TAX CALCULATION

OVERALL REPORT

FINAL VERSION

Βουρδουγιάννης Δημήτριος 4326

Νικόλαος Παραλίας Ψιλοδημητρακόπουλος 4202

TABLE OF CONTENTS

INTRODUCTION

The goal of this project is to recognize mistakes that were made when the code was written. These mistakes appear in certain patterns that we also learned to recognize. In addition, we understood the point of making a user-friendly application by exploring the application and find out what makes the user frustrated when using this application. So, we fixed every detail that made our life and consequently his life difficult.

REFACTORED DESIGN

USE CASES

1. LOAD TAXPAYER

Use case ID	UC1
Actors	User
Preconditions	The .txt or .xml files had to been created and stored in the filesystem beforehand so the user can load them.
Main flow of events	1. The use case starts when the user presses the "Load Taxpayer" button. 2. The system asks for the tax registration number. 3. The user types the tax registration number. i. If the tax registration number typed exists in the filesystem, the file is loaded and the use case ends. ii. If the tax registration number typed does not exist in the filesystem, the system prints an error message and the use case ends. iii. If the tax registration number is not a 9-digit number the system prints an error message and the use case goes back to step 2.
Alternative flow 1	The user can always return to home page by pressing "Cancel" button or closing the pop-up window.
Post Condition	The file is loaded to the application and the tax registration number is shown at the board.

2. SELECT TAXPAYER

Use case ID	UC2
Actors	User
Preconditions	The tax registration number must be loaded.
Main flow of events	1. The use case starts when the user presses the “Select Taxpayer” button. 2. The system asks for the tax registration number. 3. The user types the tax registration number. i. If the tax registration number is loaded, the specific tax registration number is selected and the use case ends. ii. If the tax registration number is not loaded, an error message is shown and the use case ends.
Alternative flow 1	The user can always return to home page by pressing “Cancel” button or closing the pop-up window.
Post Condition	The tax registration number is selected and a pop-up window is displayed.

3. DELETE TAXPAYER

Use case ID	UC3
Actors	User
Preconditions	A tax registration number must be loaded.
Main flow of events	1. The use case starts when the user presses the "Delete Taxpayer" button. 2. The system asks for a tax registration number. 3. The user types the tax registration number. i. If the tax registration number exists in the database, it is deleted and the use cases ends. ii. If the tax registration number does not exist in the database, nothing is deleted and the use cases ends.
Alternative flow 1	The user can always return to home page by pressing "Cancel" button or closing the pop-up window.
Post Condition	The tax registration number is deleted from the database and from the tax registration number board.

4. ADD RECEIPT

Use case ID	UC4
Actors	User
Preconditions	A tax registration number must be selected.
Main flow of events	1. The use case starts when the user presses the "Add Receipt" button. 2. The system asks for the receipt's details. 3. The user types the details of the receipt. i. If the details are correct, the receipt is added and the use case ends. ii. If the details are wrong, an error message is shown and the use case ends.
Alternative flow 1	The user can always return to home page by pressing "Cancel" button or closing the pop-up window.
Post Condition	The receipt is added to the database of the selected tax registration number and the receipt is shown at the Receipts board.

5. DELETE RECEIPT

Use case ID	UC5
Actors	User
Preconditions	A tax registration number must be selected.
Main flow of events	1. The use case starts when the user presses the “Delete Receipt” button. 2. The system asks for the receipt’s ID. 3. The user types the ID of the receipt. i. If the ID exists, the receipt is deleted and the use case ends. ii. If the ID does not exist, nothing is deleted and the use case ends.
Alternative flow 1	The user can always return to home page by pressing “Cancel” button or closing the pop-up window.
Post Condition	The receipt is deleted from the database of the selected tax registration number and from the Receipts board.

