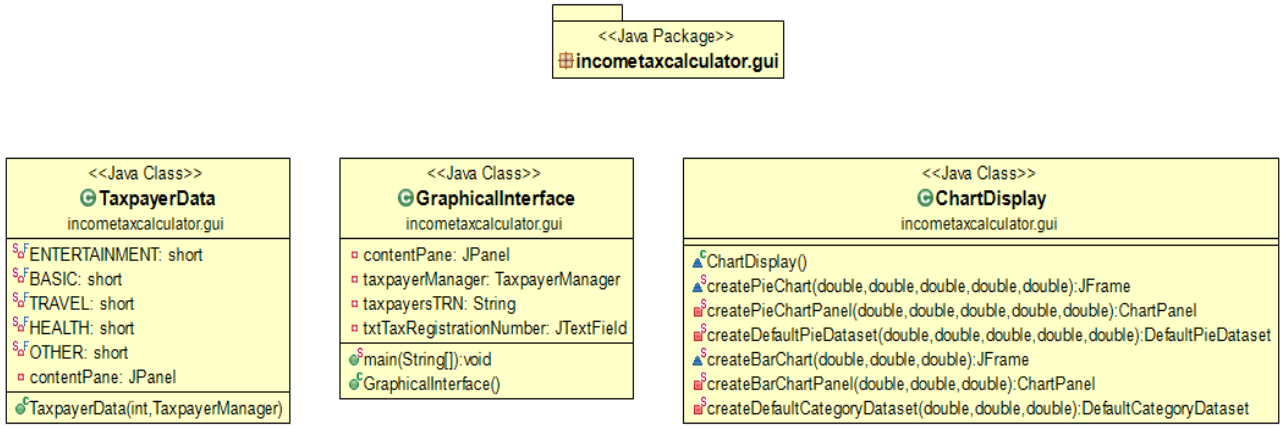
| | |
|---|---|
| **Use case ID** | 4.Add Receipt |
| **Actors** | User |
| **Pre conditions** | A taxpayer should be loaded and the window with the information of the taxpayer should be open. |
| **Main flow of events** | 1. The use case starts when the user clicks the "Add Receipt" button.<br>2. The use case shows a pop up window that asks the user to fill the receipt information that are: Receipt id, date, kind, amount, company, country, city, street, number. |
| **Alternative flow 1** | If the user inserts a registration number that isn't loaded, a pop up window will appear to notify him. |

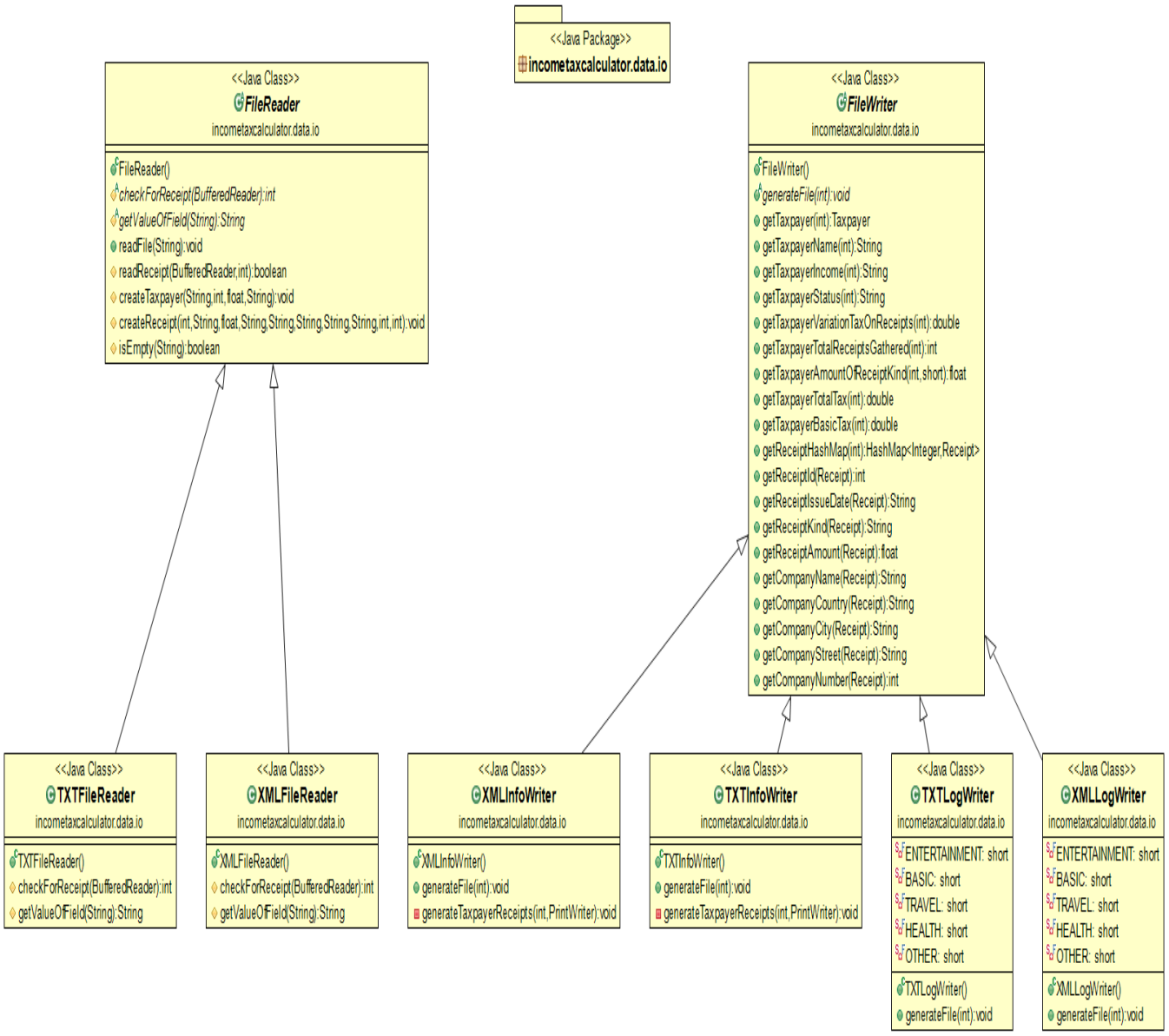
| | |
|---|---|
| **Use case ID** | 5.Delete Receipt |
| **Actors** | User |
| **Pre conditions** | A taxpayer should be loaded and the window with the information of the taxpayer should be open. |
| **Main flow of events** | 1. The use case starts when the user presses the “Delete Receipt” button.<br>2. The use case shows a pop up window that asks the user to insert the receipt ID.<br>3.The user inserts the id(number)of the receipt that wants to delete.<br>4.The use case delete the receipt and updates the “Receipts” box with the existing receipts. |

