INCOME TAX CALCULATOR

OVERALL REPORT

VERSION <1.0>

SIDIROPOULOS PANAGIOTIS 4489

TSIAPALIS DIMITRIS 4511

TABLE OF CONTENTS

INTRODUCTION

The goal of this project is to re-engineer a Java application. At a glance, the application serves for the income tax calculation of Minnesota state citizens. The tax calculation accounts for the marital status of a given citizen, his income, and the amount of money that he has spent, as witnessed by a set of receipts declared along with the income. The legacy application takes as input txt or xml files that contain the necessary data for each citizen. The tax calculation is based on a complex algorithm provided by the Minnesota state. The application further produces graphical representations of the data in terms of bar and pie charts. Finally. the application produces respective output reports in txt or xml.

The objectives of this phase of the project are to simplify the design to be more extensible and maintainable as well as to redesign the graphical interface to make it more user-friendly.

REFACTORED DESIGN

USE CASES

UC1: LoadTaxPayerFromFile	
Use case ID	UC1
Actors	The user of the application
Pre conditions	1. The user has started the application.
Main flow of events	1. The use case starts when the user selects "load taxpayer" 2. The system displays the file explorer 3. The user browses the file explorer and selects the appropriate file 4. The user confirms the action 5. The system loads the taxpayer's info based on that file that.
Post conditions	1. The tax information of the taxpayer has been loaded to the system.

Alternative flow 1	1. A taxpayer already loaded exception is raised by the system 2. The application indicates that the taxpayer from the selected file is already loaded
Post conditions	-
Alternative flow 2	1. At any time the user may leave load taxpayer screen
Post conditions	-

UC2: DisplayTaxpayerInfo	
Use case ID	UC2
Actors	The user of the application
Pre conditions	The system has at least one taxpayer loaded
Main flow of events	1. The use case starts when the user selects a taxpayer from the list 2. The system selects the taxpayer 3. The user selects “view taxpayer info” 4. The system displays the taxpayers info in a new screen
Post conditions	1. The selected taxpayer info screen is shown to the user
Alternative flow 1	1. At any time the user may leave taxpayer info screen
Post conditions	1. -

UC3: AddReceiptToTaxpayer	
Use case ID	UC3
Actors	The user of the application
Pre conditions	The user views the taxpayer info screen
Main flow of events	1. The use case starts when the user selects "add receipt" 2. The system displays a receipt form 3. The user enters to the form the information of the receipt 4. The user confirms the addition of the receipt 5. The system adds the receipt to the list of receipts of the taxpayer and updates the files containing the taxpayer info
Post conditions	1. The receipt has been added to the taxpayer and the taxpayer info files have been updated
Alternative flow 1	1. A wrong receipt input field exception is raised by the system 2. The application indicates that the user has entered wrong info in a field of the receipt form 3. The application displays the input rules of the receipt form
Post conditions	1. The use case continues at step 3 of the main flow
Alternative flow 2	1. At any time the user may leave the receipt form
Post conditions	1. Receipt addition is canceled, no receipt is added to the taxpayer

UC4: DeleteReceiptOfTaxpayer	
Use case ID	UC4
Actors	The user of the application

Pre conditions	The user views the taxpayer info screen
Main flow of events	1. The use case starts when the user selects a receipt and clicks "delete receipt" 2. The system displays a confirmation message 3. The user confirms his action 4. The system deletes the selected receipt and updates the taxpayers info files
Post conditions	1. The selected receipt has been deleted and the taxpayer's info files has been updated
Alternative flow 1	1. At any time the user may close the confirmation screen
Post conditions	2. The deletion is cancelled, no receipt is deleted

UC5: DisplayTaxCharts	
Use case ID	UC5
Actors	The user of the application
Pre conditions	The user views the taxpayer info screen
Main flow of events	1. The use case starts when the user selects "view reports" 2. The system calculates the necessary data for the charts 3. The system displays charts about receipts and taxes
Post conditions	1. The reports are visible to the user

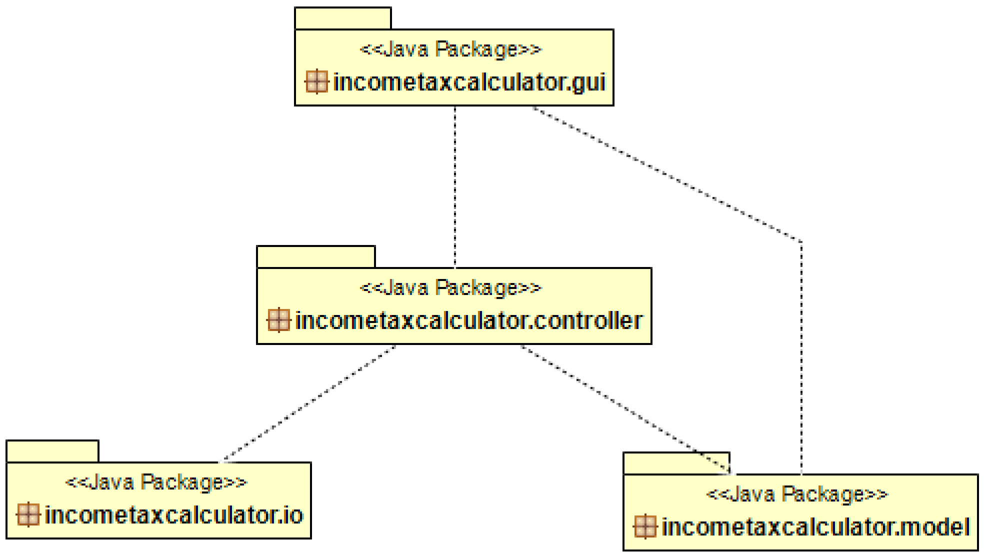
UC6: StoreTaxpayerLog	
Use case ID	UC6
Actors	The user of the application
Pre conditions	The user views the taxpayer info screen
Main flow of events	1. The use case starts when the user selects "save log" 2. The system displays the file explore 3. The user selects the directory that he/she wants the file to be saved at 4. The system displays a message requesting the user to choose the file format from a list 5. The system saves the taxpayers info as a file with the specified format to the specified location 6. The system displays a save confirmation message
Post conditions	1. The taxpayers log is stored to the specified location with the specified format
Alternative flow 1	1. The user closes the file explorer
Post conditions	1. The log file is not saved
Alternative flow 2	2. The user closes the file format selection window
Post conditions	1. The log file is not saved

