<INCOME TAX CALCULATOR>

OVERALL REPORT

VERSION <1.0>

<Bantouvakis Georgios 4119>

TABLE OF CONTENTS

INTRODUCTION

The purpose of this project was to refactor the Income Tax Calculator Application that have been given to us. I have implemented all the refactor tasks, some improvement on the GUI and unit test that cover all the use cases of the project.

The improvements on the gui include the ability to select a taxpayer who is loaded by double-click on his AFM and the ability to delete a taxpayer who is loaded by click on his AFM and press the Delete button from the keyboard.

Also, I have created same dummy files with AFM 111111111.txt and 111111111.xml to be used for the unit tests.

REFACTORED DESIGN

USE CASES

Use case ID	LoadTaxPayer
Actors	User
Pre conditions	A file with format AFM_INFO.txt or AFM_INFO.xml should exist.
Main flow of events	1. The use case starts when the user chooses to load the file of the taxpayer 2. The program loads the file and read the data of the payer.
Alternative flow 1	In case that the file does not exist, or the format is not correct, the program should throw an exception with the right error message.
Post conditions	The data are loaded.

Use case ID	TaxPayerInfoDisplay
Actors	User
Pre conditions	Taxpayer data are
Main flow of events	1. The use case starts when the user selects the taxpayer that wants from the list of all taxpayers. 2. All tax info of taxpayer are displayed to the user.
Alternative flow 1	If the taxpayer is not loaded the program should throw an exception with the right error message.
Post conditions	Info of taxpayer are displayed.

Use case ID	Add Receipt
Actors	User
Pre conditions	Taxpayer info are loaded and displayed
Main flow of events	1. The use case starts when the user selects to add receipt to taxpayer. 2. A pop up window ask for information with correct format so as the receipt added to the payer 3. After the receipt is added, the txt file that contains all user's info should be updated
Alternative flow 1	If the format of user's input is not correct the program should throw an exception with the right error message.
Post conditions	The receipt is added, and the file is update

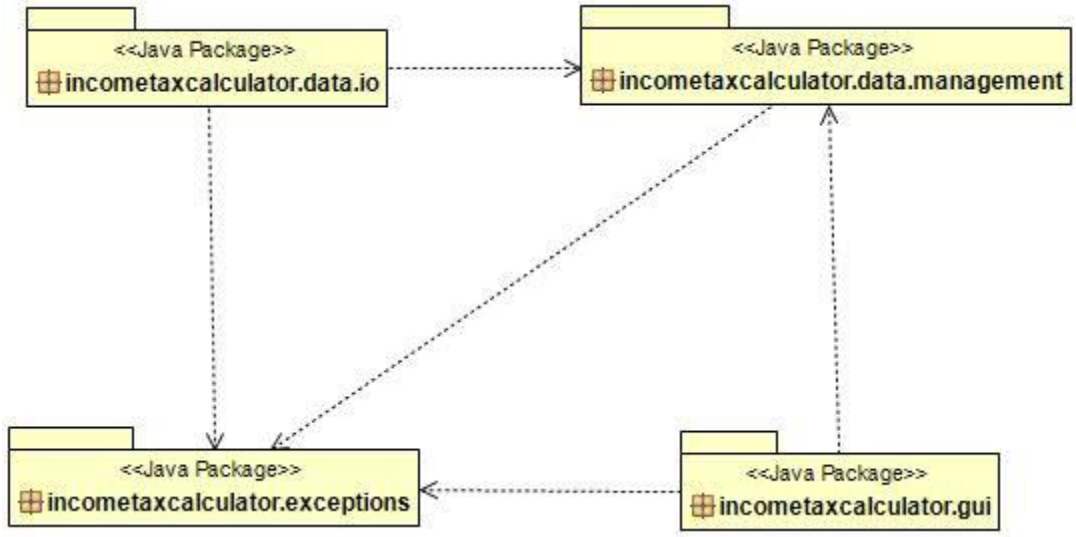
Use case ID	DeleteReceipt
Actors	User
Pre conditions	Taxpayer info are loaded and displayed
Main flow of events	1. The use case starts when the user selects to delete a specific receipt of taxpayer. 2. After the receipt is deleted the txt file that contains all user's info should be updated
Alternative flow 1	If the receipt does not exist, the program should throw an exception with the right error message.
Post conditions	The receipt is deleted, and the file is updated.