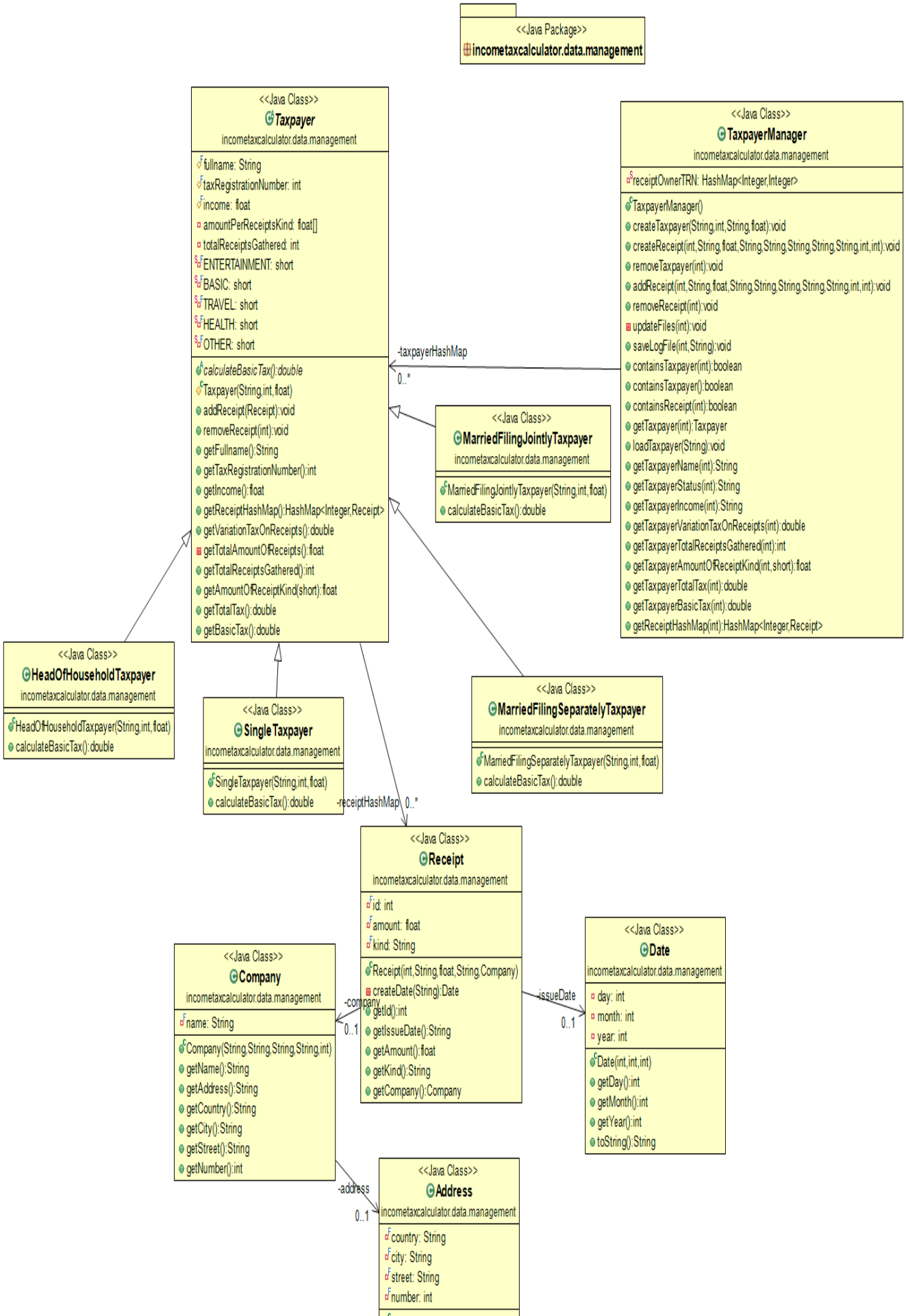
| | |
|---|---|
| **Use case ID** | 6.View Report |
| **Actors** | User |
| **Pre conditions** | A taxpayer should be loaded and the window with the information of the taxpayer should be open. |
| **Main flow of events** | 1. The use case starts when the user presses the “View Report” button.<br>2. The use case shows two pop up window .<br>3. The first one is a bar chart with the total tax of the taxpayer and the analysis of the basic tax and the increase or the decrease because of the receipts.<br>4. The second window is a pie chart for the analysis of the receipts. It shows the percentage of the total amount of every kind of the receipt. |

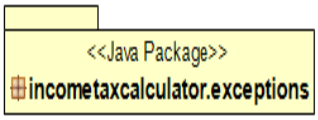
| | |
|---|---|
| **Use case ID** | 7.Save Data |
| **Actors** | User |
| **Pre conditions** | A taxpayer should be loaded and the window with the information of the taxpayer should be open. |
| **Main flow of events** | 1. The use case starts when the user presses the “Save Data” button.<br>2. The use case shows a pop up window that asks the format of the file.<br>3. The user presses selects the format and presses “ok” to save the information for the taxpayer. |
| **Alternative flow 1** | The user can select the “Cancel” button. |
| | |