UC7: RemoveTaxapayer	
Use case ID	UC7

Actors	The user of the application
Pre conditions	1. Taxpayers are loaded into the system
Main flow of events	1. The use case starts when the user selects a taxpayer form the list and clicks “remove taxpayer” 2. The system displays a confirmation message 3. The system removes the selected taxpayer
Post conditions	1. The selected taxpayer is removed from the loaded taxpayer list

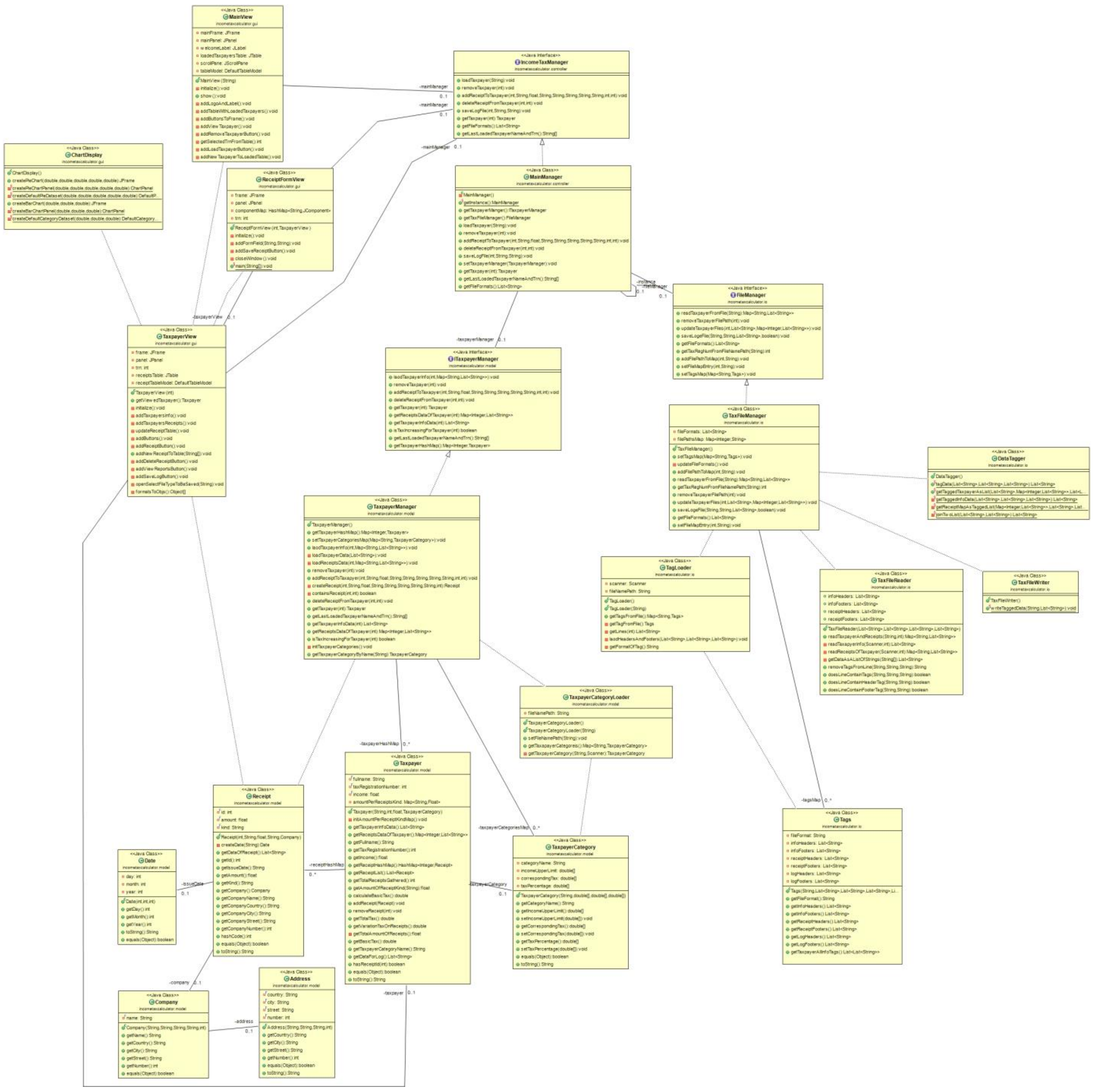
ARCHITECTURE

PACKAGE UML DIAGRAM:

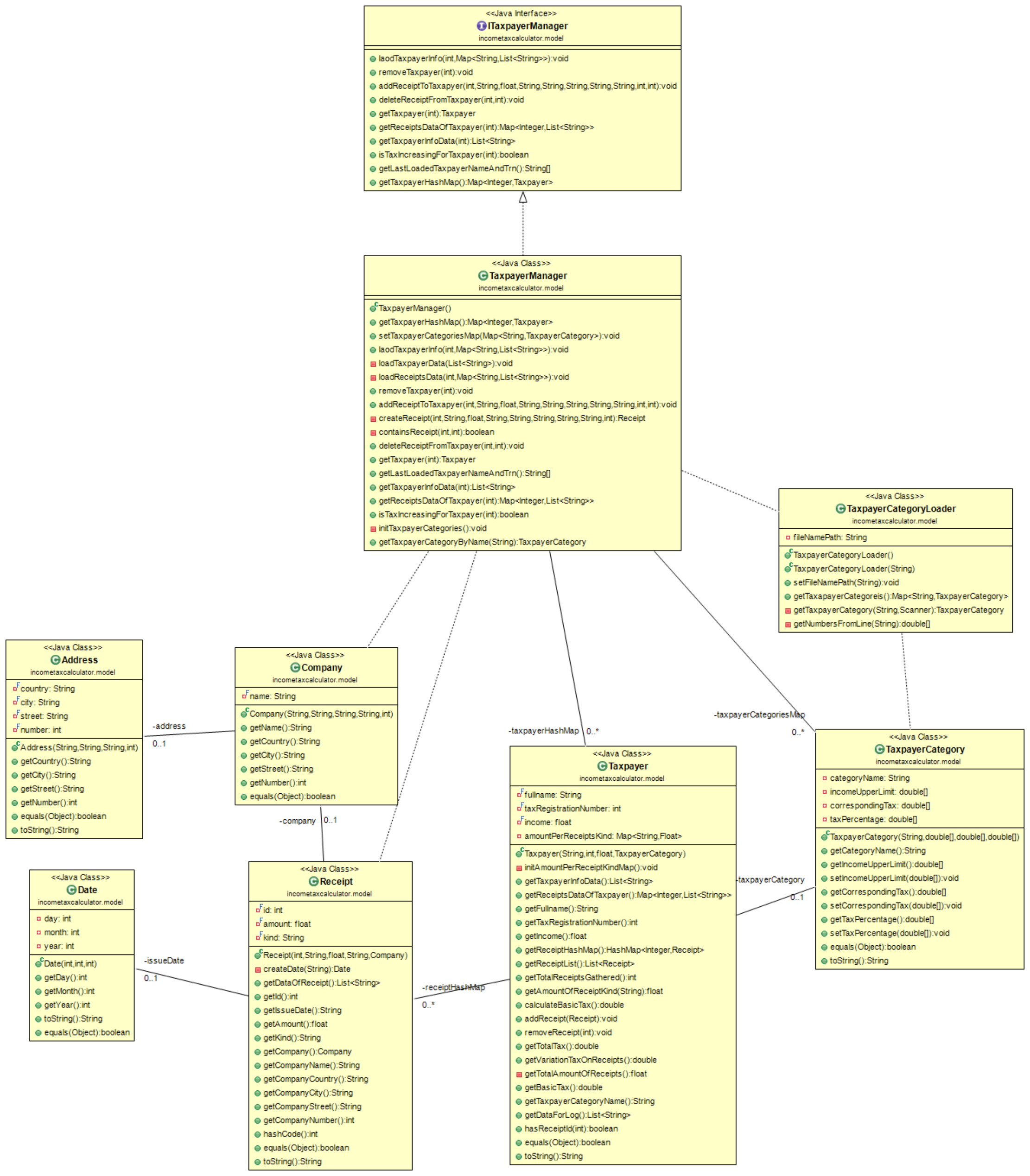


DETAILED DESIGN

Project UML Class Diagram:

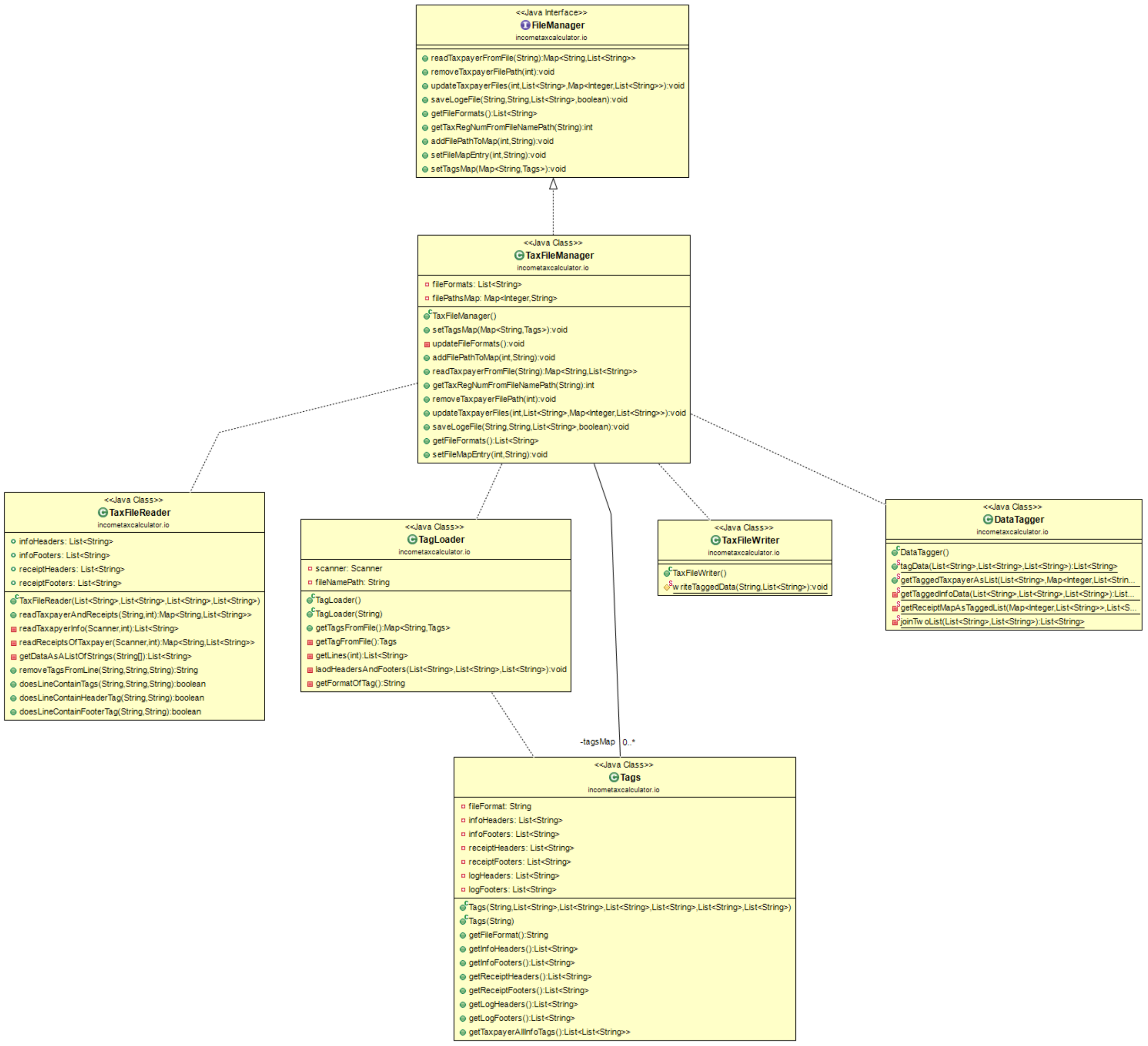


MODEL UML DIAGRAM:

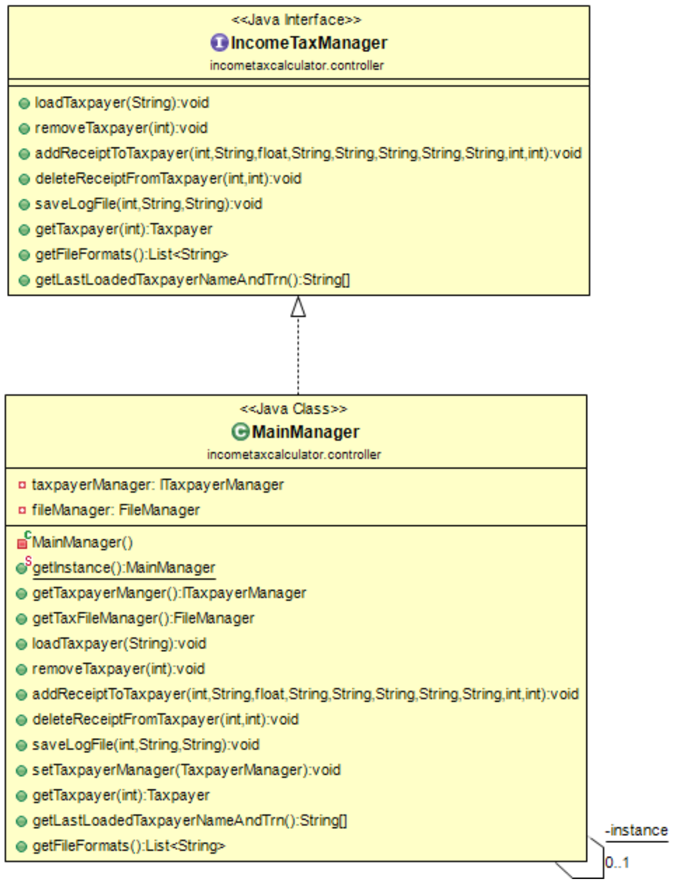


<<Java Interface>>
ITaxpayerManager
incometaxcalculator.model
laodTaxpayerInfo(int,Map<String,List<String>>):void
removeTaxpayer(int):void
addReceiptToTaxapyer(int,String,float,String,String,String,String,String,int,int):void
deleteReceiptFromTaxpayer(int,int):void
getTaxpayer(int):Taxpayer
getReceiptsDataOfTaxpayer(int):Map<Integer,List<String>>
getTaxpayerInfoData(int):List<String>
isTaxIncreasingForTaxpayer(int):boolean
getLastLoadedTaxpayerNameAndTrn():String[]
getTaxpayerHashMap():Map<Integer,Taxpayer>
<<Java Class>>
TaxpayerManager
incometaxcalculator.model
TaxpayerManager()
getTaxpayerHashMap():Map<Integer,Taxpayer>
setTaxpayerCategoriesMap(Map<String,TaxpayerCategory>):void
laodTaxpayerInfo(int,Map<String,List<String>>):void
loadTaxpayerData(List<String>):void
loadReceiptsData(int,Map<String,List<String>>):void
removeTaxpayer(int):void
addReceiptToTaxapyer(int,String,float,String,String,String,String,String,int,int):void
createReceipt(int,String,float,String,String,String,String,String,int):Receipt
containsReceipt(int,int):boolean
deleteReceiptFromTaxpayer(int,int):void
getTaxpayer(int):Taxpayer
getLastLoadedTaxpayerNameAndTrn():String[]
getTaxpayerInfoData(int):List<String>
getReceiptsDataOfTaxpayer(int):Map<Integer,List<String>>
isTaxIncreasingForTaxpayer(int):boolean
initTaxpayerCategories():void
getTaxpayerCategoryByName(String):TaxpayerCategory
<<Java Class>>
TaxpayerCategoryLoader
incometaxcalculator.model
fileNamePath: String
TaxpayerCategoryLoader()
TaxpayerCategoryLoader(String)
setFileNamePath(String):void
getTaxapayerCategoreis():Map<String,TaxpayerCategory>
getTaxpayerCategory(String,Scanner):TaxpayerCategory
getNumbersFromLine(String):double[]
<<Java Class>>
Address
incometaxcalculator.model
country: String
city: String
street: String
number: int
Address(String,String,String,int)
getCountry():String
getCity():String
getStreet():String
getNumber():int
equals(Object):boolean
toString():String
-address
0..1
<<Java Class>>
Company
incometaxcalculator.model
name: String
Company(String,String,String,String,int)
getName():String
getCountry():String
getCity():String
getStreet():String
getNumber():int
equals(Object):boolean
-company
0..1
-taxpayerHashMap
0..*
-taxpayerCategoriesMap
0..*
<<Java Class>>
Taxpayer
incometaxcalculator.model
fullname: String
taxRegistrationNumber: int
income: float
amountPerReceiptsKind: Map<String,Float>
Taxpayer(String,int,float,TaxpayerCategory)
initAmountPerReceiptKindMap():void
getTaxpayerInfoData():List<String>
getReceiptsDataOfTaxpayer():Map<Integer,List<String>>
getFullname():String
getTaxRegistrationNumber():int
getIncome():float
getReceiptHashMap():HashMap<Integer,Receipt>
getReceiptList():List<Receipt>
getTotalReceiptsGathered():int
getAmountOfReceiptKind(String):float
calculateBasicTax():double
addReceipt(Receipt):void
removeReceipt(int):void
getTotalTax():double
getVariationTaxOnReceipts():double
getTotalAmountOfReceipts():float
getBasicTax():double
getTaxpayerCategoryName():String
getDataForLog():List<String>
hasReceiptId(int):boolean
equals(Object):boolean
toString():String
<<Java Class>>
TaxpayerCategory
incometaxcalculator.model
categoryName: String
incomeUpperLimit: double[]
correspondingTax: double[]
taxPercentage: double[]
TaxpayerCategory(String,double[],double[],double[])
getCategoryName():String
getIncomeUpperLimit():double[]
setIncomeUpperLimit(double[]):void
getCorrespondingTax():double[]
setCorrespondingTax(double[]):void
getTaxPercentage():double[]
setTaxPercentage(double[]):void
equals(Object):boolean
toString():String
-taxpayerCategory
0..1
<<Java Class>>
Date
incometaxcalculator.model
day: int
month: int
year: int
Date(int,int,int)
getDay():int
getMonth():int
getYear():int
toString():String
equals(Object):boolean
-issueDate
0..1
<<Java Class>>
Receipt
incometaxcalculator.model
id: int
amount: float
kind: String
Receipt(int,String,float,String,Company)
createDate(String):Date
getDataOfReceipt():List<String>
getId():int
getIssueDate():String
getAmount():float
getKind():String
getCompany():Company
getCompanyName():String
getCompanyCountry():String
getCompanyCity():String
getCompanyStreet():String
getCompanyNumber():int
hashCode():int
equals(Object):boolean
toString():String
-receiptHashMap
0..*

IO UML CLASS DIAGRAM:

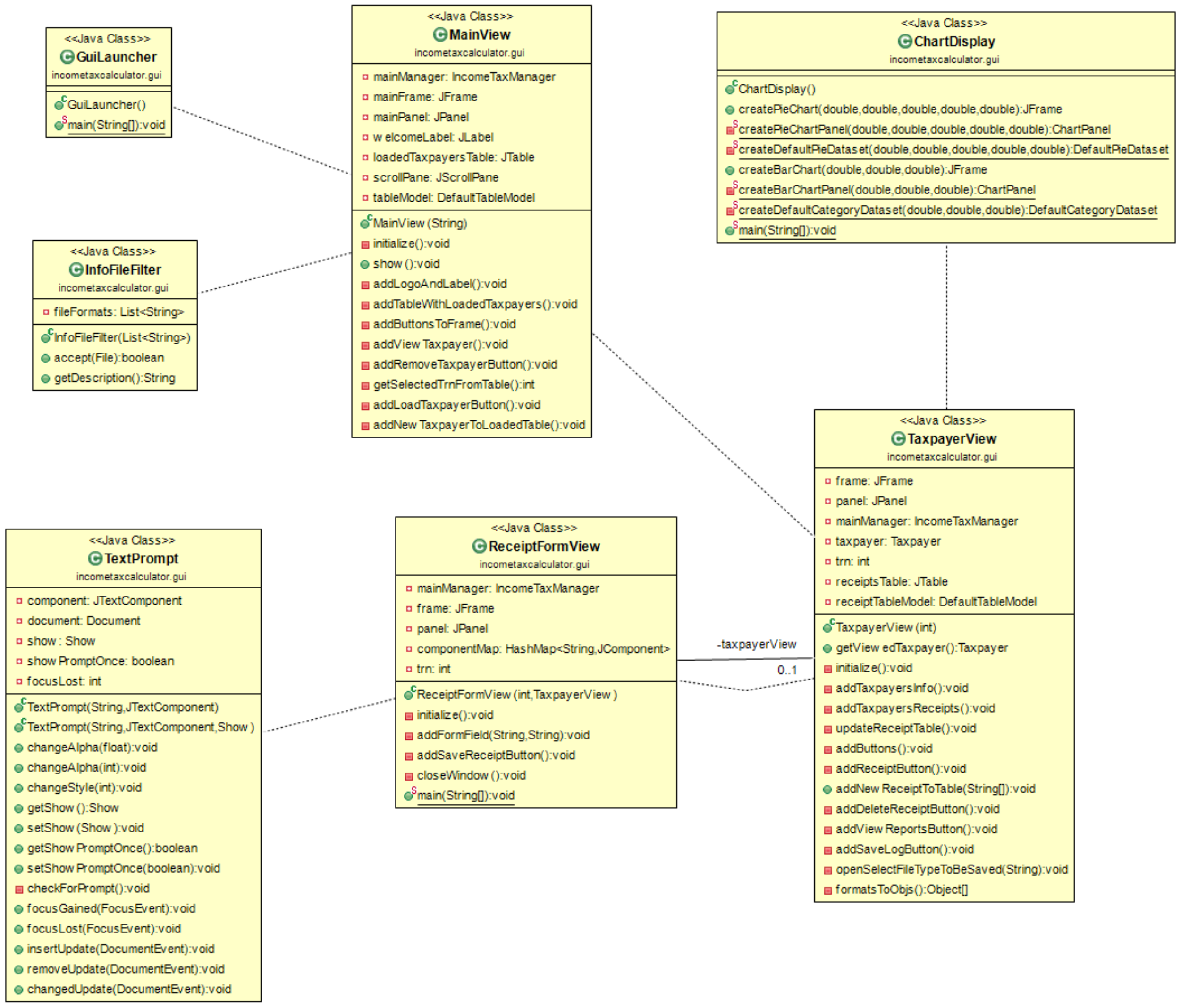

<<Java Interface>>
FileManager
incometaxcalculator.io
readTaxpayerFromFile(String):Map<String,List<String>>
removeTaxpayerFilePath(int):void
updateTaxpayerFiles(int,List<String>,Map<Integer,List<String>>):void
saveLogeFile(String,String,List<String>,boolean):void
getFileFormats():List<String>
getTaxRegNumFromFileNamePath(String):int
addFilePathToMap(int,String):void
setFileMapEntry(int,String):void
setTagsMap(Map<String,Tags>):void
<<Java Class>>
TaxFileManager
incometaxcalculator.io
fileFormats: List<String>
filePathsMap: Map<Integer,String>
TaxFileManager()
setTagsMap(Map<String,Tags>):void
updateFileFormats():void
addFilePathToMap(int,String):void
readTaxpayerFromFile(String):Map<String,List<String>>
getTaxRegNumFromFileNamePath(String):int
removeTaxpayerFilePath(int):void
updateTaxpayerFiles(int,List<String>,Map<Integer,List<String>>):void
saveLogeFile(String,String,List<String>,boolean):void
getFileFormats():List<String>
setFileMapEntry(int,String):void
<<Java Class>>
TaxFileReader
incometaxcalculator.io
infoHeaders: List<String>
infoFooters: List<String>
receiptHeaders: List<String>
receiptFooters: List<String>
TaxFileReader(List<String>,List<String>,List<String>,List<String>)
readTaxpayerAndReceipts(String,int):Map<String,List<String>>
readTaxapyerInfo(Scanner,int):List<String>
readReceiptsOfTaxpayer(Scanner,int):Map<String,List<String>>
getDataAsAListOfStrings(String[]):List<String>
removeTagsFromLine(String,String,String):String
doesLineContainTags(String,String,String):boolean
doesLineContainHeaderTag(String,String):boolean
doesLineContainFooterTag(String,String):boolean
<<Java Class>>
TagLoader
incometaxcalculator.io
scanner: Scanner
fileNamePath: String
TagLoader()
TagLoader(String)
getTagsFromFile():Map<String,Tags>
getTagFromFile():Tags
getLines(int):List<String>
laodHeadersAndFooters(List<String>,List<String>,List<String>):void
getFormatOfTag():String
<<Java Class>>
TaxFileWriter
incometaxcalculator.io
TaxFileWriter()
writeTaggedData(String,List<String>):void
<<Java Class>>
DataTagger
incometaxcalculator.io
DataTagger()
tagData(List<String>,List<String>,List<String>):List<String>
getTaggedTaxpayerAsList(List<String>,Map<Integer,List<Strin...
getTaggedInfoData(List<String>,List<String>,List<String>):List...
getReceiptMapAsTaggedList(Map<Integer,List<String>>,List<S...
joinTwoList(List<String>,List<String>):List<String>
-tagsMap
0..*
<<Java Class>>
Tags
incometaxcalculator.io
fileFormat: String
infoHeaders: List<String>
infoFooters: List<String>
receiptHeaders: List<String>
receiptFooters: List<String>
logHeaders: List<String>
logFooters: List<String>
Tags(String,List<String>,List<String>,List<String>,List<String>,List<String>,List<String>)
Tags(String)
getFileFormat():String
getInfoHeaders():List<String>
getInfoFooters():List<String>
getReceiptHeaders():List<String>
getReceiptFooters():List<String>
getLogHeaders():List<String>
getLogFooters():List<String>
getTaxpayerAllInfoTags():List<List<String>>