Use case ID	DisplayReport
Actors	User
Pre conditions	Taxpayer info are loaded and displayed
Main flow of events	1. The use case starts when the user selects to see the report of the taxpayer. 2. A pop up window shown up with 2 graphs: 2.1. A bar chart with the tax that payer should pay and detailed tax report (basic tax and the impact of the receipts) 2.2 A pie chart with analysis about the different types of receipts.
Post conditions	The 2 types of charts are displayed.

Use case ID	SaveTaxPayerInfo
Actors	User
Pre conditions	Taxpayers' info is loaded and displayed
Main flow of events	1. The use case starts when the user selects to save taxpayer's info. 2. The user should select in what form the data will be save , in a txt file or an xml file 3. A file with format AFM_LOG.txt or AFM_LOG.xml is created with users' info
Post conditions	The file is created.

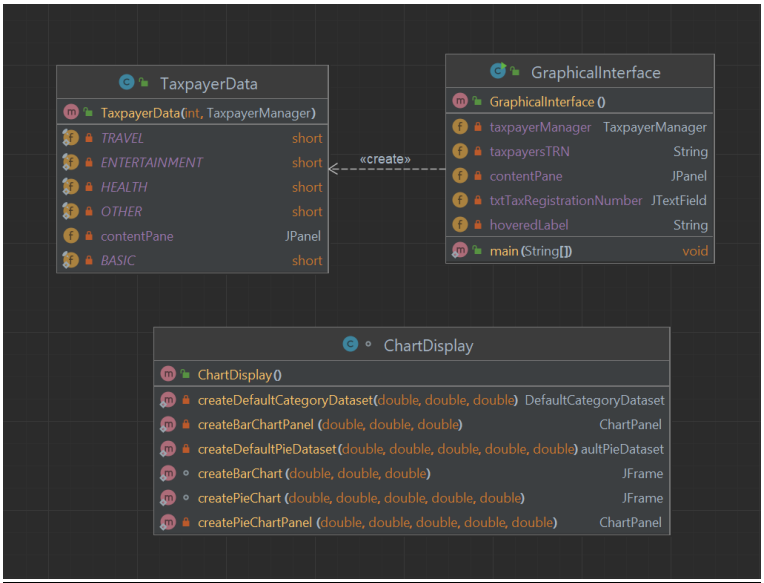
Use case ID	Delete Taxpayer
Actors	User
Pre conditions	Taxpayers are loaded and displayed in a list
Main flow of events	1. The use case starts when the user selects to delete a Taxpayer from the list of taxpayers 2. A pop up window show up that asks from user to give payer's AFM.
Alternative flow 1	If the payer does not exist, the program should throw an exception with the right error message.
Post conditions	The payer is deleted and removed from the list of payers.

