TAXPAYER MANAGER IMPROVED

OVERALL REPORT

FINAL VERSION

ARGYRIOS ZEZOS 4588, FOTIOS PAPPAS 4773

TABLE OF CONTENTS

INTRODUCTION

The objectives of this phase of the project are to reengineer a legacy application that serves as a management system for minnesota taxpayers. After capturing it’s design to get a grasp of it’s functionality and architecture, tests needs to be created to identify what works before it is modified. Afterward, the actual refactoring takes place constantly being tested. Finally the gui needs to be improved.

REFACTORED DESIGN

USE CASES

The use cases identified from requirements are the following.

1.Load Taxpayer

2.Select Taxpayer

3.Add Receipt

4.Remove Receipt

5.Create Report

6.Save Log File

7.Remove taxpayer

All of them were implemented as methods inside class TaxpayerManager,

except from uc’s 2,5 which don’t correspond to particular methods inside

package gui. A detailed description follows.

Load Taxpayer

Use case ID	1
Actors	The accountant-administrator.
Pre conditions	The data file must be available and of the right format (txt or xml).
Main flow of events	1. The uc starts when the admin selects the desired input file. 2. The system loads it's data in a suitable memory structure.
Alternative flow 1	At any time the admin can cancel the operation or exit the application.
Post conditions	The file contents must be stored appropriately in memory.

Select Taxpayer

Use case ID	2
Actors	The accountant-administrator.
Pre conditions	The input file must have been loaded.
Main flow of events	1. The admin selects a taxpayer. 2. The system displays selected taxpayer's information.
Alternative flow 1	At any time the admin can cancel the operation or exit the application.
Post conditions	None.

Add Receipt

Use case ID	3
Actors	The accountant-administrator.
Pre conditions	The admin must have selected a taxpayer.
Main flow of events	1. The admin selects add receipt option. 2. The system diplays a form, using data collected from form 3. The system stores the receipt and updates the data file.
Alternative flow 1	At any time the admin can cancel the operation or exit the application.
Post conditions	The additional receipts must have been loaded in memory and the data file must have been updated.

Remove Receipt

Use case ID	4
Actors	The accountant-administrator.
Pre conditions	The admin must have selected a taxpayer.
Main flow of events	1. The admin selects receipt to be deleted. 2. The system deletes the receipt from memory and updates data file.
Alternative flow 1	At any time the admin can cancel the operation or exit the application.
Post conditions	The selected receipts must have been deleted and the data file updated.

Create Report

Use case ID	5
Actors	The accountant-administrator.
Pre conditions	The admin must have selected a taxpayer.
Main flow of events	1. The admin selects create report option. 2. The system performs the necessary calculations. 3. The system displays a bar chart with the due tax, the basic tax and it's fluctuation due to receipts. Also a pie chart showcasing the areas on which money was spent.
Alternative flow 1	At any time the admin can cancel the operation or exit the application.
Post conditions	None.

Save Log File

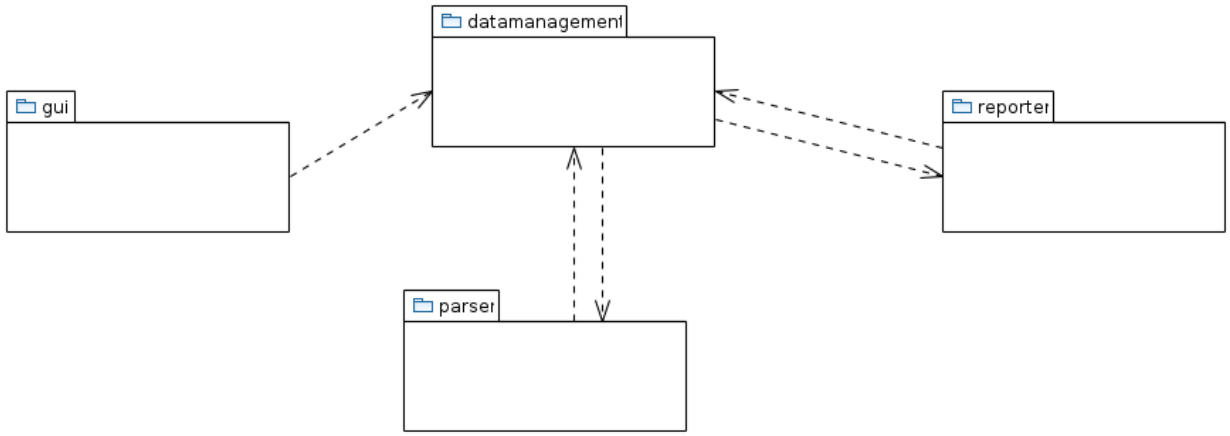
Use case ID	6
Actors	The accountant-administrator.
Pre conditions	The admin must have selected a taxpayer.
Main flow of events	1. The admin selects the option to create a log and it's format. 2. The system creates a log file of the desired format.
Alternative flow 1	At any time the admin can cancel the operation or exit the application.
Post conditions	A log file must have been created.