CONTROLLER UML CLASS DIAGRAM:



<<Java Interface>>
IncomeTaxManager
incometaxcalculator.controller
loadTaxpayer(String):void
removeTaxpayer(int):void
addReceiptToTaxpayer(int,String,float,String,String,String,String,String,int,int):void
deleteReceiptFromTaxpayer(int,int):void
saveLogFile(int,String,String):void
getTaxpayer(int):Taxpayer
getFileFormats():List<String>
getLastLoadedTaxpayerNameAndTrn():String[]
<<Java Class>>
MainManager
incometaxcalculator.controller
taxpayerManager: ITaxpayerManager
fileManager: FileManager
MainManager()
getInstance():MainManager
getTaxpayerManger():ITaxpayerManager
getTaxFileManager():FileManager
loadTaxpayer(String):void
removeTaxpayer(int):void
addReceiptToTaxpayer(int,String,float,String,String,String,String,String,int,int):void
deleteReceiptFromTaxpayer(int,int):void
saveLogFile(int,String,String):void
setTaxpayerManager(TaxpayerManager):void
getTaxpayer(int):Taxpayer
getLastLoadedTaxpayerNameAndTrn():String[]
getFileFormats():List<String>
-instance
0..1

GUI UML CLASS DIAGRAM:



<<Java Class>>
GuiLauncher
incometaxcalculator.gui
GuiLauncher()
main(String[]):void
<<Java Class>>
MainView
incometaxcalculator.gui
mainManager: IncomeTaxManager
mainFrame: JFrame
mainPanel: JPanel
welcomeLabel: JLabel
loadedTaxpayersTable: JTable
scrollPane: JScrollPane
tableModel: DefaultTableModel
MainView(String)
initialize():void
show():void
addLogoAndLabel():void
addTableWithLoadedTaxpayers():void
addButtonsToFrame():void
addViewTaxpayer():void
addRemoveTaxpayerButton():void
getSelectedTrnFromTable():int
addLoadTaxpayerButton():void
addNewTaxpayerToLoadedTable():void
<<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
main(String[]):void
<<Java Class>>
InfoFileFilter
incometaxcalculator.gui
fileFormats: List<String>
InfoFileFilter(List<String>)
accept(File):boolean
getDescription():String
<<Java Class>>
TaxpayerView
incometaxcalculator.gui
frame: JFrame
panel: JPanel
mainManager: IncomeTaxManager
taxpayer: Taxpayer
trn: int
receiptsTable: JTable
receiptTableModel: DefaultTableModel
TaxpayerView(int)
getViewedTaxpayer():Taxpayer
initialize():void
addTaxpayersInfo():void
addTaxpayersReceipts():void
updateReceiptTable():void
addButtons():void
addReceiptButton():void
addNewReceiptToTable(String[]):void
addDeleteReceiptButton():void
addViewReportsButton():void
addSaveLogButton():void
openSelectFileTypeToBeSaved(String):void
formatsToObjs():Object[]
<<Java Class>>
TextPrompt
incometaxcalculator.gui
component: JTextComponent
document: Document
show: Show
showPromptOnce: boolean
focusLost: int
TextPrompt(String,JTextComponent)
TextPrompt(String,JTextComponent,Show)
changeAlpha(float):void
changeAlpha(int):void
changeStyle(int):void
getShow():Show
setShow(Show):void
getShowPromptOnce():boolean
setShowPromptOnce(boolean):void
checkForPrompt():void
focusGained(FocusEvent):void
focusLost(FocusEvent):void
insertUpdate(DocumentEvent):void
removeUpdate(DocumentEvent):void
changedUpdate(DocumentEvent):void
<<Java Class>>
ReceiptFormView
incometaxcalculator.gui
mainManager: IncomeTaxManager
frame: JFrame
panel: JPanel
componentMap: HashMap<String,JComponent>
trn: int
ReceiptFormView(int,TaxpayerView)
initialize():void
addFormField(String,String):void
addSaveReceiptButton():void
closeWindow():void
main(String[]):void
-taxpayerView
0..1

SUMMARY OF CHANGES

1. Separation of responsibilities of TaxpayerManager into two subsystems the TaxFileManger class responsible for reading and writing data and the class with the same name (TaxpayerManger) for managing taxpayers. The MainManager class was also used as a façade, which orchestrates the implementation of use cases using these two subsystems.

2. Used the “\resources\taxpayerProperties.txt” file to load taxpayer class constants to eliminate Taxpayer subclasses.

3. Used the “\resources\tagsProperties.txt” file to load tags for different file types to eliminate the hierarchy of reader and writer classes, greater extensibility, and adding the omitted check that each field has correct tags.

4. Addition of acceptance tests.

5. Changed the user interface to make it easier to use by providing multiple feedback messages and to enable the user to save and load taxpayer files from any file explorer folder on their computer and in any of the file formats that contain the file with the tags.

6. Class and package name changes and reassignment of classes to packages

CHANGES IN DETAIL:

LARGE CLASS TAXPAYERMANAGER

We split the taxpayerManager into two classes, one that is responsible only for managing taxpayers (we kept the same name TaxpayerManager) and the TaxFileManager that is responsible for managing files (reading taxpayers from a file, writing log and info files). So the new TaxpayerManager class takes the taxpayer data that has been read and loads it into taxpayer objects. Adding receipts to the taxpayer is done through a method of the taxpayer class.