6. VIEW REPORT

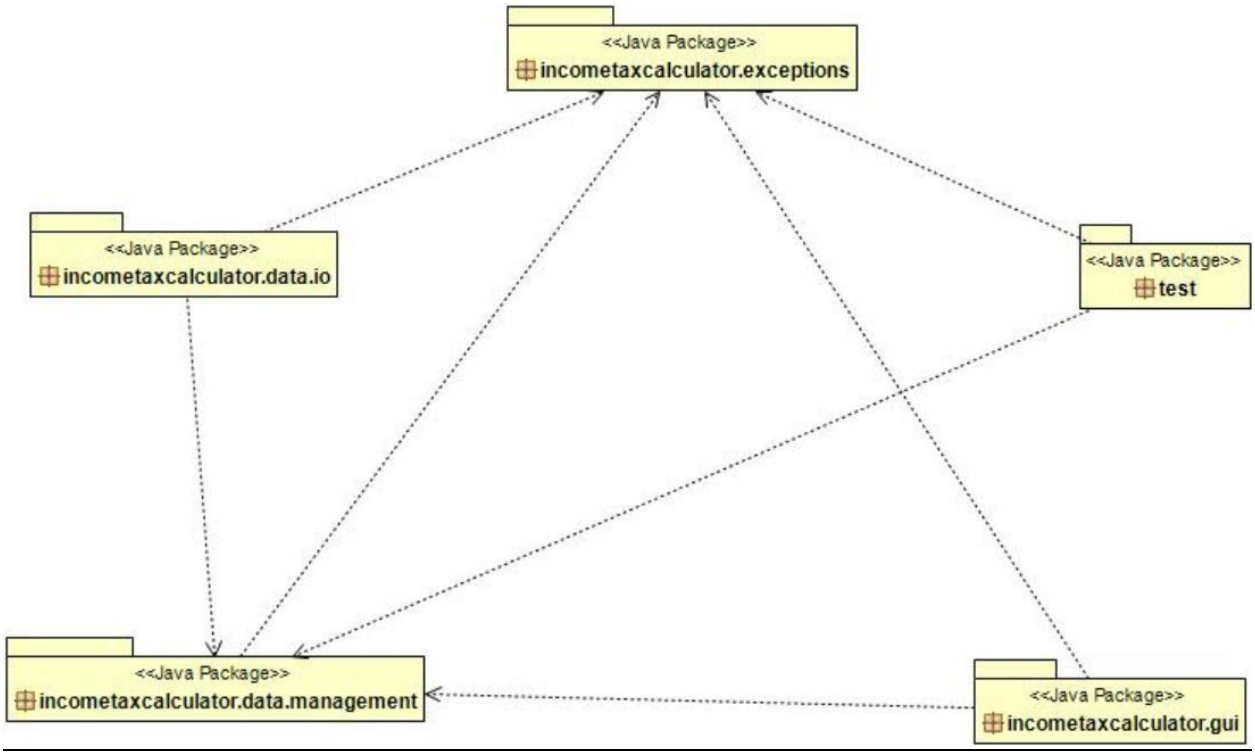
Use case ID	UC6
Actors	User
Preconditions	A tax registration number must be selected.
Main flow of events	1. The use case starts when the user presses the “View Report” button and the use case ends.
Post Condition	The pie-chart and bar-chart are shown in two different pop-up windows.

7. SAVE DATA

Use case ID	UC7
Actors	User
Preconditions	A tax registration number must be selected.
Main flow of events	1. The use case starts when the user presses the “Save Data” button. 2. The system asks the user in which format to save the database of the selected tax registration number. 3. The user selects the format of the database and the use case ends.
Alternative flow 1	The user can always return to home page by pressing “Cancel” button or closing the pop-up window.
Post Condition	If the database exists in the specified format, the system updates it. Otherwise, a new database is created in the specified format.