Remove taxpayer

Use case ID	7
Actors	The accountant-administrator.
Pre conditions	The taxpayer input file must have been loaded.
Main flow of events	1. The admin selects a taxpayer to be deleted. 2. The system deletes the selected taxpayer from memory.
Alternative flow 1	At any time the admin can cancel the operation or exit the application.
Post conditions	The selected taxpayer must have been deleted from memory.

ARCHITECTURE

Overall UML package diagram that shows the architecture of the refactored application.



DETAILED DESIGN

Here is how the different problems of the old design were addressed.

Requirements clearly demand 4 packages.

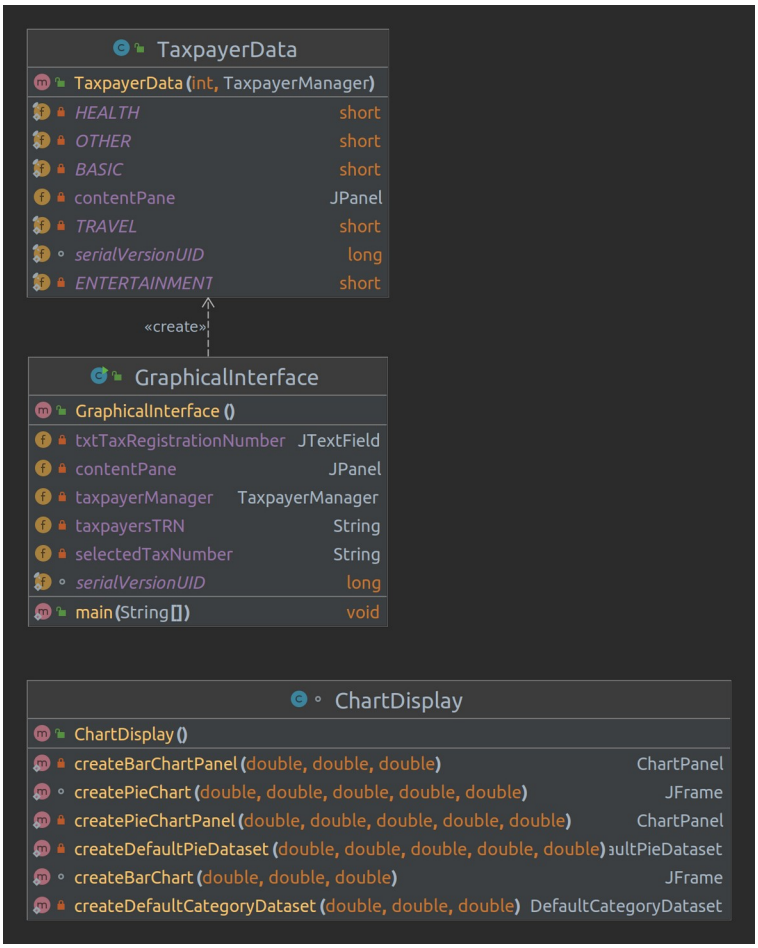
datamanagement.io was split to parser and reporter.

Each exception was moved to the package that uses it.

As requested, factories were constructed to hide object creation details.