ARCHITECTURE

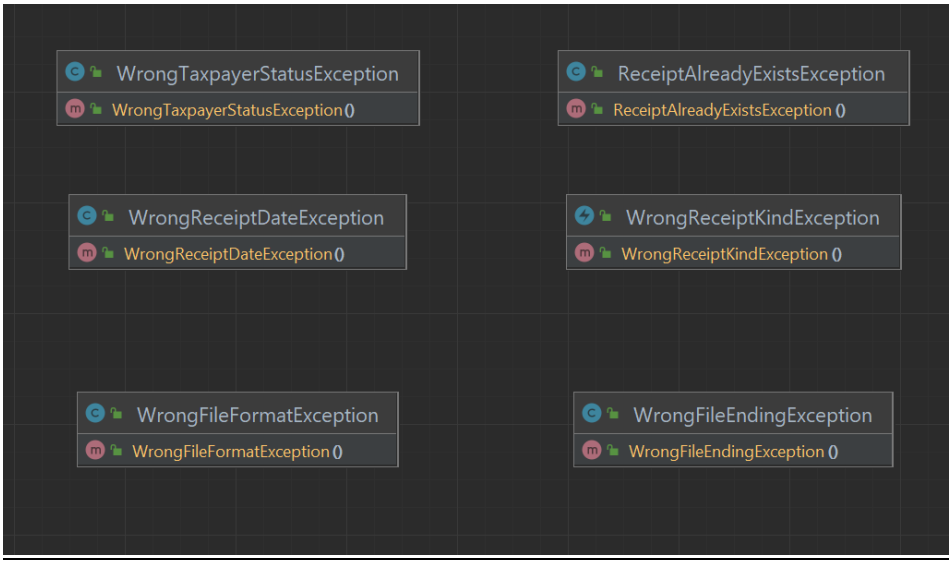


DETAILED DESIGN

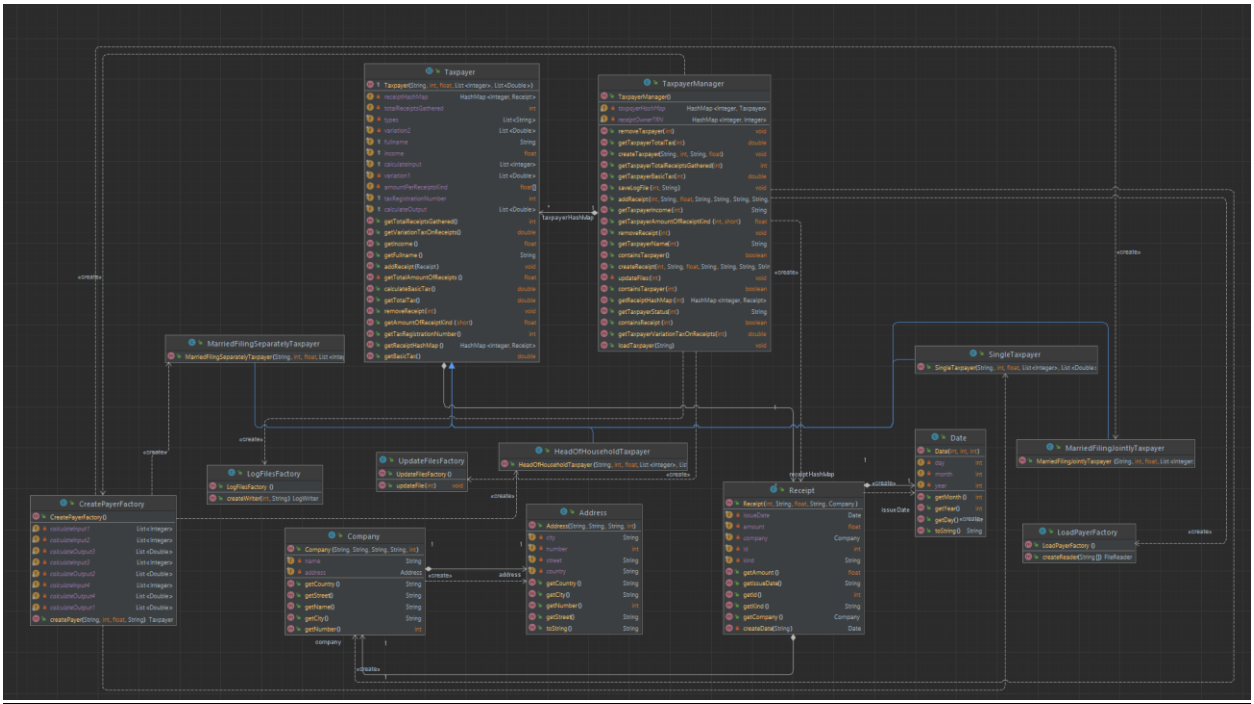
Package: incometaxcalculator.gui



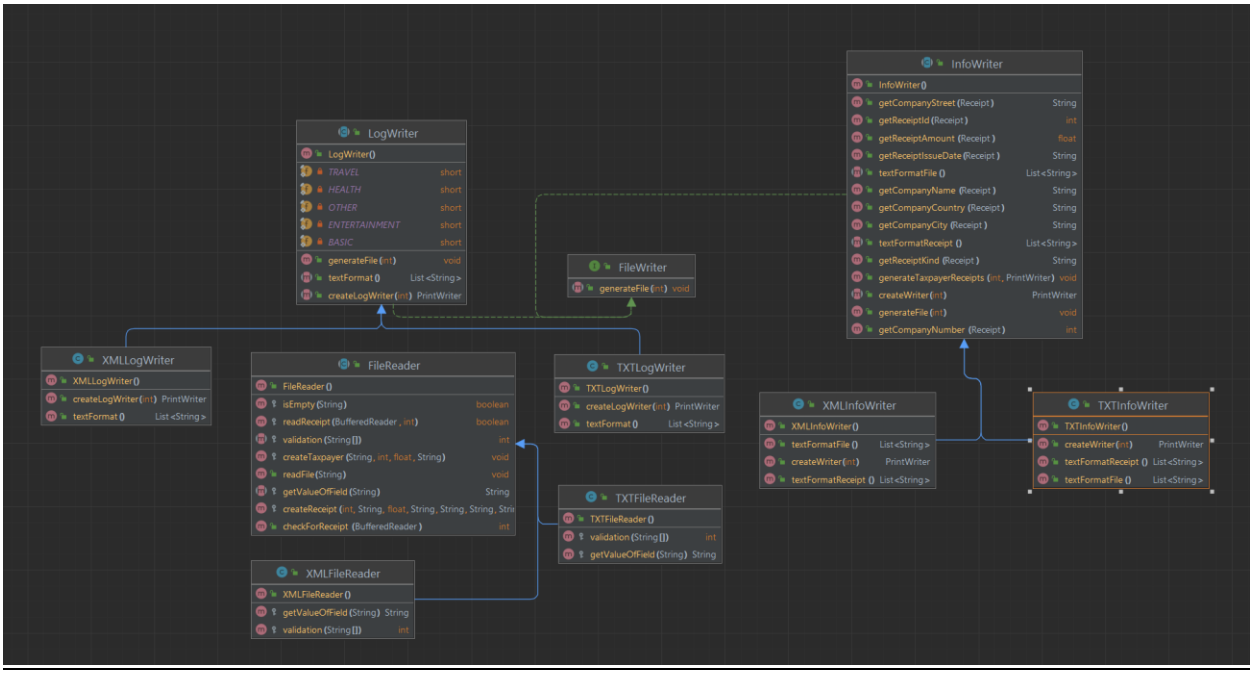
Package: incometaxcalculator.exceptions



Package: incometaxcalculator.data.management



Package: incometaxcalculator.data.io



- The main problems of the old project were:
    - Dead Code: This problem was handled by just delete the methods that were not used
    - Duplicate Code: This problem was solved by creating abstract methods to the top layers and create specific classes that implement only the methods that were necessary
    - Complex logic: For example, complex if – else logic was solved by replacing the code with for loops to be easier to read and extend

- Not small and compact methods: For example, TaxpayerManager class was responsible for too many actions. I removed where it was possible object creation to different files to clean TaxpayerManager and make the project more extendable

CLASSES RESPONSIBILITIES AND COLLABORATIONS (CRC CARDS)

Class Name: FileReader	
Responsibilities	**Collaborations**
▪ This class is an abstract class who is responsible to implement methods that are used to read the files that contains taxpayers' info	▪ This class has 2 abstract methods that are implemented from TXTFileReader and XMLFilerReader.

Interface Name: FileWriter	
Responsibilities	**Collaborations**
▪ This interface is responsible is implemented from other classes to update taxpayer file when needed	▪ This interface is implemented from InfoWriter, LogWriter, TXTInfoWriter, XMLInfoWriter, TXTLogWriter, XMLLogWriter

