INCOME TAX CALCULATOR

DESIGN RECOVERY AND QUALITY ASSESSMENT REPORT

VERSION <1.0>

Γκαβαρδίνας Όθωνας, 2620

Μπουρλή Στυλιανή, 2774

–

TABLE OF CONTENTS

INTRODUCTION

Goal of the project

The goal of this project is to reengineer a Java application. At a glance, the application serves for the income tax calculation of the Minnesota state citizens. The tax calculation accounts for the marital status of a given citizen, his income, and the amount of money 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.

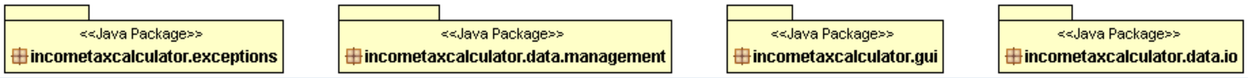
Goal of this phase (2st phase)

The goal of this phase is to repair problems like bad names, unnecessary complexity, duplicate code, “Middle Man” classes and large classes.

DESIGN RECOVERY

ARCHITECTURE

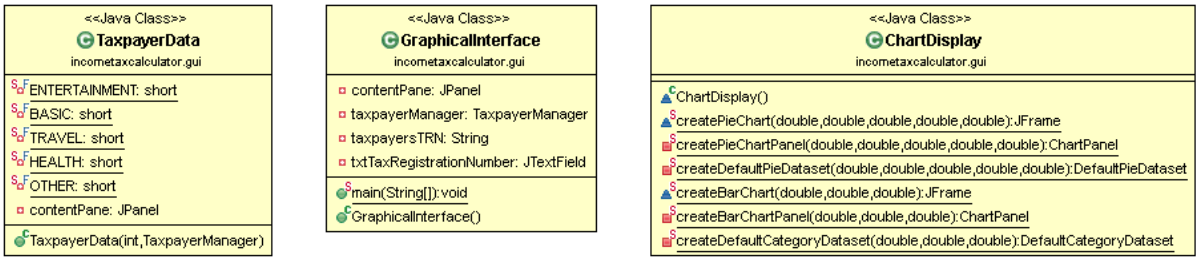
- **UML package diagram**


<<Java Package>>
incometaxcalculator.exceptions
<<Java Package>>
incometaxcalculator.data.management
<<Java Package>>
incometaxcalculator.gui
<<Java Package>>
incometaxcalculator.data.io