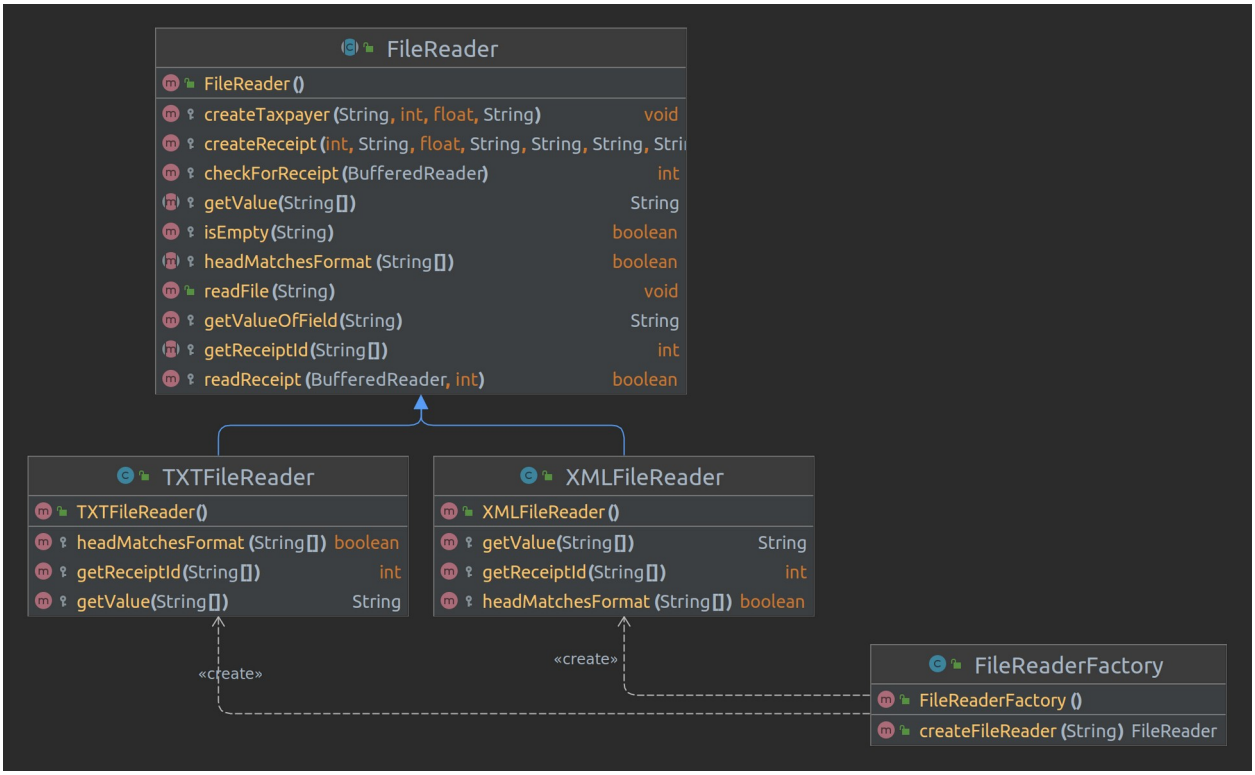
What follows are the UML class diagrams for the classes of each package of the refactored application.