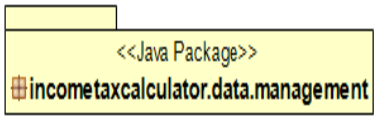
## ARCHITECTURE

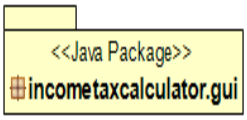


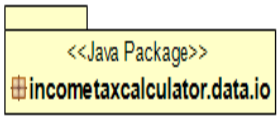

<<Java Package>>
incometaxcalculator.data.io
<<Java Class>>
FileReader
incometaxcalculator.data.io
FileReader()
checkForReceipt(BufferedReader):int
getValueOfField(String):String
readFile(String):void
readReceipt(BufferedReader,int):boolean
createTaxpayer(String,int,float,String):void
createReceipt(int,String,float,String,String,String,String,String,int,int):void
isEmpty(String):boolean
<<Java Class>>
FileWriter
incometaxcalculator.data.io
FileWriter()
generateFile(int):void
getTaxpayer(int):Taxpayer
getTaxpayerName(int):String
getTaxpayerIncome(int):String
getTaxpayerStatus(int):String
getTaxpayerVariationTaxOnReceipts(int):double
getTaxpayerTotalReceiptsGathered(int):int
getTaxpayerAmountOfReceiptKind(int,short):float
getTaxpayerTotalTax(int):double
getTaxpayerBasicTax(int):double
getReceiptHashMap(int):HashMap<Integer,Receipt>
getReceiptId(Receipt):int
getReceiptIssueDate(Receipt):String
getReceiptKind(Receipt):String
getReceiptAmount(Receipt):float
getCompanyName(Receipt):String
getCompanyCountry(Receipt):String
getCompanyCity(Receipt):String
getCompanyStreet(Receipt):String
getCompanyNumber(Receipt):int
<<Java Class>>
TXTFileReader
incometaxcalculator.data.io
TXTFileReader()
checkForReceipt(BufferedReader):int
getValueOfField(String):String
<<Java Class>>
XMLFileReader
incometaxcalculator.data.io
XMLFileReader()
checkForReceipt(BufferedReader):int
getValueOfField(String):String
<<Java Class>>
XMLInfoWriter
incometaxcalculator.data.io
XMLInfoWriter()
generateFile(int):void
generateTaxpayerReceipts(int,PrintWriter):void
<<Java Class>>
TXTInfoWriter
incometaxcalculator.data.io
TXTInfoWriter()
generateFile(int):void
generateTaxpayerReceipts(int,PrintWriter):void
<<Java Class>>
TXTLogWriter
incometaxcalculator.data.io
ENTERTAINMENT: short
BASIC: short
TRAVEL: short
HEALTH: short
OTHER: short
TXTLogWriter()
generateFile(int):void
<<Java Class>>
XMLLogWriter
incometaxcalculator.data.io
ENTERTAINMENT: short
BASIC: short
TRAVEL: short
HEALTH: short
OTHER: short
XMLLogWriter()
generateFile(int):void


<<Java Package>>
incometaxcalculator.data.management
<<Java Class>>
Taxpayer
incometaxcalculator.data.management
fullname: String
taxRegistrationNumber: int
income: float
amountPerReceiptsKind: float[]
totalReceiptsGathered: int
ENTERTAINMENT: short
BASIC: short
TRAVEL: short
HEALTH: short
OTHER: short
calculateBasicTax():double
Taxpayer(String,int,float)
addReceipt(Receipt):void
removeReceipt(int):void
getFullname():String
getTaxRegistrationNumber():int
getIncome():float
getReceiptHashMap():HashMap<Integer,Receipt>
getVariationTaxOnReceipts():double
getTotalAmountOfReceipts():float
getTotalReceiptsGathered():int
getAmountOfReceiptKind(short):float
getTotalTax():double
getBasicTax():double
<<Java Class>>
TaxpayerManager
incometaxcalculator.data.management
receiptOwnerTRN: HashMap<Integer,Integer>
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
saveLogFile(int,String):void
containsTaxpayer(int):boolean
containsTaxpayer():boolean
containsReceipt(int):boolean
getTaxpayer(int):Taxpayer
loadTaxpayer(String):void
getTaxpayerName(int):String
getTaxpayerStatus(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..*
<<Java Class>>
MarriedFilingJointlyTaxpayer
incometaxcalculator.data.management
MarriedFilingJointlyTaxpayer(String,int,float)
calculateBasicTax():double
<<Java Class>>
HeadOfHouseholdTaxpayer
incometaxcalculator.data.management
HeadOfHouseholdTaxpayer(String,int,float)
calculateBasicTax():double
<<Java Class>>
SingleTaxpayer
incometaxcalculator.data.management
SingleTaxpayer(String,int,float)
calculateBasicTax():double
<<Java Class>>
MarriedFilingSeparatelyTaxpayer
incometaxcalculator.data.management
MarriedFilingSeparatelyTaxpayer(String,int,float)
calculateBasicTax():double
-receiptHashMap
0..*
<<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
<<Java Class>>
Company
incometaxcalculator.data.management
name: String
Company(String,String,String,String,int)
getName():String
getAddress():String
getCountry():String
getCity():String
getStreet():String
getNumber():int
-company
0..1
-address
0..1
<<Java Class>>
Address
incometaxcalculator.data.management
country: String
city: String
street: String
number: int










Above are the UMLs of the existing project.

## Use Case Tests

| | |
|---|---|
| **Use case ID** | 1. Load Taxpayer Test |
| **Actors** | User |
| **Main flow of events** | 1. The user doesn't insert a valid registration number the use case shows a pop up window will appear with a message “This file doesn't exist”.<br>2. The user inserts a number with less or more than 9 digits or insert letters or character and the use case will inform him that the registration number must contain only 9 digits. |

| | |
|---|---|
| **Use case ID** | 2. Delete Taxpayer Test |
| **Actors** | User |
| **Main flow of events** | 1. The user doesn't insert a valid registration number the use case shows a pop up window will appear with a message “This file doesn't exist”.<br>2. The user inserts a number with less or more than 9 digits or insert letters or character and the use case will inform him that the registration number must contain only 9 digits. |