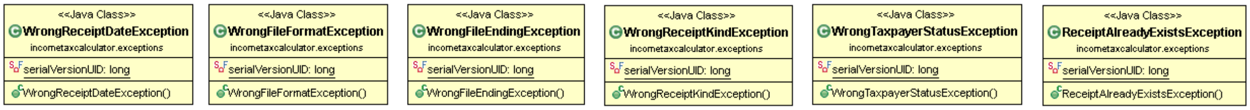
DETAILED DESIGN

- **UML class diagrams for the classes of each package:**

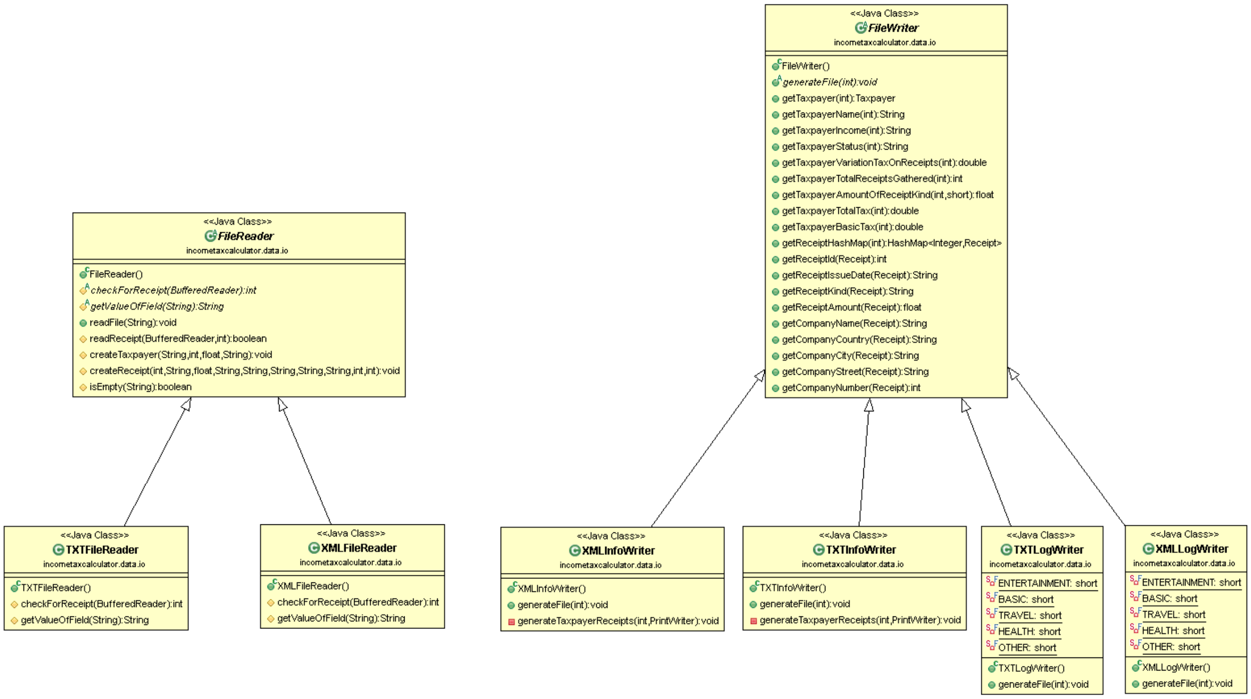
- Package **incometaxcalculator.gui**


<<Java Class>>
TaxpayerData
incometaxcalculator.gui
ENTERTAINMENT: short
BASIC: short
TRAVEL: short
HEALTH: short
OTHER: short
contentPane: JPanel
TaxpayerData(int,TaxpayerManager)
<<Java Class>>
GraphicalInterface
incometaxcalculator.gui
contentPane: JPanel
taxpayerManager: TaxpayerManager
taxpayersTRN: String
txtTaxRegistrationNumber: JTextField
main(String[]):void
GraphicalInterface()
<<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

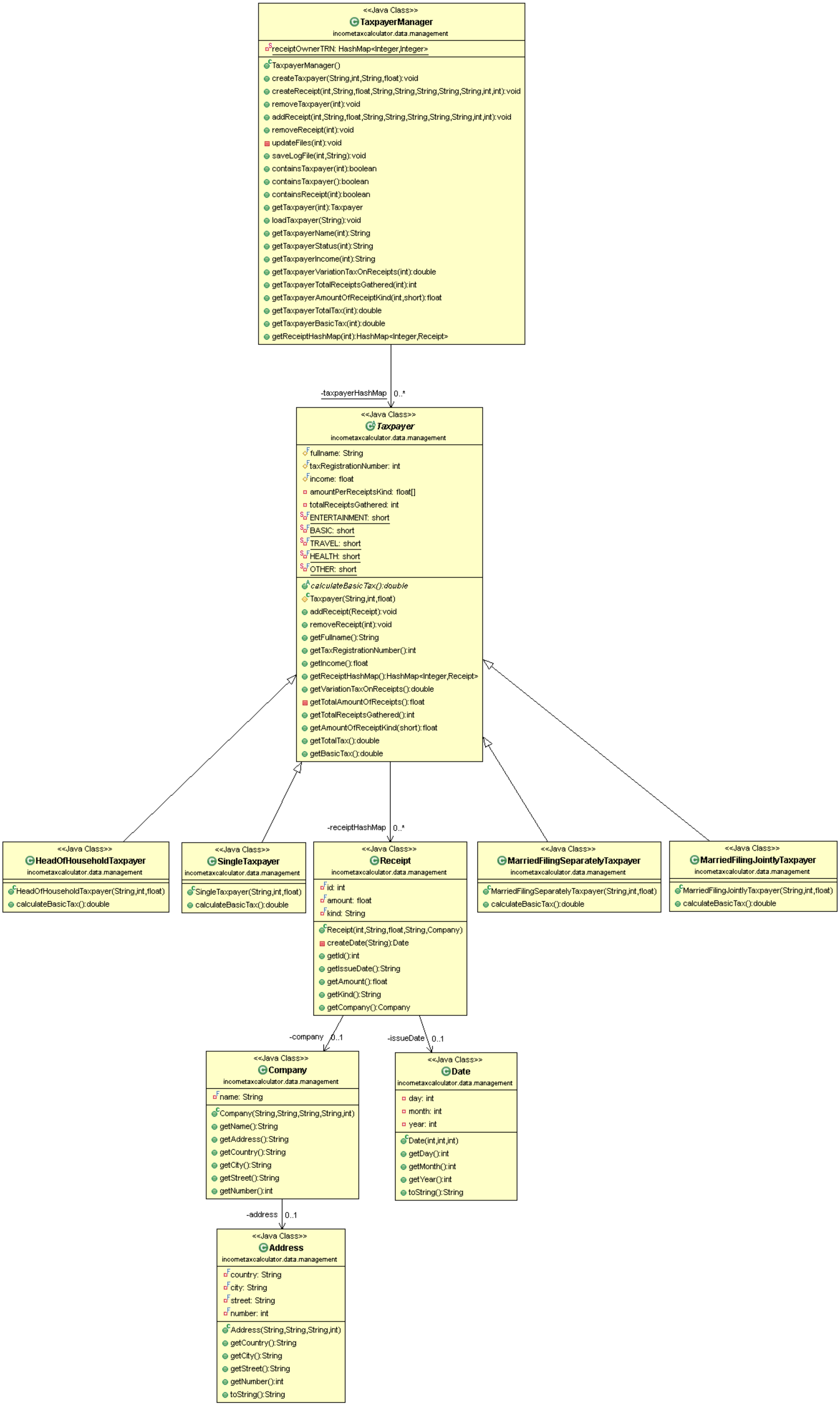
- Package **incometaxcalculator.exceptions**