Package gui

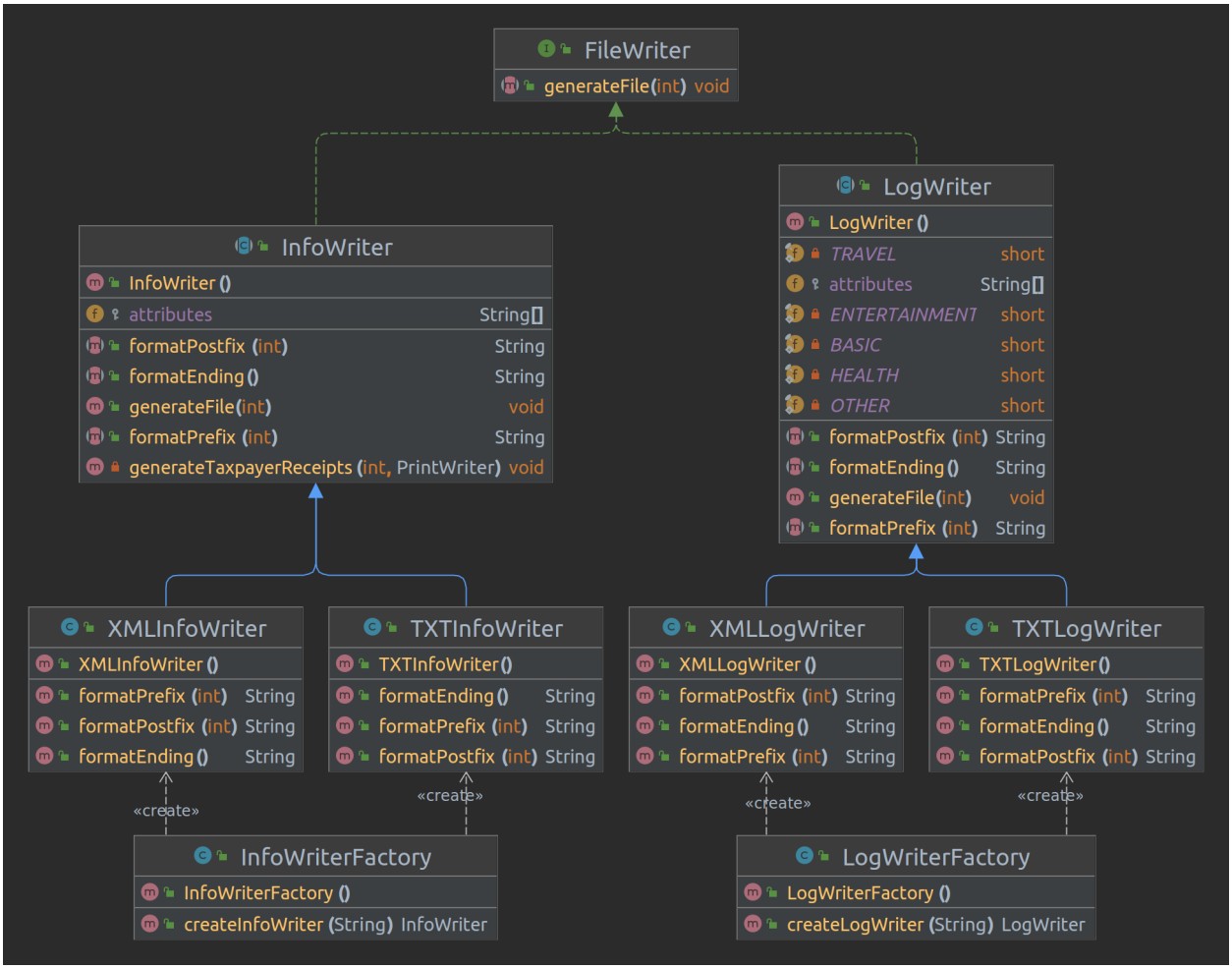

TaxpayerData
TaxpayerData (int, TaxpayerManager)
HEALTH short
OTHER short
BASIC short
contentPane JPanel
TRAVEL short
serialVersionUID long
ENTERTAINMENT short
«create»
GraphicalInterface
GraphicalInterface ()
txtTaxRegistrationNumber JTextField
contentPane JPanel
taxpayerManager TaxpayerManager
taxpayersTRN String
selectedTaxNumber String
serialVersionUID long
main (String[]) void
ChartDisplay
ChartDisplay ()
createBarChartPanel (double, double, double) ChartPanel
createPieChart (double, double, double, double, double) JFrame
createPieChartPanel (double, double, double, double, double) ChartPanel
createDefaultPieDataset (double, double, double, double, double) ultPieDataset
createBarChart (double, double, double) JFrame
createDefaultCategoryDataset (double, double, double) DefaultCategoryDataset

Package datamanagement

Package parser



Package reporter



CLASSES RESPONSIBILITIES AND COLLABORATIONS (CRC CARDS)

Here is a brief description in terms of a CRC card for each class

Class Name: TaxpayerData	
Responsibilities	**Collaborations**
Provides a graphical view of the taxpayer's data. Enables actor to activate certain use cases through buttons such as: add/delete receipt, view report, save data.	Uses Receipt. Uses TaxpayerManager.