ARCHITECTURE

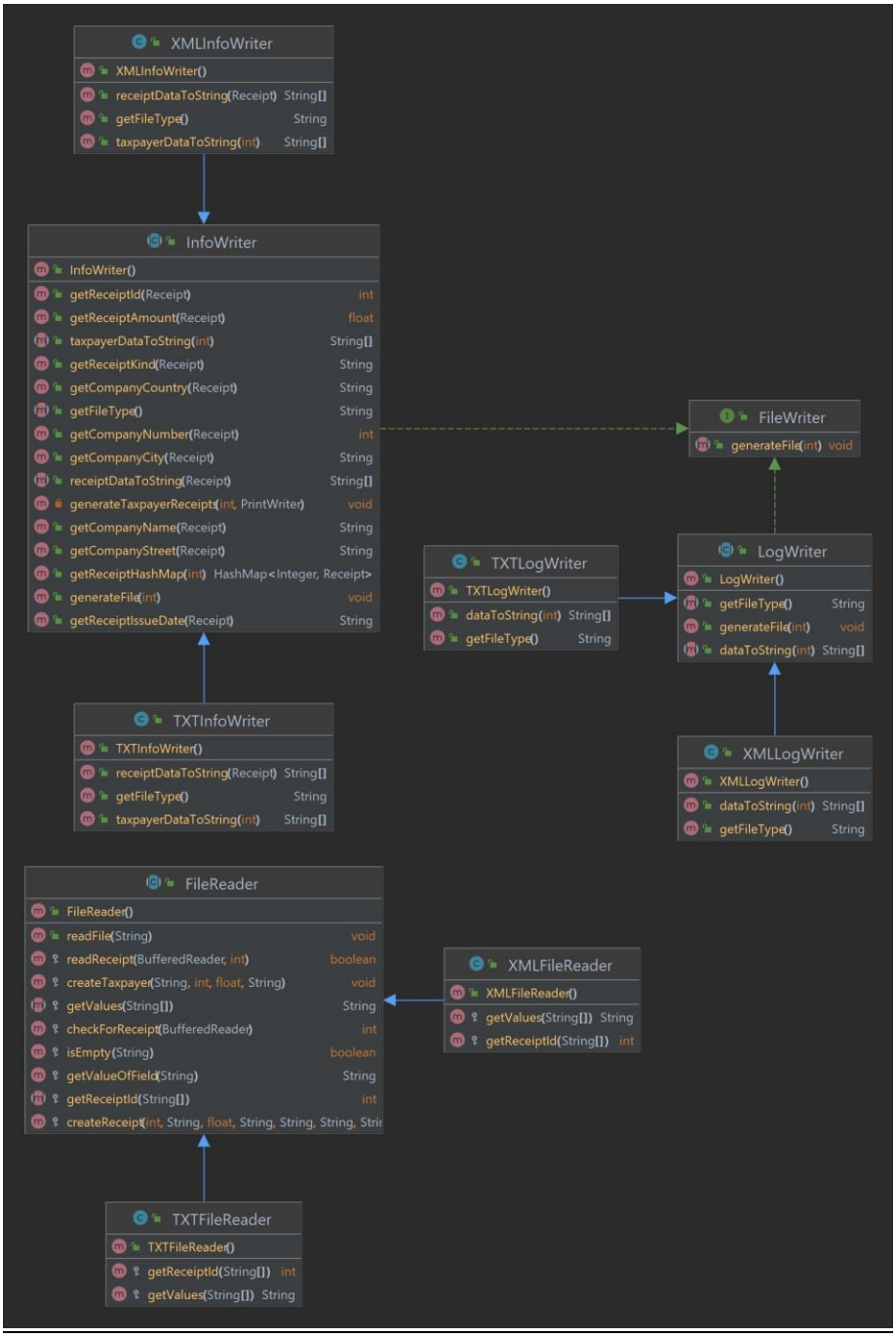
- Package Diagram



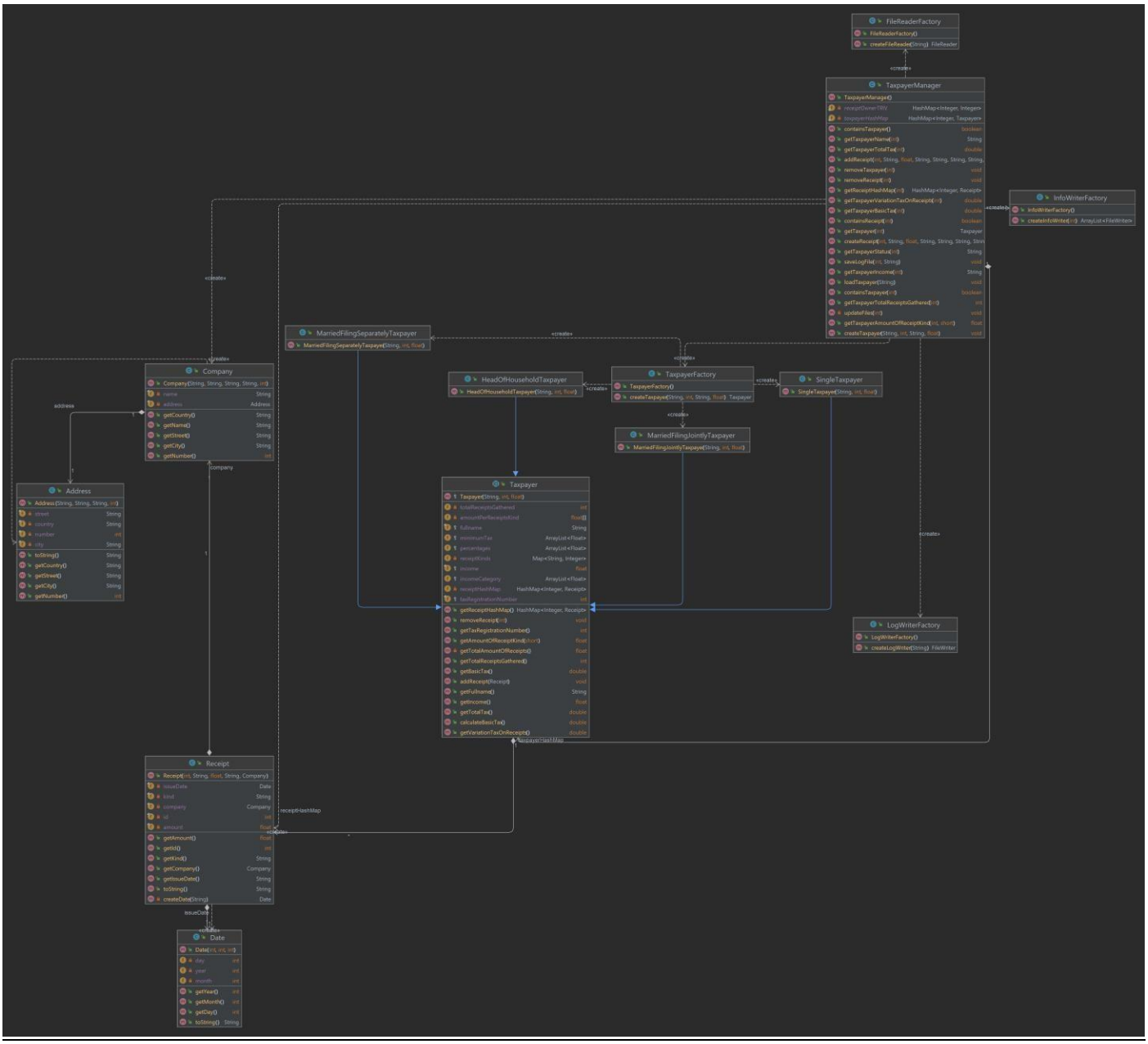
DETAILED DESIGN

- Class Diagrams

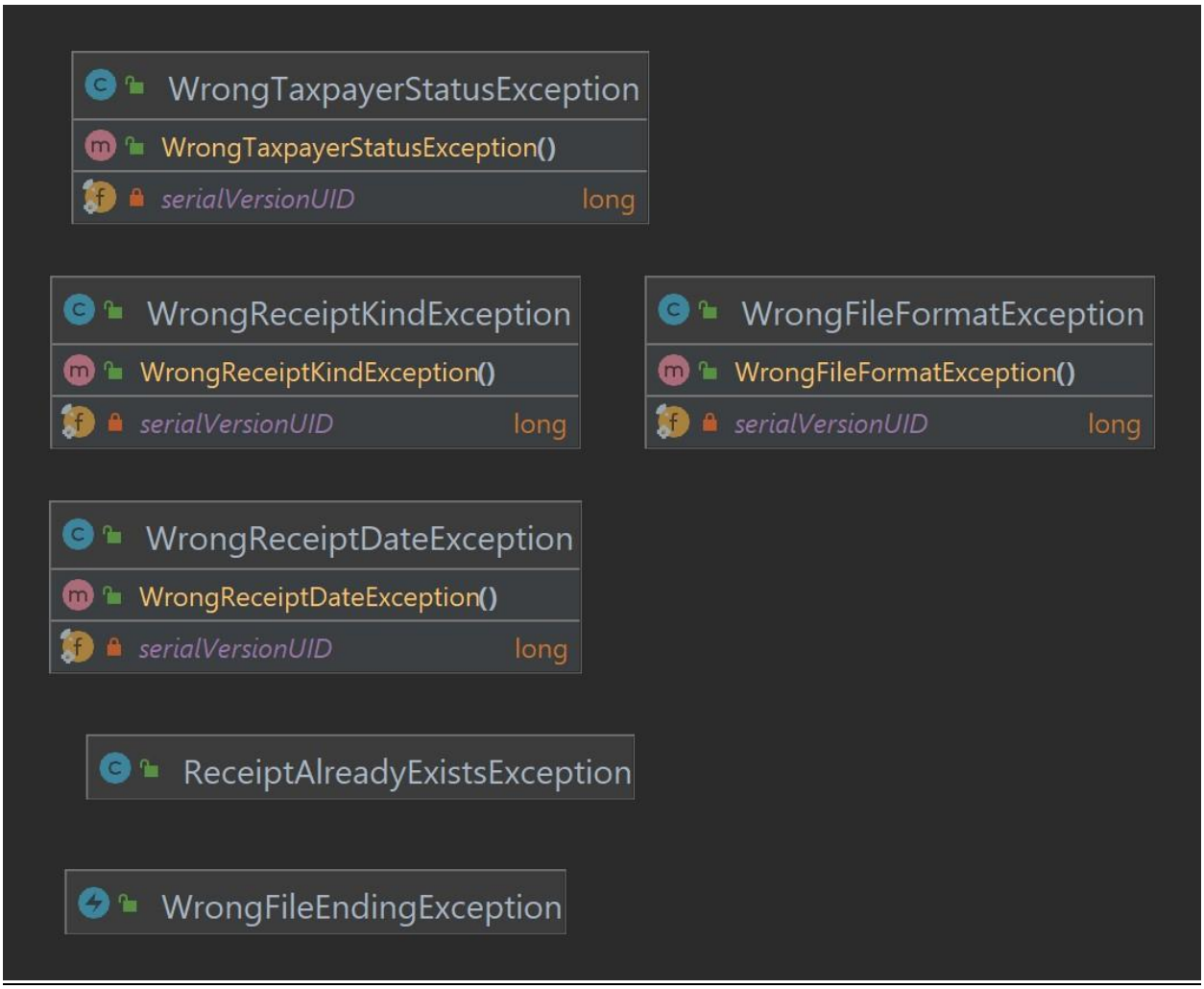
Io package



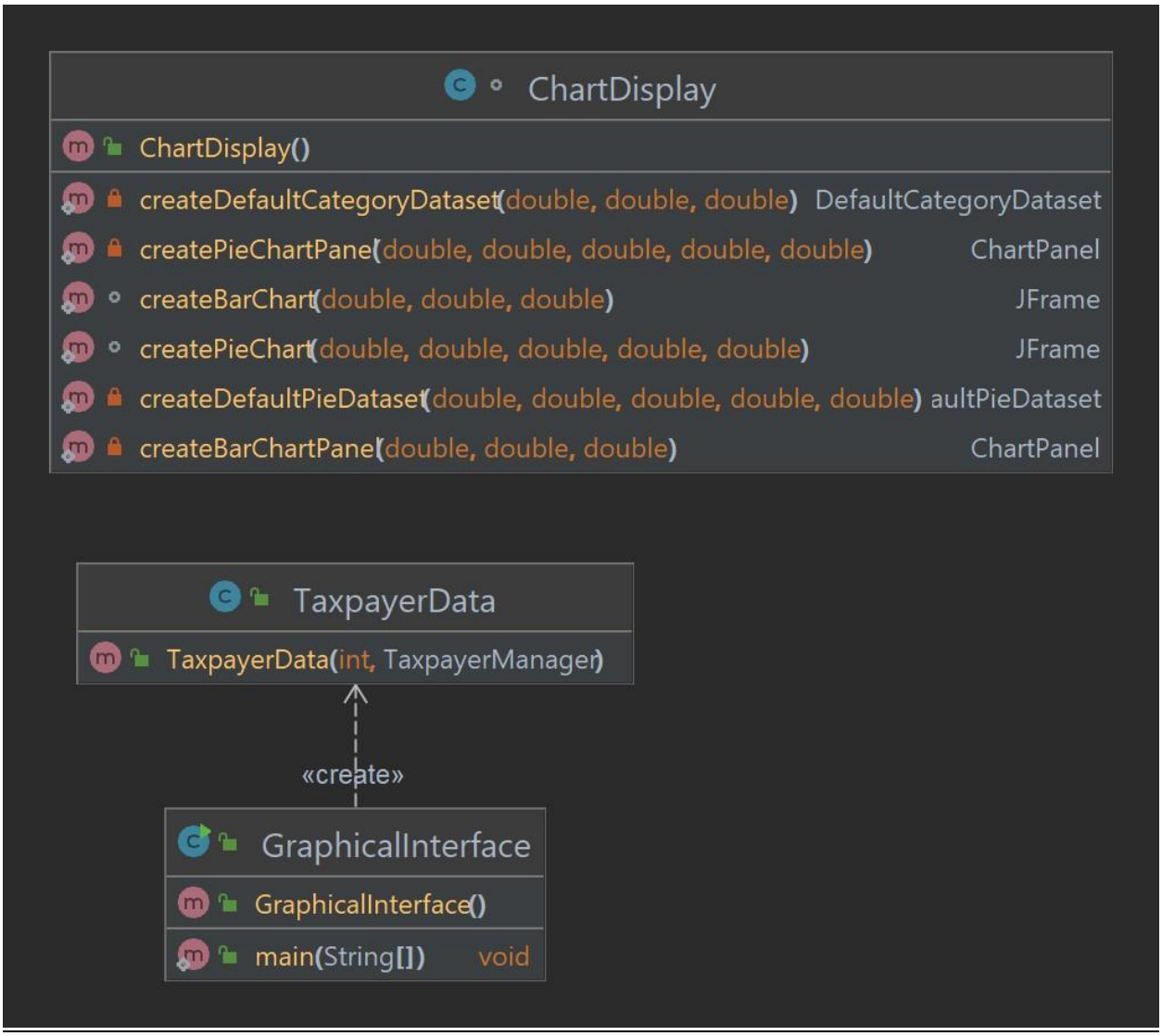
Management package



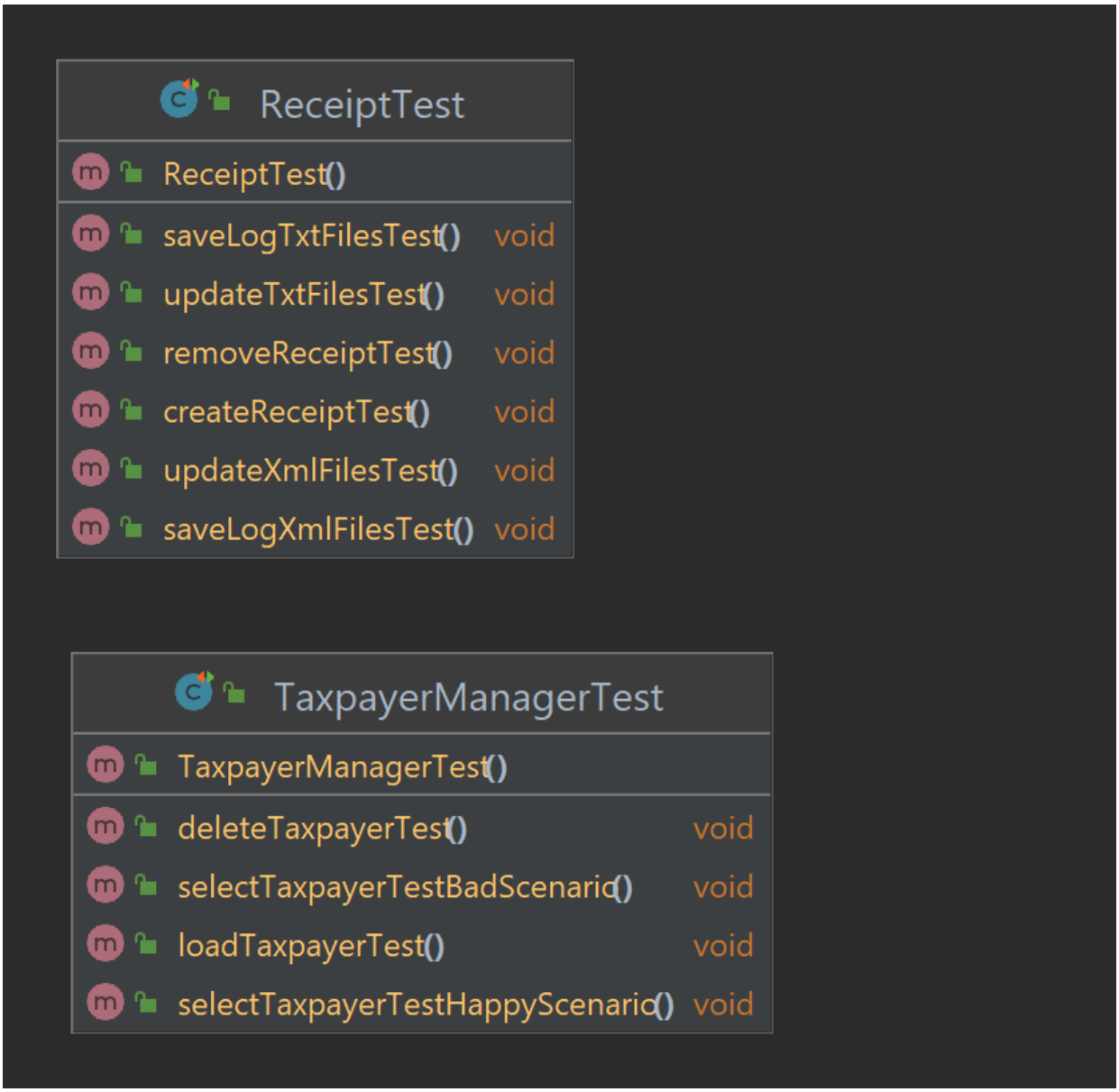
Exceptions


WrongTaxpayerStatusException