<<Java Class>>
WrongReceiptDateException
incometaxcalculator.exceptions
serialVersionUID: long
WrongReceiptDateException()
<<Java Class>>
WrongFileFormatException
incometaxcalculator.exceptions
serialVersionUID: long
WrongFileFormatException()
<<Java Class>>
WrongFileEndingException
incometaxcalculator.exceptions
serialVersionUID: long
WrongFileEndingException()
<<Java Class>>
WrongReceiptKindException
incometaxcalculator.exceptions
serialVersionUID: long
WrongReceiptKindException()
<<Java Class>>
WrongTaxpayerStatusException
incometaxcalculator.exceptions
serialVersionUID: long
WrongTaxpayerStatusException()
<<Java Class>>
ReceiptAlreadyExistsException
incometaxcalculator.exceptions
serialVersionUID: long
ReceiptAlreadyExistsException()

- Package **incometaxcalculator.data.io**


<<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>>
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>>
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

- Package **incometaxcalculator.data.management**


<<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>>
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
-receiptHashMap
0..*
<<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>>
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
<<Java Class>>
MarriedFilingSeparatelyTaxpayer
incometaxcalculator.data.management
MarriedFilingSeparatelyTaxpayer(String,int,float)
calculateBasicTax():double
<<Java Class>>
MarriedFilingJointlyTaxpayer
incometaxcalculator.data.management
MarriedFilingJointlyTaxpayer(String,int,float)
calculateBasicTax():double
-company
0..1
-issueDate
0..1
<<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
<<Java Class>>
Date
incometaxcalculator.data.management
day: int
month: int
year: int
Date(int,int,int)
getDay():int
getMonth():int
getYear():int
toString():String
-address
0..1
<<Java Class>>
Address
incometaxcalculator.data.management
country: String
city: String
street: String
number: int
Address(String,String,String,int)
getCountry():String
getCity():String
getStreet():String
getNumber():int
toString():String

IMPLEMENTATION

Class Name: FileReader	
Responsibilities	**Collaborations**
▪ This class is an abstract calss, which contains two abstract methods (checkForReiceipt, getValueOfField). ▪ This class is responsible for reading some information from file for a Taxpayer and the receipts of these informations.	▪ TXTFileReader ▪ XMLFileReader ▪ TaxpayerManager

Class Name: TXTFileReader	
Responsibilities	**Collaborations**
▪ This class implements checkForReceipt method and getValueOfField method, which is located in FileReader class. ▪ This class is responsible for reading information from a txt file.	▪ FileReader

Class Name: XMLFileReader	
Responsibilities	**Collaborations**

▪ This class implements checkForReceipt method and getValueOfField method, which is located in FileReader class. ▪ This class is responsible to read information from a xml file.	▪ FileReader

Class Name: FileWriter	
Responsibilities	**Collaborations**
▪ This class is an abstract class which contains an abstract method (generateFile). ▪ This class is responsible for writing in a file information for a Taxpayer and these information receipts to a file.	▪ XMLInfoWriter ▪ TXTInfoWriter ▪ TXTLogWriter ▪ XMLLogWriter ▪ TaxpayerManager ▪ Receipt

Class Name: TXTInfoWriter	
Responsibilities	**Collaborations**
▪ This class implements generateFile method, that is located in FileWriter class. ▪ This class is responsible for	▪ FileWriter

updating information of a Taxpayer’s receipts. This is done by writing to a file.	