Class Name: GraphicalInterface	
Responsibilities	**Collaborations**
Hosts the main method. Acts as the graphical home screen. Enables actor to activate certain use cases through buttons such as: load taxpayer, select taxpayer, delete taxpayer.	Has a TaxpayerManager.

Class Name: ChartDisplay	
Responsibilities	**Collaborations**
Responsible for creating the pie and bar charts using taxpayer’s data.	-

Class Name: Address	
Responsibilities	**Collaborations**
Store company’s address data.	Composite of Company.

Class Name: Company	
Responsibilities	**Collaborations**
Stores information regarding a particular company.	Composed of Address.

Class Name: Date	
Responsibilities	**Collaborations**
Stores information about the creation date of a particular receipt.	Composite of Address.

Class Name: Receipt	
Responsibilities	**Collaborations**
Holds the information about a Taxpayer’s Receipt.	Composite of Taxpayer. Composed of Date, Company.

Class Name: Taxpayer	
Responsibilities	**Collaborations**
Holds data of a taxpayer on wich performs calculations. Provides helper methods for some use cases.	Composite of Taxpayer Manager Composed of Receipt

Class Name: TaxpayerFactory	
Responsibilities	**Collaborations**
Responsible for creating the different types of taxpayers.	Used by TaxpayerManager

Class Name: TaxpayerManager	
Responsibilities	**Collaborations**
Bussiness logic class that implements 5 of 7 use cases. Performs the core funtionality of the application.	Uses Parser Uses Reporter Has many Taxpayers

Class Name: HeadOfHouseholdTaxpayer	
Responsibilities	**Collaborations**
Specific type of taxpayer wich extends.	Is a taxpayer

Class Name: MarriedFilingJointlyTaxpayer	
Responsibilities	**Collaborations**
Specific type of taxpayer wich extends.	Is a taxpayer

Class Name: SingleTaxpayer	
Responsibilities	**Collaborations**
Specific type of taxpayer wich extends.	Is a taxpayer

Class Name: MarriedFilingSeparatelyTaxpayer	
Responsibilities	**Collaborations**
Specific type of taxpayer wich extends.	Is a taxpayer

Class Name: FileReader	
Responsibilities	**Collaborations**
Abstract class that is responsible for parsing taxpayer and receipt data from a text or xml format file.	Uses TaxpayerManager

Class Name: FileReaderFactory	
Responsibilities	**Collaborations**
Creates the different types of FileReaders	Uses FileReader

Class Name: TXTFileReader	
Responsibilities	**Collaborations**
Specific type of FileReader wich extends. Parses text fromatted files.	Is a FileReader

Class Name: XMLFileReader	
Responsibilities	**Collaborations**
Specific type of FileReader wich extends. Parses xml formatted files.	Is a FileReader

Class Name: FileWriter	
Responsibilities	**Collaborations**
Interface with a single method for generating a file.	-

Class Name: InfoWriter	
Responsibilities	**Collaborations**
Abstract class implementing FileWriter. Responsible for updating INFO files.	Uses TaxpayerManager Uses Receipt

Class Name: LogWriter	
Responsibilities	**Collaborations**
Abstract class implementing FileWriter. Responsible for updating LOG files.	Uses TaxpayerManager

Class Name: InfoWriterFactory	
Responsibilities	**Collaborations**
Creates the different types of InfoWriters.	Uses InfoWriter.

Class Name: LogWriterFactory	
Responsibilities	**Collaborations**
Creates the different types of LogWriters.	Uses LogWriter.

Class Name: XMLInfoWriter	
Responsibilities	**Collaborations**
Specific type of InfoWriter wich extends. Updates xml formatted INFO files.	Is an InfoWriter

Class Name: TXTInfoWriter	
Responsibilities	**Collaborations**
Specific type of InfoWriter wich extends. Updates txt formatted INFO files.	Is an InfoWriter

Class Name: XMLLogWriter	
Responsibilities	**Collaborations**
Specific type of LogWriter wich extends. Creates xml formatted LOG files.	Is a LogWriter

Class Name: TXTLogWriter	
Responsibilities	**Collaborations**
Specific type of LogWriter wich extends. Creates txt formatted LOG files.	Is a LogWriter

Report ends here, thank you for reading.