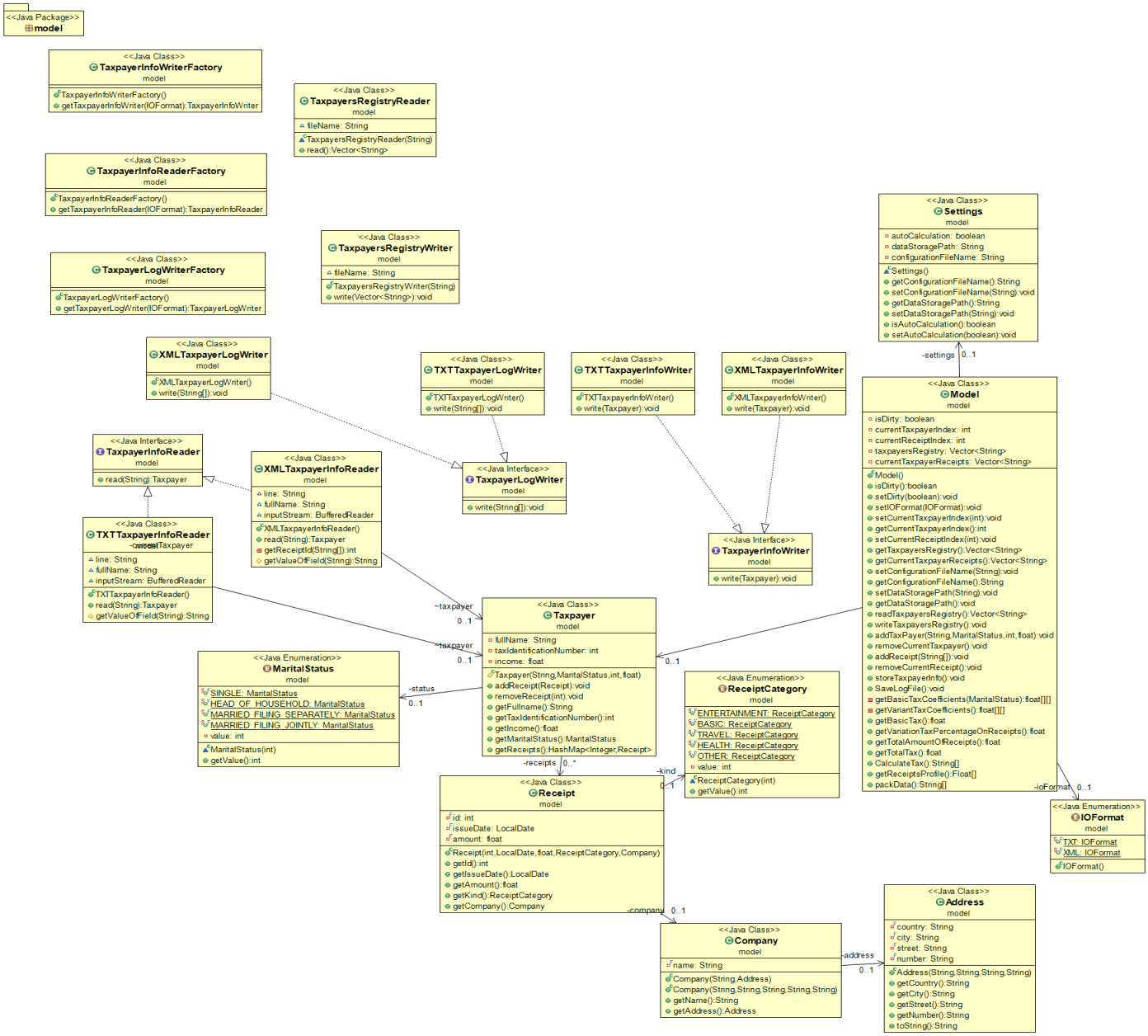
| Use case ID | 3. Select taxpayer Test |
|---|---|
| Actors | User |
| Main flow of events | 1. The user inserts a registration number that isn't loaded, a pop up window will appear to notify him.<br>2. The user inserts a number with less or more than 9 digits or insert letters or character and the use case will inform him that the registration number must contain only 9 digits. |

| Use case ID | 4. Add Receipt Test |
|---|---|
| Actors | User |
| Main flow of events | 1. The user inserts a registration number that isn't loaded, a pop up window will appear to notify him. |

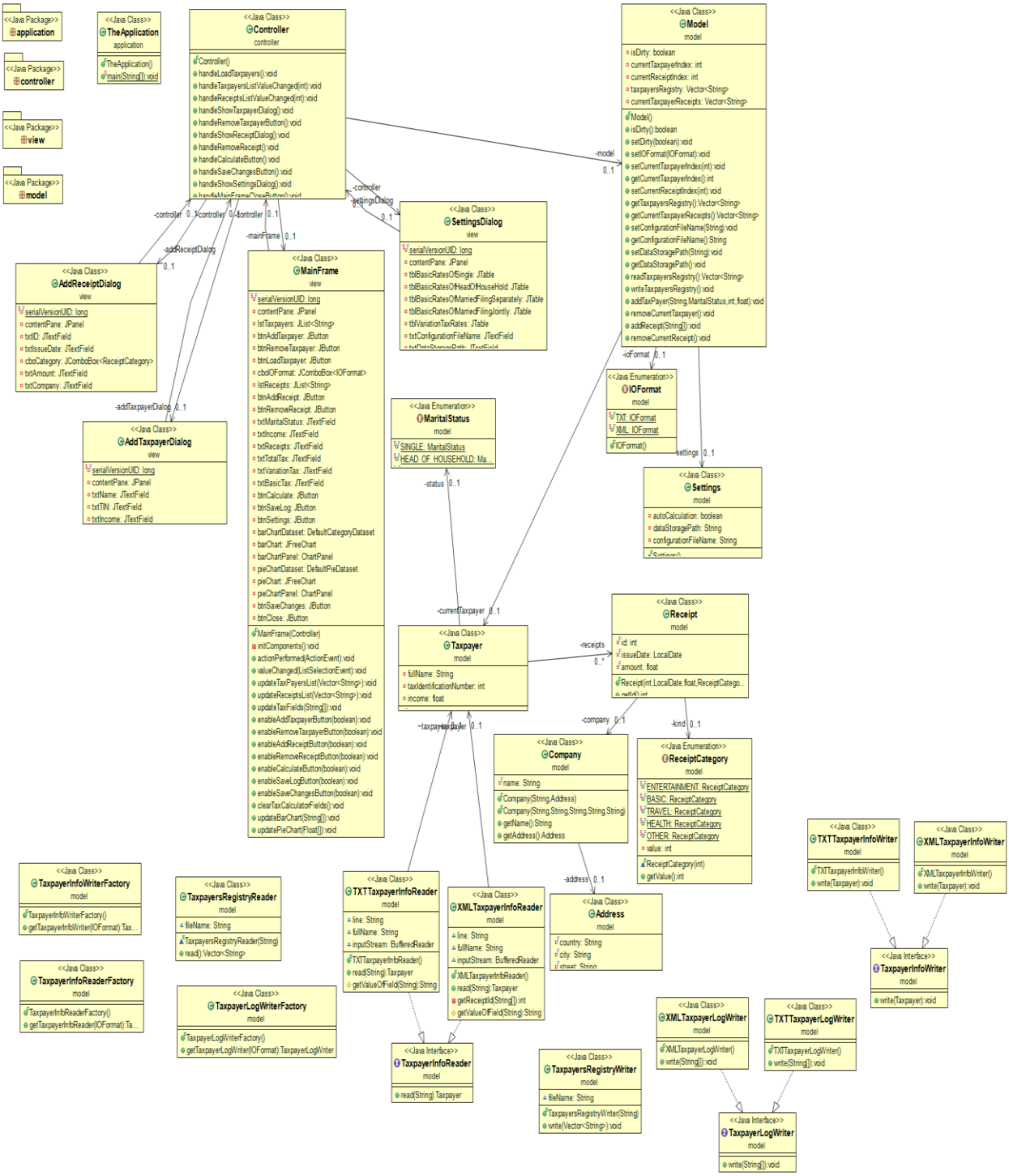
| Use case ID | 5. Delete Receipt Test |
|---|---|
| Actors | User |
| Main flow of events | 1. The user inserts a receipt number that doesn't exist, a pop up window will appear to notify him. |