Class Name: TXTLogWriter	
Responsibilities	**Collaborations**
▪ This class implements generateFile method, that is located in FileWriter class. ▪ This class is responsible for writing information of a Taxpayer and their receipts to a txt file.	▪ FileWriter

Class Name: XMLInfoWriter	
Responsibilities	**Collaborations**
▪ This class implements generateFile method, that is located in FileWriter class. ▪ This class is responsible for writing information of a Taxpayer and their receipts to a xml file.	▪ FileWriter

Class Name: XMLLogWriter

Responsibilities	**Collaborations**
▪ This class implements generateFile method, that is located in FileWriter class. ▪ This class is responsible for updating information of a Taxpayer’s receipts. This is done by writing to a file.	▪ FileWriter

Class Name: Address	
Responsibilities	**Collaborations**
▪ This class is responsible for an address creation. An address contains country, city, street and number. ▪ This class contains methods that return the data of its fields.	▪ Company

Class Name: Company	
Responsibilities	**Collaborations**
▪ This class is responsible for a company creation. A company contains name and address. ▪ This class contains methods	▪ Address ▪ Receipt

that return the data of its fields.	

Class Name: Date	
Responsibilities	**Collaborations**
▪ This class is responsible for a date creation. A date contains day, month and year. ▪ This class contains methods that return the data of its fields.	▪ Receipt

Class Name: HeadOfHouseholdTaxpayer	
Responsibilities	**Collaborations**
▪ This class implements calculateBasicTax method, that is located in Taxpayer class. ▪ This class includes the algorithm, that calculates the tax for a Head of Household Taxpayer.	▪ Taxpayer

Class Name: MarriedFilingJointlyTaxpayer	
Responsibilities	**Collaborations**

▪ This class implements calculateBasicTax method, that is located in Taxpayer class. ▪ This class includes the algorithm, that calculates the tax for a Married Filling Jointly Taxpayer.	▪ Taxpayer

Class Name: MarriedFilingSeparatelyTaxpayer	
Responsibilities	**Collaborations**
▪ This class implements calculateBasicTax method, that is located in Taxpayer class. ▪ This class includes the algorithm, that calculates the tax for a Married Filling Separately Taxpayer.	▪ Taxpayer

Class Name: SingleTaxpayer	
Responsibilities	**Collaborations**
▪ This class implements calculateBasicTax method, that is located in Taxpayer class. ▪ This class includes the algorithm, that calculates the	▪ Taxpayer

tax for a Single Taxpayer.	

Class Name: Receipt	
Responsibilities	**Collaborations**
▪ This class is responsible for the creation of a receipt. ▪ This class contains methods that return the data of its fields.	▪ Taxpayer ▪ Company ▪ Date

Class Name: Taxpayer	
Responsibilities	**Collaborations**
▪ This class is an abstract class, which contains an abstract method(calculateBasicTax). ▪ This class contains a Hashmap in which the receipts are stored. ▪ This class is responsible for adding or removing a receipt for a Taxpayer. ▪ This class contains methods that return the data of its fields and also a method which calculates the tax based on the	▪ HeadOfHouseholdTaxpayer ▪ MarriedFilingJointlyTaxpayer ▪ MarriedFilingSeparetlyTaxpayer ▪ SingleTaxpayer ▪ Receipt ▪ TaxpayerManager

receipts.	

Class Name: TaxpayerManager	
Responsibilities	**Collaborations**
▪ This class contains two HashMaps, one for the Taxpayers and one for the receipts. ▪ This class includes a method that creates Taxpayer objects and adds them to the HashMap and also a method that removes a Taxpayer from the HashMap. ▪ This class includes a method that creates a receipt and adds it to the HashMap and a method that removes a receipt from the HashMap. ▪ This class contains a method for updating files when a change takes place and also a method that saves data in a file and a method that loads data from file. ▪ This class contains a method that checks if a Taxpayer	▪ HeadOfHouseholdTaxpayer ▪ MarriedFilingJointlyTaxpayer ▪ MarriedFilingSeparetlyTaxpayer ▪ SingleTaxpayer ▪ Receipt ▪ Company ▪ TXTFileReader ▪ XMLFileReader ▪ FileReader ▪ TXTLogWriter ▪ XMLLogWriter ▪ XMLInfoWriter ▪ TXTInfoWriter ▪ FileWriter

exists, a method that checks if a receipt exists and some methods that return the data of its fields.	