Class Name: InfoWriter	
Responsibilities	**Collaborations**
▪ This class extends FileWriter interface and is responsible to write the updates that happen to each user to his files.	▪ This class has 2 inheritors TXTInfoWriter and XMLInfoWriter

Class Name: LogWriter	
Responsibilities	**Collaborations**
▪ This class extends FileWriter interface and is responsible to write the log files when user wants to save taxpayer info.	▪ This class has 2 inheritors TXTLogWriter, XMLLogWriter

Class Name: TXTFileReader	
Responsibilities	**Collaborations**
▪ This class extends FileReader class with the appropriate methods when the file that is loaded has txt format	▪ Extends FileReader

Class Name: TXTInfoWriter	
Responsibilities	**Collaborations**
▪ This class extends InfoWriter class with the appropriate methods when the updates that happens to a taxpayers should write to a .txt file	▪ Extends InfoWriter

Class Name: TXTLogWriter	
Responsibilities	**Collaborations**
▪ This class extends LogWriter class with the appropriate methods when the user wants to save the log in a .txt format	▪ Extends LogWriter

Class Name: XMLFileReader	
Responsibilities	**Collaborations**
▪ This class extends FileReader class with the appropriate methods when the file that is loaded has xml format	▪ Extends FileReader

Class Name: XMLInfoWriter	
Responsibilities	**Collaborations**
▪ This class extends InfoWriter class with the appropriate methods when the updates that happens to a taxpayers should write to a .xml file	▪ Extends InfoWriter