# UML Diagrams of the Refactored application

## MODEL



## Project

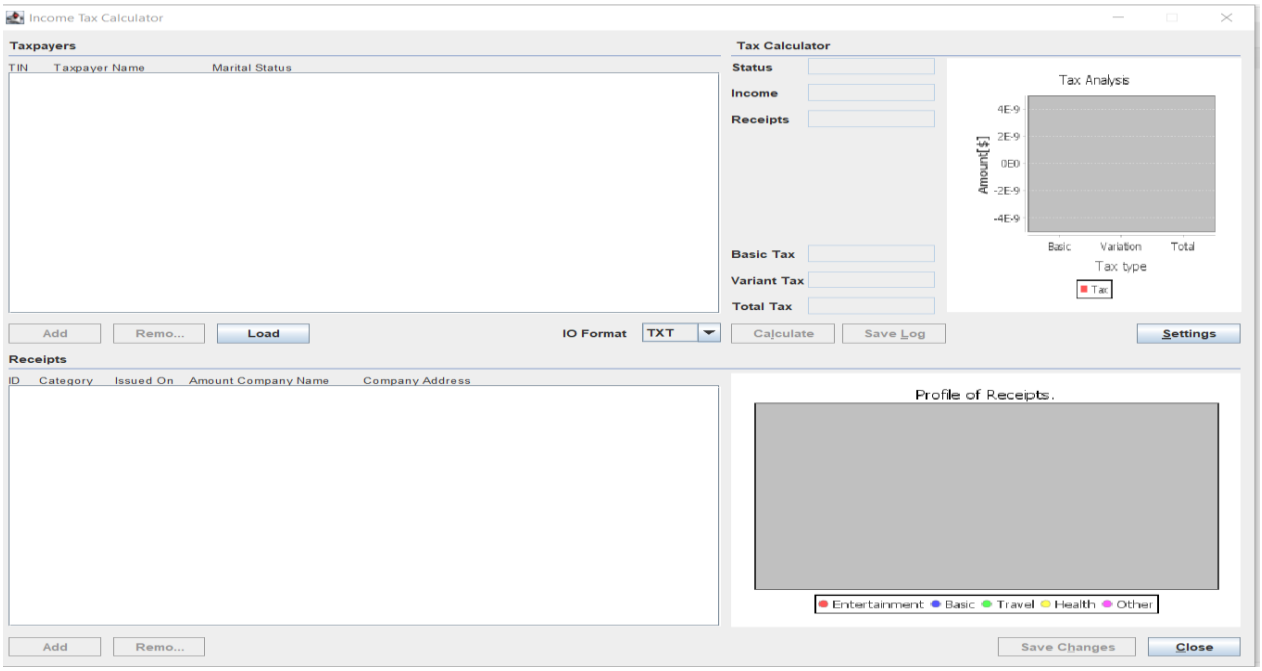

<<Java Package>> application
<<Java Package>> controller
<<Java Package>> view
<<Java Package>> model
<<Java Class>> TheApplication
application
TheApplication()
main(String[]):void
<<Java Class>> Controller
controller
Controller()
handleLoadTaxpayers():void
handleTaxpayersListValueChanged(int):void
handleReceiptsListValueChanged(int):void
handleShowTaxpayerDialog():void
handleRemoveTaxpayerButton():void
handleShowReceiptDialog():void
handleRemoveReceipt():void
handleCalculateButton():void
handleSaveChangesButton():void
handleShowSettingsDialog():void
<<Java Class>> Model
model
isDirty: boolean
currentTaxpayerIndex: int
currentReceiptIndex: int
taxpayersRegistry: Vector<String>
currentTaxpayerReceipts: Vector<String>
Model()
isDirty():boolean
setDirty(boolean):void
setIOFormat(IOFormat):void
setCurrentTaxpayerIndex(int):void
getCurrentTaxpayerIndex():int
setCurrentReceiptIndex(int):void
getTaxpayersRegistry():Vector<String>
getCurrentTaxpayerReceipts():Vector<String>
setConfigurationFileName(String):void
getConfigurationFileName():String
setDataStoragePath(String):void
getDataStoragePath():void
readTaxpayersRegistry():Vector<String>
writeTaxpayersRegistry():void
addTaxPayer(String,MaritalStatus,int,float):void
removeCurrentTaxpayer():void
addReceipt(String[]):void
removeCurrentReceipt():void
-model 0..1
-controller 0..1
-controller 0..1
-controller 0..1
-controller 0..1
-mainFrame 0..1
-addReceiptDialog 0..1
-addTaxpayerDialog 0..1
<<Java Class>> SettingsDialog
view
serialVersionUID: long
contentPane: JPanel
tblBasicRatesOfSingle: JTable
tblBasicRatesOfHeadOfHouseHold: JTable
tblBasicRatesOfMarriedFilingSeparately: JTable
tblBasicRatesOfMarriedFilingJointly: JTable
tblVariationTaxRates: JTable
txtConfigurationFileName: JTextField
<<Java Class>> AddReceiptDialog
view
serialVersionUID: long
contentPane: JPanel
txtID: JTextField
txtIssueDate: JTextField
cboCategory: JComboBox<ReceiptCategory>
txtAmount: JTextField
txtCompany: JTextField
<<Java Class>> MainFrame
view
serialVersionUID: long
contentPane: JPanel
lstTaxpayers: JList<String>
btnAddTaxpayer: JButton
btnRemoveTaxpayer: JButton
btnLoadTaxpayer: JButton
cboIOFormat: JComboBox<IOFormat>
lstReceipts: JList<String>
btnAddReceipt: JButton
btnRemoveReceipt: JButton
txtMaritalStatus: JTextField
txtIncome: JTextField
txtReceipts: JTextField
txtTotalTax: JTextField
txtVariationTax: JTextField
txtBasicTax: JTextField
btnCalculate: JButton
btnSaveLog: JButton
btnSettings: JButton
barChartDataset: DefaultCategoryDataset
barChart: JFreeChart
barChartPanel: ChartPanel
pieChartDataset: DefaultPieDataset
pieChart: JFreeChart
pieChartPanel: ChartPanel
btnSaveChanges: JButton
btnClose: JButton
MainFrame(Controller)
initComponents():void
actionPerformed(ActionEvent):void
valueChanged(ListSelectionEvent):void
updateTaxPayersList(Vector<String>):void
updateReceiptsList(Vector<String>):void
updateTaxFields(String[]):void
enableAddTaxpayerButton(boolean):void
enableRemoveTaxpayerButton(boolean):void
enableAddReceiptButton(boolean):void
enableRemoveReceiptButton(boolean):void
enableCalculateButton(boolean):void
enableSaveLogButton(boolean):void
enableSaveChangesButton(boolean):void
clearTaxCalculatorFields():void