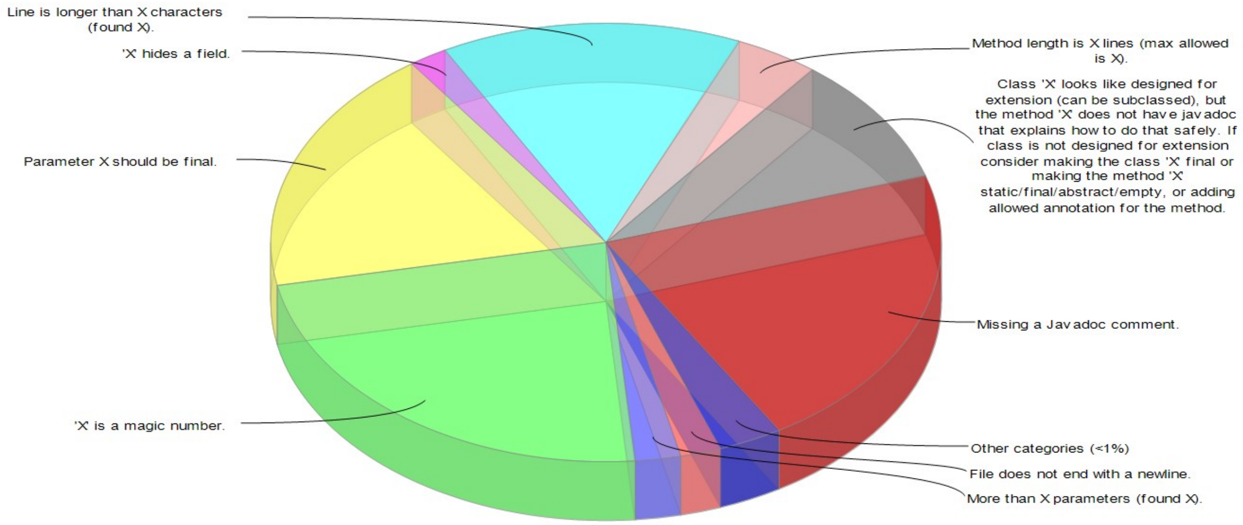
STYLE VIOLATIONS

Συνολικά το check style εντόπισε 912 violations, τα οποία περιγράφονται επιγραμματικά παρακάτω:

- Χρήση μαγικών αριθμών, δηλαδή σταθερών που δεν ανατίθενται σε μεταβλητές. (212 violations)
- Έλλειψη Javadoc σχολίων για την περιγραφή του κώδικα. (194 violations)
- Έλλειψη δήλωσης ως final μεταβλητών που δεν τροποποιούνται στο χρόνο εκτέλεσης. (167 violations)
- Ύπαρξη περισσοτέρων από 80 χαρακτήρων σε μία γραμμή κώδικα. (132 violations)
- Ύπαρξη abstract, ή μητρικής κλάσης σε κληρονομικότητα, με public ή protected μεθόδους, οι οποίες δε διαθέτουν javadoc, το οποίο να περιγράφει τον τρόπο χρήσης τους. (87 violations)
- Ύπαρξη μεθόδων με πλήθος γραμμών μεγαλύτερο του ορίου των 10 γραμμών. (37 violations)
- Ύπαρξη μεθόδων που περιέχουν περισσότερες από δύο παραμέτρους. (~20 violations) **–** Ύπαρξη κλάσης που δεν τελειώνει με αλλαγή γραμμής. (~20 violations) **–** Ύπαρξη προσωρινής μεταβλητής ή παραμέτρου, η οποία έχει ίδιο όνομα με ένα πεδίο. (~20 violations)
- Δήλωση πίνακα με brackets δίπλα από τη μεταβλητή και όχι δίπλα στον τύπο του, με αποτέλεσμα να μην είναι ευδιάκριτο ότι πρόκειται για πίνακα. (8 violations)

- Το μικρότερο μέρος των περιπτώσεων (1-4 violations) αποτελείται από τα εξής:
  - **-** Δεν υπάρχει package-info.java αρχείο για τα javadock comments στα πακέτα.
  - **-** Γίνεται ανάθεση μέσα σε δομές επανάληψης και διακλάδωσης.