TAXPAYER CHANGES

First, we used some arrays in the Taxpayer base class that will have constants that change depending on the different type of taxpayer. So we ended up only having constants in the subclasses, so we decided to allow the user to load the constants for the class dynamically via a file (taxpayerProperties.txt). We added a TaxpayerCategory class which will contain the name of the taxpayer category and arrays of constants for that category. This class replaced the string status field of the taxpayer and thus having only the basic class we calculate the tax according to the category of the taxpayer and the user himself can very easily add new types of taxpayers. Also the complex code with chained if-else statements was replaced by a for loop with an internal check based on some data we stored as arrays.

DUPLICATE CODE IN INFO & LOG WRITER CLASSES:

Initially we used separate template methods and factories to reduce duplicate code in these classes and to isolate the creation of their subclasses. We ended up with the subclasses based on the file type containing only constants which are the corresponding tags. So we considered that conceptually the writer subclasses should not contain only constants so we decided to give the user the ability to load tags dynamically from a file (tagsProperties.txt) when starting the application. This way we avoid large class hierarchies and leave only one TaxFileWriter class which only writes data to files. Using the utility DataTagger class we add the tags to the data and then through the TaxFileWriter we write the tagged data to the files needed for INFO and LOG and possibly for other types of files that will result from future expansion.

DUPLICATE CODE IN FILEREADER CLASSES

We worked similarly as for the Writers, with the difference that the remaining reader class (TaxFileReader) contains the methods responsible for removing the tags present in the Info files.

OTHER CHANGES:

We used the TagLoader class to load the tags from tagsProperties.txt file as mentioned above. Possibly we could have some code reuse with TaxFileReader but due to limited time it was not investigated further. Renamed several classes and methods and restructured the packages names and structure.

CLASSES RESPONSIBILITIES AND COLLABORATIONS (CRC CARDS)

Class Name: MainManager	
Responsibilities	**Collaborations**
▪ Delegates user actions from gui to the respective subsystem	▪ TaxFile manager ▪ TaxpayerManger ▪ GUI Classes

Class Name: TaxpayerManger	
Responsibilities	**Collaborations**
▪ Loads taxpayer objects and adds them to loaded taxpayers ▪ Adds receipts to a specific taxpayer ▪ Deletes receipt of a taxpayer ▪ Removes taxpayers objects from loaded taxpayers ▪ Loads taxpayer categories from file	▪ Taxpayer ▪ Receipt ▪ TaxpayerCategoryLoader ▪ TaxpayerCategory ▪ Company

Class Name: Taxpayer	
Responsibilities	**Collaborations**
▪ Holds data of taxpayer ▪ Calculates the taxes of taxpayer ▪ Adds receipts from taxpayer ▪ Deletes receipt from taxpayer	▪ TaxpayerManger ▪ Receipt ▪ TaxpayerCategory ▪ TaxpayerView

Class Name: TaxpayerCategoyLoader	
Responsibilities	**Collaborations**
▪ Loads TaxpayerCategory objects from a file	▪ TaxpayerCategory ▪ TaxpayerManager

Class Name: TaxpayerCategory	
Responsibilities	**Collaborations**
▪ Holds data for the tax calculation of a taxpayer category	▪ Taxpayer ▪ TaxpayerManager

Class Name: Date	
Responsibilities	**Collaborations**
▪ Holds data of the issue date of a Receipt	▪ Receipt

Class Name: Company	
Responsibilities	**Collaborations**
▪ Holds data for the company issued a receipt	▪ Receipt

Class Name: Address	
Responsibilities	**Collaborations**
▪ Holds data of the address of the company	▪ Company

Class Name: Receipt	
Responsibilities	**Collaborations**

▪ Holds the data of a receipt	▪ Taxpayer ▪ Company ▪ Date ▪ TaxpayerManager ▪ TaxpayerView

Class Name: TaxFileManager	
Responsibilities	**Collaborations**
▪ Manages objects responsibles for: ○ Loading tags from file ○ Reading taxpayer info file ○ Updating taxpayer info files ○ Saving log file	▪ MainManager ▪ TagLoader ▪ TaxFileReader ▪ DataTagger ▪ TaxFileWriter

Class Name: TagLoader	
Responsibilities	**Collaborations**

▪ Loads tags from file	▪ Tag ▪ TaxFileManager

Class Name: TaxFileReader	
Responsibilities	**Collaborations**
▪ Reads taxpayer info file	▪ TaxFileManager

Class Name: TaxFileWriter	
Responsibilities	**Collaborations**
▪ Writes data to file	▪ TaxFileManager

Class Name: DataTagger	
Responsibilities	**Collaborations**
▪ Adds tags to data and returns tagged data	▪ TaxFileManager

Class Name: MainView

Responsibilities	Collaborations
▪ Displays main windows of the app containing the	▪ MainManager ▪ TaxpayerView

Class Name: TaxpayerView	
Responsibilities	**Collaborations**
▪ Displays taxpayer info and receipts	▪ MainView ▪ TaxpayerView ▪ ReceiptFormView ▪ Taxpayer ▪ Receipt

Class Name: ChartDisplay	
Responsibilities	**Collaborations**
▪ Creates and displays the charts	▪ TaxpayerView

Class Name: ReceiptFormView	
Responsibilities	**Collaborations**
▪ Displays a receipt form used for the addition of new receipt	▪ MainManager ▪ TaxpayerView