WrongTaxpayerStatusException()
serialVersionUID long
WrongReceiptKindException
WrongReceiptKindException()
serialVersionUID long
WrongFileFormatException
WrongFileFormatException()
serialVersionUID long
WrongReceiptDateException
WrongReceiptDateException()
serialVersionUID long
ReceiptAlreadyExistsException
WrongFileEndingException

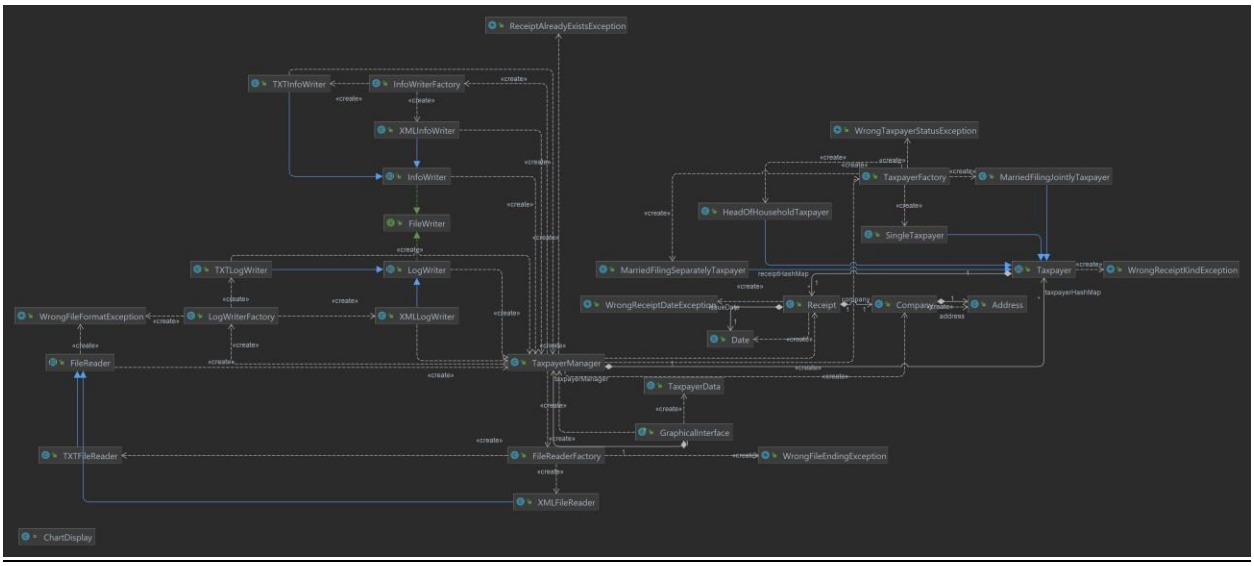
Gui package


ChartDisplay
ChartDisplay()
createDefaultCategoryDataset(double, double, double) DefaultCategoryDataset
createPieChartPane(double, double, double, double, double) ChartPanel
createBarChart(double, double, double) JFrame
createPieChart(double, double, double, double, double) JFrame
createDefaultPieDataset(double, double, double, double, double) aultPieDataset
createBarChartPane(double, double, double) ChartPanel
TaxpayerData
TaxpayerData(int, TaxpayerManager)
«create»
GraphicalInterface
GraphicalInterface()
main(String[]) void

Tests package



Skeleton



The diagrams are also in the "Uml Diagrams" folder for further inspect.