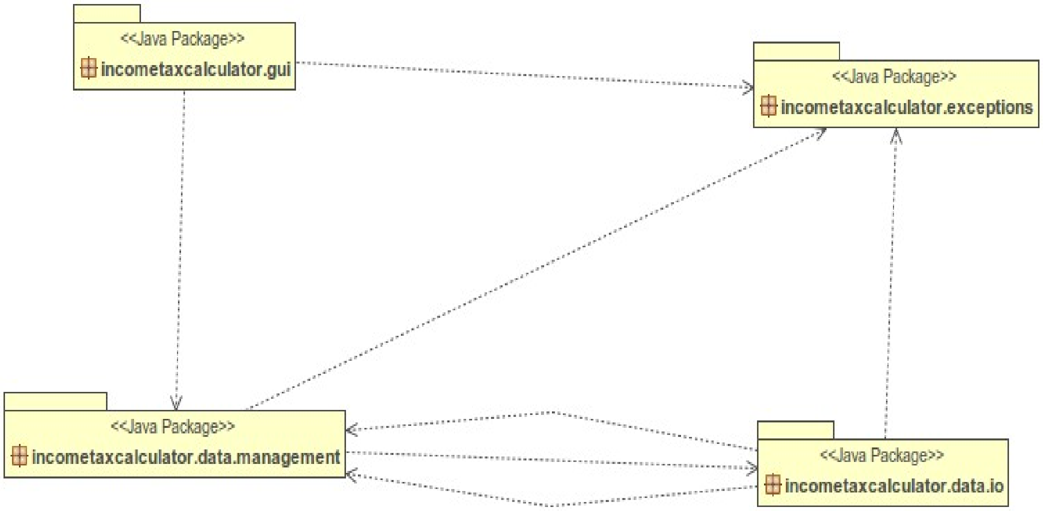
- Υπάρχουν protected μεταβλητές, αντί για private μεταβλητές με set-get μεθόδους.
- Χρησιμοποιείται η μεταβλητή TRN, δηλαδή χωρίς cammelCase και 1ο γράμμα lower.
- Το αρχείο (κλάση) έχει μέγεθος > 200 γραμμές.
- Δεν υπάρχει λευκός χαρακτήρας μετά το χαρακτήρα ";", αλλά σχόλιο.
- Υπάρχει κώδικας με boolean τιμές που μπορεί να απλοποιηθεί.
- Σε κάποιο while υπάρχει μόνο ο χαρακτήρας ";".
- Το ίδιο while συνοδεύεται από "{" και "}".
- Μια κλάση δεν διαθέτει ορισμένο constructor, αλλά τον public constructor.



REFACTORED DESIGN

ARCHITECTURE

- **UML package diagram**

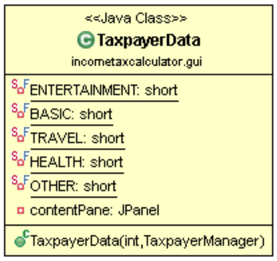


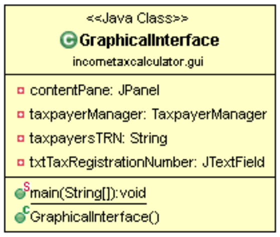
<<Java Package>>
incometaxcalculator.gui
<<Java Package>>
incometaxcalculator.exceptions
<<Java Package>>
incometaxcalculator.data.management
<<Java Package>>
incometaxcalculator.data.io