updateBarChart(String[]):void
updatePieChart(Float[]):void
<<Java Class>> AddTaxpayerDialog
view
serialVersionUID: long
contentPane: JPanel
txtName: JTextField
txtTIN: JTextField
txtIncome: JTextField
<<Java Enumeration>> IOFormat
model
TXT: IOFormat
XML: IOFormat
IOFormat()
-ioFormat 0..1
-settings 0..1
<<Java Enumeration>> MaritalStatus
model
SINGLE: MaritalStatus
HEAD_OF_HOUSEHOLD: Ma...
-status 0..1
<<Java Class>> Settings
model
autoCalculation: boolean
dataStoragePath: String
configurationFileName: String
-currentTaxpayer 0..1
<<Java Class>> Taxpayer
model
fullName: String
taxIdentificationNumber: int
income: float
-taxpayerBuyer 0..1
<<Java Class>> Receipt
model
id: int
issueDate: LocalDate
amount: float
Receipt(int,LocalDate,float,ReceiptCatego...
-receipts 0..*
-company 0..1
-kind 0..1
-address 0..1
<<Java Class>> Company
model
name: String
Company(String,Address)
Company(String,String,String,String,String)
getName():String
getAddress():Address
<<Java Enumeration>> ReceiptCategory
model
ENTERTAINMENT: ReceiptCategory
BASIC: ReceiptCategory
TRAVEL: ReceiptCategory
HEALTH: ReceiptCategory
OTHER: ReceiptCategory
value: int
ReceiptCategory(int)
getValue():int
<<Java Class>> Address
model
country: String
city: String
street: String
<<Java Class>> TaxpayerInfoWriterFactory
model
TaxpayerInfoWriterFactory()
getTaxpayerInfoWriter(IOFormat):Tax...
<<Java Class>> TaxpayersRegistryReader
model
fileName: String
TaxpayersRegistryReader(String)
read():Vector<String>
<<Java Class>> TXTTaxpayerInfoReader
model
line: String
fullName: String
inputStream: BufferedReader
TXTTaxpayerInfoReader()
read(String):Taxpayer
getValueOfField(String):String
<<Java Class>> XMLTaxpayerInfoReader
model
line: String
fullName: String
inputStream: BufferedReader
XMLTaxpayerInfoReader()
read(String):Taxpayer
getReceiptId(String[]):int
getValueOfField(String):String
<<Java Class>> TXTTaxpayerInfoWriter
model
TXTTaxpayerInfoWriter()
write(Taxpayer):void
<<Java Class>> XMLTaxpayerInfoWriter
model
XMLTaxpayerInfoWriter()
write(Taxpayer):void
<<Java Class>> TaxpayerInfoReaderFactory
model
TaxpayerInfoReaderFactory()
getTaxpayerInfoReader(IOFormat):Ta...
<<Java Class>> TaxpayerLogWriterFactory
model
TaxpayerLogWriterFactory()
getTaxpayerLogWriter(IOFormat):TaxpayerLogWriter
<<Java Interface>> TaxpayerInfoReader
model
read(String):Taxpayer
<<Java Class>> TaxpayersRegistryWriter
model
fileName: String
TaxpayersRegistryWriter(String)
write(Vector<String>):void
<<Java Interface>> TaxpayerInfoWriter
model
write(Taxpayer):void
<<Java Class>> XMLTaxpayerLogWriter
model
XMLTaxpayerLogWriter()
write(String[]):void
<<Java Class>> TXTTaxpayerLogWriter
model
TXTTaxpayerLogWriter()
write(String[]):void
<<Java Interface>> TaxpayerLogWriter
model
write(String[]):void

# Προβλήματα και λύσεις

1. Αρχικά μεταποιήσαμε την αρχιτεκτονική της εφαρμογής, χρησιμοποιώντας το pattern MVC(Model View Controller). Συγκεκριμένα, οι κλάσεις MainFrame, AddTaxpayerDialog και AddReceiptDialog αποτελούν το View. Η κλαση Model το Model και η κλαση Controller τον Controller.

   Η MainFrame είναι η βασική φόρμα όπου όταν ο χρήστης αλληλοεπιδρά ενεργοποιείται , καλεί την Controller, η οποία ενημερώνοντας και ζητώντας πληροφορίες από την Model αποφασίζει τι θα πραγματοποιηθεί και ενημερώνει την MainFrame.