Refactor

- Management package

1. We removed getAddress, because the method was never used.

2. We created a Map (receiptKinds) in which we save the receipt kinds as key and a unique integer as value for each kind. Then we simplified the addReceipt and removeReceipt methods by replacing all the if-else statements with just one if-else statement. With our changes it is easier to add more receipt kinds. For the getVariationTaxOnReceipts method we added two Array lists with the tax increase/decrease and the percentages. Then we replaced all the if-else statements with a for-loop only in one if statement.

3. The basic tax is calculated in the super class Taxpayer, where we created 3 array lists that help calculate the basic tax in one method (calculateBasicTax) instead of four methods. The subclasses are used to add values in the array lists, in the constructor method.

4. For each one of the methods, we created a simple parameterized factory that creates and returns the object each method needs. We simplified the conditions by calling the object's method just one time.

- <u>Io package</u>

1. First, we spotted the differences between the similar methods (checkForReceipt and getValueOfField) of the two subclasses and we created a new method, in each subclass, which implements the differences. Then, we moved the identical methods at the FileReader class and the new methods that were created are now abstract methods in the FileReader class.

2. The classes that were calling FileWriter have been changed and now they call directly TaxpayerManager. So, we removed the methods that were useless after this change (Taxpayer's accessor methods). The rest of the accessor methods in FileWriter class (Receipt's and Company's accessor methods) were pushed down to the classes that need them. Finally, FileWriter had no use as an abstract class, so we changed it to an interface.

3. We created the abstract class InfoWriter that implements FileWriter interface and we moved the identical methods there (accessor methods). We pushed up the two core methods of the subclasses to the abstract class. The differences between them are implemented in 3 new methods (getFileType, taxpayerDataToString, receiptDataToString).

4. We created the abstract class LogWriter that implements FileWriter interface. We pushed up the core method of the subclasses to the abstract class. The differences between them are implemented in 2 new methods (getFileType, dataToString).

- Graphical Interface

  1. We changed how we load, select and delete taxpayer in a way that is easier to use. More specifically we implemented a file chooser to select the file directly from the filesystem. Moreover, selecting and deleting a taxpayer is done by clicking the tax registration number on the" Tax Registration Number" board and pressing the suitable button. Also, you can select a taxpayer by double clicking on its number in the "Tax Registration Number" board.

  2. We changed how we add and delete receipts in a more user-friendly way. We added a drop-down menu for the receipt kinds and we specified how the user is supposed to fill the date. Also, deleting a receipt now is done by clicking the receipt ID number on the "Receipts" board and pressing "Delete Receipt" button. Also, you can select a receipt by double clicking on its number in the "Receipts" board, to view this receipt's data (We added a toString method in Receipt class, which returns a string with the receipt's data).

We ensured that the functionality of the program has remained the same by running the tests we added, each time we refactored the code. Also, we run the application and verified that the results are as expected.

Specifications

- For testing purposes, we created two info files, 111111111_info.txt & 111111111_info.xml. For the tests to be successful these two files must be at the project folder.
- For the "Load Taxpayer" use case, the file selected by the user must be in the project directory. (The problem is originated in how the project was created in the first place. It could be changed, so the user could choose a file wherever is being located, but major changes are needed.

CLASSES RESPONSIBILITIES AND COLLABORATIONS (CRC CARDS)

Class Name:	
Responsibilities	**Collaborations**
▪ List the responsibilities of the class in simple precise statements - don't use passive voice. ○ E.g., This class is responsible for this task. ○ This class performs this activity...	▪ List the dependencies and associations of this class with the other classes of the project

Gui

Class Name: ChartDisplay	
Responsibilities	**Collaborations**
• Creates bar chart • Creates pie chart • Shows charts	• -

Class Name: GraphicalInterface	
Responsibilities	**Collaborations**
• Runs the application • Creates the windows, the buttons and all graphical interface for taxpayers	• TaxpayerData • TaxpayerManager

Class Name: TaxpayerData	
Responsibilities	**Collaborations**
• Presents data of the taxpayer • Creates the windows, the buttons and all graphical interface for receipts	• GraphicalInterface

Exceptions

Class Name: ReceiptAlreadyExistsException	
Responsibilities	**Collaborations**
• Handles errors	• TaxpayerManager

Class Name: WrongFileEndingException	
Responsibilities	**Collaborations**
• Handles errors	• FileReaderFactory

Class Name: WrongFileFormatException	
Responsibilities	**Collaborations**
• Handles errors	• FileReader • LogWriterFactory

Class Name: WrongReceiptDateException	
Responsibilities	**Collaborations**
• Handles errors	• Receipt

Class Name: WrongReceiptKindException	
Responsibilities	**Collaborations**
• Handles errors	• Taxpayer

Class Name: WrongTaxpayerStatusException	
Responsibilities	**Collaborations**
• Handles errors	• TaxpayerFactory