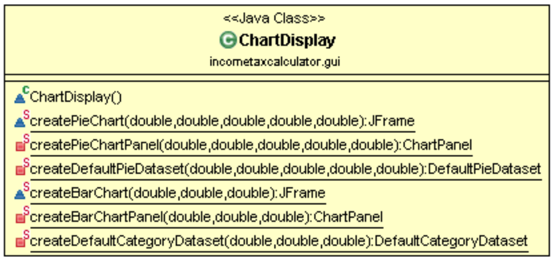
DETAILED DESIGN

- **UML class diagrams for the classes of each package:**

- Package **incometaxcalculator.gui**

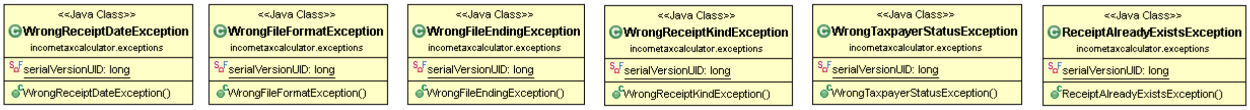




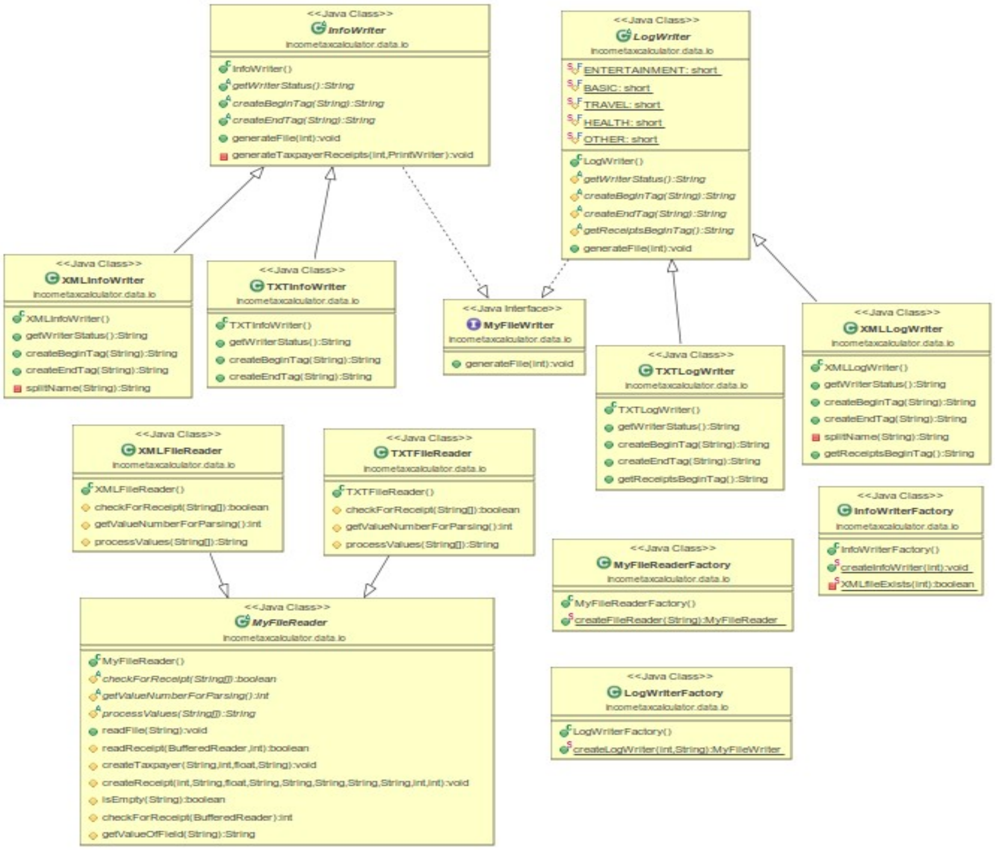


<<Java Class>>
ChartDisplay
incometaxcalculator.gui
ChartDisplay()
createPieChart(double,double,double,double,double):JFrame
createPieChartPanel(double,double,double,double,double):ChartPanel
createDefaultPieDataset(double,double,double,double,double):DefaultPieDataset
createBarChart(double,double,double):JFrame
createBarChartPanel(double,double,double):ChartPanel
createDefaultCategoryDataset(double,double,double):DefaultCategoryDataset