Class Name: XMLLogWriter	
Responsibilities	**Collaborations**
▪ This class extends LogWriter class with the appropriate methods when the user wants to save the log in a .xml format	▪ Extends LogWriter

Class Name: Address	
Responsibilities	**Collaborations**
▪ This class is responsible to create Address type objects	▪ Company class uses Address objects

Class Name: Company	
Responsibilities	**Collaborations**
▪ This class is responsible to create Company type objects	▪

Class Name: Company	
Responsibilities	**Collaborations**
▪ This class is responsible to create Company type objects	▪

Class Name: CreatePayerFactory	
Responsibilities	**Collaborations**
▪ This class is responsible to create taxpayer objects.	▪ TaxpayerManager class uses this class

Class Name: Date	
Responsibilities	**Collaborations**
▪ This class is responsible to create Date type objects	▪

Class Name: HeadOfHouseholdTaxpayer	
Responsibilities	**Collaborations**
▪ This class is responsible to taxpayer of HeadOfHouseholdTaxpayer type	▪ Extends Taxpayer class

Class Name: LoadPayerFactory	
Responsibilities	**Collaborations**
▪ This class is responsible to create FilerReader objects.	▪ TaxpayerManager class uses this class

Class Name: LogFilesFactory	
Responsibilities	**Collaborations**
▪ This class is responsible to create LogWriter objects.	▪ TaxpayerManager class uses this class

Class Name: MarriedFilingJointlyTaxpayer	
Responsibilities	**Collaborations**
▪ This class is responsible to taxpayer of MarriedFilingJointlyTaxpayer type	▪ Extends Taxpayer class

Class Name: MarriedFilingSeparatelyTaxpayer	
Responsibilities	**Collaborations**
▪ This class is responsible to taxpayer of MarriedFilingSeparatelyTaxpayer type	▪ Extends Taxpayer class

Class Name: Receipt	
Responsibilities	**Collaborations**
▪ This class is responsible to create Receipt type objects	▪

Class Name: SingleTaxpayer	
Responsibilities	**Collaborations**
▪ This class is responsible to taxpayer of SingleTaxpayer type	▪ Extends Taxpayer class

Class Name: Taxpayer	
Responsibilities	**Collaborations**
▪ This class is responsible to implement all methods to manipulate taxpayer's objects	▪

Class Name: TaxpayerManager	
Responsibilities	**Collaborations**
▪ This class is responsible to implement all methods that connect taxpayer with receipts and taxes	▪

Class Name: UpdateFilesFactory	
Responsibilities	**Collaborations**
▪ This class is responsible to create infoWriter objects.	▪ TaxpayerManager class uses this class

Class Name: ReceiptAlreadyException	
Responsibilities	**Collaborations**
▪ This class is responsible to throw an error with message: Receipt already exists	▪ TaxpayerManager class uses this class

Class Name: WrongFileEndingException	
Responsibilities	**Collaborations**
▪ This class is responsible to throw an error with message: Please check your file ending and try again	▪ TaxpayerManager class uses this class

Class Name: WrongFileFOrmatException	
Responsibilities	**Collaborations**
▪ This class is responsible to throw an error with message: Please check your file format and try again	▪ TaxpayerManager class uses this class

Class Name: WrongReceiptDataException	
Responsibilities	**Collaborations**
▪ This class is responsible to throw an error with message: Please make sure your date is DD/MM/YYYY and try again.	▪ TaxpayerManager class uses this class

Class Name: WrongReceiptKindException	
Responsibilities	**Collaborations**
▪ This class is responsible to throw an error with message: Please check receipts kind and try again	▪ TaxpayerManager class uses this class

Class Name: WrongTaxpayerStatusException	
Responsibilities	**Collaborations**
▪ This class is responsible to throw an error with message: Please check taxpayer's status and try again!	▪ TaxpayerManager class uses this class

Class Name: ChartDisplay	
Responsibilities	**Collaborations**
▪ This class is responsible to create the gui for the 2 types of charts that the app needs	▪ TaxpayerData class uses this class

Class Name: GraphicalInterface	
Responsibilities	**Collaborations**
▪ This class is responsible to create the main gui of the app	▪

Class Name: TaxpayerData	
Responsibilities	**Collaborations**
▪ This class is responsible to create the gui about taxpayers' info	▪