2. Μεταποίηση του GUI:Για την αναδιαμόρφωση του GUI χρησιμοποιήσαμε το Java Swing toolkit, WindowBuilder και JFreeChart. Οι αλλαγές που έγιναν σύμφωνα με τα υπαρχοντα use cases είναι τα εξής:



- **Load Taxpayer**: πλέον η φόρτωση των Taxpayers γίνεται με το πατημα του κουμπί [Load]. Εμφανίζονται οι Taxpayers και για τον καθένα εμφανίζεται ο αριθμός TIN και το Marital Status του. Ο χρήστης επιλέγει από αυτή την λίστα(use case Select Taxpayer) και εμφανίζεται η λίστα με τις αποδείξεις που του αντιστοιχούν. Η εφαρμογη διαβαζει και γραφει το μητρωο φορολογουμενων (TaxpayersRegistry.txt)

- **Add Taxpayer**: αφού έχουμε κάνει Load μπορούμε να προσθέσουμε κάποιον νέο taxpayers πατώντας το κουμπί [Add]. Εμφανίζεται ένα pop-up dialog box οπού ο χρήστης εισάγει τις πληροφορίες για τον νέο taxpayer (Full Name, TIN, Income, Marital Status).

- **Delete Taxpayer**: αφού έχουμε κανει Load και έχει επιλεχθεί κάποιος taxpayer μπορούμε να τον αφαιρέσουμε με το button [Remove]. Εμφανίζεται ένα pop up confirmation dialog box που επιβεβαιώνει ότι ο χρήστης θέλει να πραγματοποιηθεί η αλλαγή.

- Add Receipt: Αφού έχουμε κάνει Load και έχει επιλεχθεί κάποιος taxpayer, έχει εμφανιστεί η λίστα με τις αποδείξεις του. Μπορεί ο χρήστης να προσθέσει μια νεα απόδειξη για τον συγκεκριμένο taxpayer πατώντας το κουμπί [Add]. Εμφανίζεται ένα pop up dialog box οπού ο χρήστης εισάγει τις πληροφορίες για την νέα απόδειξη ( id, issue date, category, amount, company, Company address: country, city, street, number).

- Delete Receipt: Επιλέγοντας κάποιο receipt το κουμπί [Remove] μπορεί ο χρήστης να αφαιρέσει την συγκεκριμένη απόδειξη. Εμφανίζεται ένα pop up window για την επιβεβαιώση ότι ο χρήστης θέλει να πραγματοποιηθεί η αφαίρεση.

- View Report: Έχοντας επιλέξει κάποιον Taxpayer υπάρχει η επιλογή <<Calculate>> που πραγματοποιεί το συγκεκριμένο use case. Συγκεκριμένα , εμφανίζει τα 2 charts (bar chart, pie chart )

- Save Data: Αφού γίνει ο υπολογισμός και εμφανίσει τα charts μπορεί ο χρήστης να αποθηκεύσει τις αλλαγές τα αποτελεσματα του υπολογισμου με το button [Save Log].

- Settings: πατώντας το κουμπί [Settings] ο χρήστης μπορεί να μεταποιήσει τους παράγοντες για τον υπολογισμό του φόρου (πχ. όρια εισοδήματος για κάθε Marital Status) όπως και να προσθέσει configuration files ή να αλλάξει που θα αποθηκεύονται τα αρχεία(Data Storage Path)

- Close: Ο χρήστης μπορεί να κλείσει την εφαρμογή με το κουμπί [Close].

- Τέλος, για την εμφάνιση των pop up παραθύρων υπάρχουν οι κλάσεις AddReceiptDialog, AddTaxpayerDialog, SettingsDialog οπού διαχειρίζονται και καλούν αντίστοιχα κατά την αλληλεπίδραση με τον χρήστη την Controller.

3.Πακέτο data.management : Μεταφορά αρχείων στο πακέτο Model. Η λειτουργίες της κλάσης TaxpayerManager μεταφέρθηκαν στην κλάση Model. Αλλαγές στις κλάσεις :

- Company: Σβήστηκαν συναρτήσεις που δεν χρειαζόνταν (δεν αφορούσαν το company) και έμειναν πληροφορίες μόνο για το name και το Address.( Unnecessary Complexity)

- Address: μετατροπή του number σε String , για τη περίπτωση του αριθμού με μορφή $24^{A}$.

- Date: διαγραφή της κλάσης και χρήση της Local Date της Java όπου χρειάζεται.

- Δημιουργία enumerators για το Marital Status και Receipt Category και IO Formats(TXT,XML) και χρήση τους στις υπόλοιπες κλάσεις χωρίς την χρήση των constants που υπήρχαν (ENTERTAIMENT etc).

- Receipt: αφαίρεση μεθόδων που υπολογίζουν δεδομένα( το οποίο γίνεται στο Model).

- Taxpayer: μεταφορά μεθόδων που δεν αφορούν το αντικείμενο(υπολογισμού δεδομένων, το οποίο γίνεται στο Model).

- Settings: νέα κλάση για την προσθήκη της λειτουργίας settings της φόρμας.

- Model: Η συγκεκριμένη κλάση είναι υπεύθυνή για τα δεδομένα και το status της εφαρμογής. Ενημερώνεται για τις αλλαγές που συμβαίνουν κατά την χρήση της εφαρμογής και προσφέρει τα απαραίτητα δεδομένα στην Controller ώστε να γίνουν οι λειτουργίες. Εδώ αποθηκεύεται ο taxpayer που επέλεξε ο χρήστης , το σύνολο των αποδείξεων του συγκεκριμένου taxpayer και η επιλεγμένη receipt όταν επιλεχθεί . Γίνεται η σύνδεση με το I/O και γίνονται οι υπολογισμοί για τα charts, το οποίο γινόταν στο project στην κλάση Taxpayer. Πιο αναλυτικά :

  -οι συντελεστές για τον υπολογισμού φόρου που βασίζονται στο Marital status πλέον υπάρχουν σε ξεχωριστά αρχεία της μορφής *MARITAL_STATUS.tcf.* Στην function getBasicTaxCoefficients σύμφωνα με το marital status διαβάζεται το ανάλογο αρχείο και δίνονται οι συντελεστές.