Management

Class Name: Address	
Responsibilities	**Collaborations**
• Creates address	• Company

Class Name: Company	
Responsibilities	**Collaborations**
• Creates address with name	• Address • Receipt • TaxpayerManager

Class Name: Date	
Responsibilities	**Collaborations**
• Creates date	• Receipt

Class Name: FileReaderFactory	
Responsibilities	**Collaborations**
• Generates object based on given information	• WrongFileEndingException • XMLFileReader • TaxpayerManager • TXTFileReader

Class Name: InfoWriterFactory	
Responsibilities	**Collaborations**
• Generates object based on given information	• TXTInfoWriter • XMLInfoWriter • TaxpayerManager

Class Name: LogWriterFactory	
Responsibilities	**Collaborations**
• Generates object based on given information	• TXTLogWriter • XMLLogWriter • TaxpayerManager • WrongFileFormatException

Class Name: TaxpayerFactory	
Responsibilities	**Collaborations**
• Generates object based on given information	• MarriedFilingSeperatlyTaxpayer • MarriedFilingJointlyTaxpayer • TaxpayerManager • WrongTaxpayerStatusException • SingleTaxpayer

Class Name: HeadOfHouseholdTaxpayer	
Responsibilities	**Collaborations**
• Keeps head of household tax data	• TaxpayerFactory • Taxpayer

Class Name: MarriedFillingJointlyTaxpayer	
Responsibilities	**Collaborations**
• Keeps married filing jointly tax data	• TaxpayerFactory • Taxpayer

Class Name: MarriedFillingSeperatlyTaxpayer	
Responsibilities	**Collaborations**
• Keeps married filing seperatly tax data	• TaxpayerFactory • Taxpayer

Class Name: SingleTaxpayer	
Responsibilities	**Collaborations**
• Keeps single tax data	• TaxpayerFactory • Taxpayer

Class Name: Taxpayer	
Responsibilities	**Collaborations**
• Calculates basic tax • Creates Taxpayer • Adds receipts • Removes receipts • Gives informations for receipts	• SingleTaxpayer • MarriedFillingSeperatlyTaxpayer • MarriedFillingJointlyTaxpayer • HeadOfHouseholdTaxpayer • WrongReceiptKindException • TaxpayerManager • Receipt

Class Name: TaxpayerManager	
Responsibilities	**Collaborations**
• Loads taxpayer • Creates taxpayers • Remove taxpayers • Add receipts • Remove receipts • Saves log file • Gives informations for receipts and taxpayers	• GraphicalInterface • FileReaderFactory • Company • Taxpayer • TaxpayerFactory • Receipt • ReceiptAlreadyExistsException • TXTInfoWriter • InfoWriterFactory • XMLInfoWriter • InfoWriter • TXTLogWriter • LofWriter • LogWriterFactory

	• FileReader

Class Name: Receipt	
Responsibilities	**Collaborations**
• Creates receipt • Creates date for receipt • Adds receipts • Gives information for receipts	• Company • Taxpayer • WrongReceiptDateException • TaxpayerManager • Date

Io

Class Name: FileWriter (Interface)	
Responsibilities	**Collaborations**
• Generates file	• InfoWriter • LogWriter

Class Name: InfoWriter	
Responsibilities	**Collaborations**
• Implements FileWriter interface • Writes in info file • Generates taxpayer receipts • Gives information for receipts	• FileWriter • TXTInfoWriter • XMLInfoWriter • TaxpayerManager

Class Name: LogWriter	
Responsibilities	**Collaborations**
• Implements FileWriter interface • Writes in log file	• FileWriter • TXTLogWriter • XMLLogWriter • TaxpayerManager

Class Name: TXTInfoWriter	
Responsibilities	**Collaborations**
• Returns file type • Returns taxpayer data • Returns receipt data	• InfoWriter • InfoWriterFactory • TaxpayerManager

Class Name: XMLInfoWriter	
Responsibilities	**Collaborations**
• Returns file type • Returns taxpayer data • Returns receipt data	• InfoWriter • InfoWriterFactory • TaxpayerManager

Class Name: TXTLogWriter	
Responsibilities	**Collaborations**
• Returns file type • Returns log file's data	• LogWriter • LogWriterFactory • TaxpayerManager

Class Name: XMLLogWriter	
Responsibilities	**Collaborations**
• Returns file type • Returns log file's data	• LogWriter • LogWriterFactory • TaxpayerManager

Class Name: FileReader	
Responsibilities	**Collaborations**
• Gives information for receipts • Checks if there is a specific receipt in the file • Reads file • Reads receipt • Creates taxpayer • Creates receipt • Checks if file is empty	• TXTFileReader • WrongFileFormatException • XMLFileReader

Class Name: TXTFileReader	
Responsibilities	**Collaborations**
• Returns receipt id • Returns a specific value	• FileReader • FileReaderFatctory

Class Name: XMLFileReader	
Responsibilities	**Collaborations**
• Returns receipt id • Returns a specific value	• FileReader • FileReaderFatctory