  -στην getBasicTax παίρνοντας το income του συγκεκριμένου taxpayer που έχει επιλεχθεί και τους συντελεστές σύμφωνα με το marital status του από την getBasicTaxCoefficients υπολογίζεται ο βασικός φόρος.

  - οι συντελεστές για την μείωση ή αύξηση του φόρου σύμφωνα με το συνολικό ποσό αποδείξεων αγορών του taxpayer βρίσκονται στο αρχείο Variant.tcf. Η getVariantTaxCoefficients διαβάζει αυτό το αρχείο και επιστρέφει έναν πίνακα με τους συντελεστές.

  - η getVariantTaxPercentageOnReceipts μας δινει το ποσοστο των αποδειξεων ως προ το εισοδημα.

  -getTotalTax υπολογίζει τον συνολικό φόρο .

  -CalculateTax υπολογιζει και επιστρεφει τα επιμερους συστατικα του φορου (βασικος, μεταβλητος, κλπ)

  -getReceiptsProfile επιστρεφει ένα το σύνολο των αποδείξεων για κάθε Receipt kind.

4.Πακέτο data.io: Για το συγκεκριμένο πακέτο χρησιμοποιήθηκε το pattern Factory. Για τις 3 λειτουργίες (Read Info, Write Info και Write Log) δημιουργήθηκαν τα κατάλληλα interfaces και τα κατάλληλα Factories τα οποία ανάλογα την επιλογή του χρήστη καλούν τις κλάσεις για την διαχείριση αυτών των ενεργειών. Συγκεκριμένα :

- Για την ανάγνωση των αρχείων που υπάρχουν ήδη στο σύστημα δημιουργήθηκε το interface TaxpayerInfoReader και το Factory TaxpayerInfoReaderFactory που δημιουργεί ανάλογα το είδος του αρχείου που θέλουμε να διαβάσουμε ενα αντικειμενο τυπου TXTTaxpayerInfoReader ή XMLTaxpayerInfoReader.

- Για το γράψιμο αρχείων δημιουργήθηκε το interface TaxpayerInfoWriter και το factory TaxpayerInfoWriterFactory που δημιουργεί ανάλογα το είδος του αρχείου που θέλουμε να γράψουμε ενα αντικειμενο τυπου TXT TaxpayerInfoWriter ή XMLTaxpayerInfoWriter. Προφανώς, οι συναρτήσεις που υπήρχαν στην FileWriter και καλούσαν μεθόδους της TaxpayerManager δεν υπάρχουν πλέον και η επικοινωνία των κλάσεων γίνεται άμεσα με των κλασεων Taxpayer και Model.

- Για το γράψιμο αρχείων LOG δημιουργήθηκε το interface TaxpayerLogWriter και το factory TaxpayerLogWriterFactory που καλεί ανάλογα το είδος του αρχείου που θέλουμε να γράψουμε την TXT TaxpayerLogWriter ή την XMLTaxpayerLogWriter.

- Για την ανάγνωση και εγγραφή του αρχείου TaxpayersRegistry που περιχέει όλους τους taxpayers, δημιουργήθηκαν 2 κλάσεις , οι TaxpayerRegistryReader και TaxpayerRegistryWriter.

## CLASSES RESPONSIBILITIES AND COLLABORATIONS (CRC CARDS)

- For each class give a brief description in terms of a CRC card (see the format below)

| **Class Name:** `TheApplication` | |
|---|---|
| **Responsibilities** | **Collaborations** |
| ○ Creating the object Controller | ▪ With Controller class |

| **Class Name:** `Controller` | |
|---|---|
| **Responsibilities** | **Collaborations** |
| ○ Creating MainFrame element and handle events(handleLoadTaxpayers(etc) | ▪ MainFrame class<br>▪ Model class |

| **Class Name: MainFrame** | |
|---|---|
| **Responsibilities** | **Collaborations** |
| ○ Create Application window | ▪ Controller class |

| **Class Name : AddReceiptDialog** |
|---|

| Responsibilities | Collaborations |
|---|---|
| ○ Create Pop up window for adding a receipt | ▪ Controller |

| Class Name: AddTaxpayerDialog | |
|---|---|
| **Responsibilities** | **Collaborations** |
| ○ Create Pop up window for adding a taxpayer | ▪ Controller |

| Class Name: SettingsDialog | |
|---|---|
| **Responsibilities** | **Collaborations** |
| ○ Create Pop up window for Settings | ▪ Controller |

| Class Name: Address, Company, Taxpayer, Receipt, Settings. |
|---|

| Responsibilities | Collaborations |
|---|---|
| ▪ Create classes for this elements | ▪ Model |

| Enum Name: MaritalStatus, ReceiptCategory, IOFormat | |
|---|---|
| **Responsibilities** | **Collaborations** |
| • Enumerators for marital status receipt category and IO Format | ▪ Model, View |

| Class Name: Model | |
|---|---|
| **Responsibilities** | **Collaborations** |
| ▪ Has the data and status of the application<br>▪ Calculates basic tax and total tax | ▪ Controller<br>▪ Read and Write INFO classes<br>▪ Read and Write Registry classes<br>▪ Write LOG files classes |

| Class Name: TaxpayerInfoReader, TaxpayerInfoReaderFactory, TXTTaxpayerInfoReader, |
|---|

| XMLTaxpayerInfoReader | |
|---|---|
| **Responsibilities** | **Collaborations** |
| ▪ Read info files that can be in 2 forms, txt or xml, using Factory pattern | ▪ Model |

| Class Name: TaxpayerInfoWriter, TaxpayerInfoWriterFactory, TXTTaxpayerInfoWriter, XMLTaxpayerInfoWriter | |
|---|---|
| **Responsibilities** | **Collaborations** |
| ▪ Write info files that can be in 2 forms, txt or xml, using Factory pattern | ▪ Model |

| Class Name: TaxpayerLogWriter, TaxpayerLogWriterFactory, TXTTaxpayerLogWriter, XMLTaxpayerLogWriter | |
|---|---|
| **Responsibilities** | **Collaborations** |
| ▪ Write log files that can be in 2 forms, txt or xml, using Factory pattern | ▪ Model |

| Class Name: TaxpayerRegisterReader, TaxpayerRegisterWriter |
|---|

| **Responsibilities** | **Collaborations** |
| --- | --- |
| ▪ Read and Write registry file | ▪ Model |