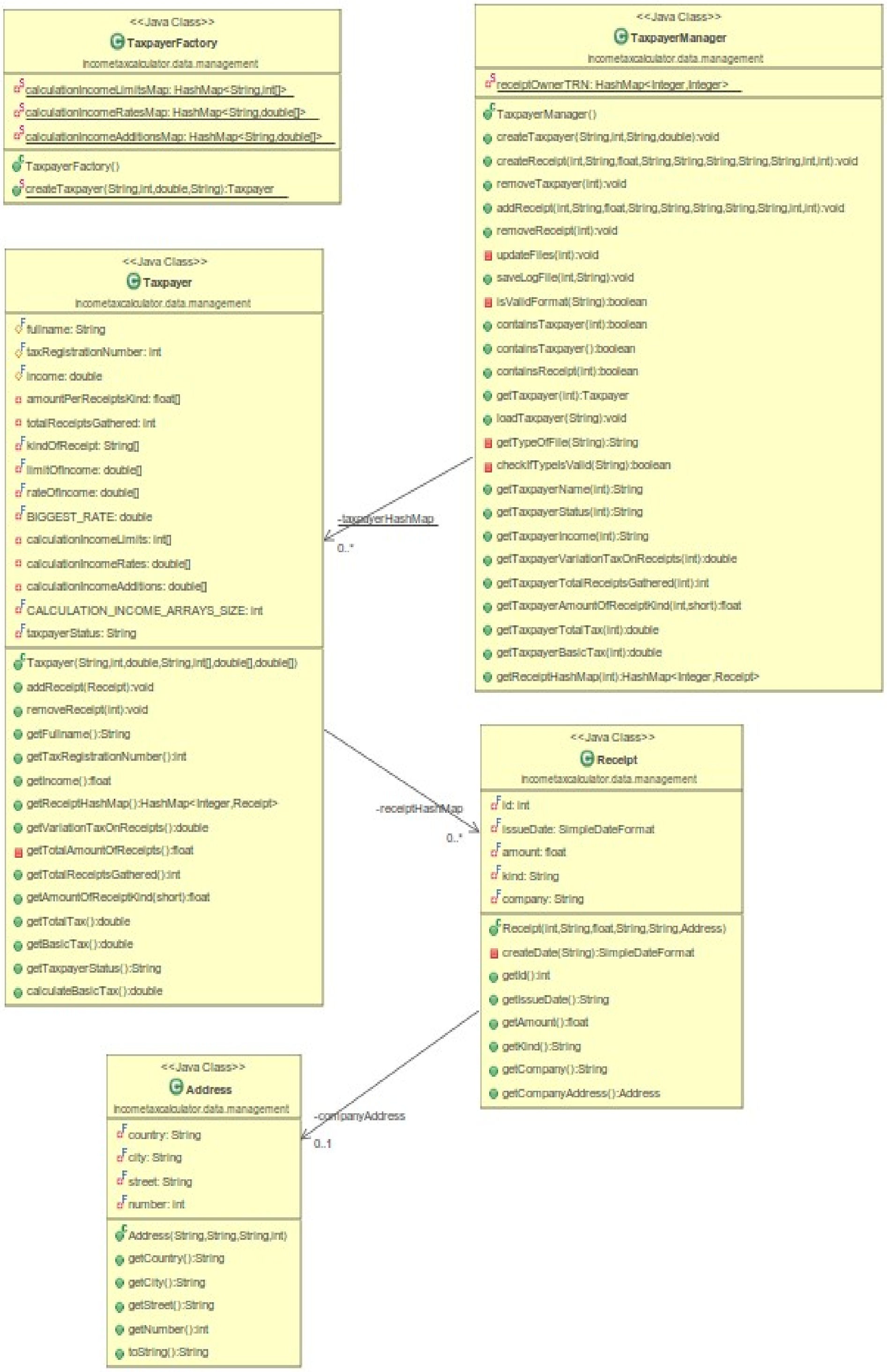
- Package **incometaxcalculator.exceptions**


<<Java Class>>
WrongReceiptDateException
incometaxcalculator.exceptions
serialVersionUID: long
WrongReceiptDateException()
<<Java Class>>
WrongFileFormatException
incometaxcalculator.exceptions
serialVersionUID: long
WrongFileFormatException()
<<Java Class>>
WrongFileEndingException
incometaxcalculator.exceptions
serialVersionUID: long
WrongFileEndingException()
<<Java Class>>
WrongReceiptKindException
incometaxcalculator.exceptions
serialVersionUID: long
WrongReceiptKindException()
<<Java Class>>
WrongTaxpayerStatusException
incometaxcalculator.exceptions
serialVersionUID: long
WrongTaxpayerStatusException()
<<Java Class>>
ReceiptAlreadyExistsException
incometaxcalculator.exceptions
serialVersionUID: long
ReceiptAlreadyExistsException()

- Package **incometaxcalculator.data.io**


<<Java Class>>
InfoWriter
incometaxcalculator.data.io
InfoWriter()
getWriterStatus():String
createBeginTag(String):String
createEndTag(String):String
generateFile(int):void
generateTaxpayerReceipts(int,PrintWriter):void
<<Java Class>>
LogWriter
incometaxcalculator.data.io
ENTERTAINMENT: short
BASIC: short
TRAVEL: short
HEALTH: short
OTHER: short
LogWriter()
getWriterStatus():String
createBeginTag(String):String
createEndTag(String):String
getReceiptsBeginTag():String
generateFile(int):void
<<Java Class>>
XMLInfoWriter
incometaxcalculator.data.io
XMLInfoWriter()
getWriterStatus():String
createBeginTag(String):String
createEndTag(String):String
splitName(String):String
<<Java Class>>
TXTInfoWriter
incometaxcalculator.data.io
TXTInfoWriter()
getWriterStatus():String
createBeginTag(String):String
createEndTag(String):String
<<Java Interface>>
MyFileWriter
incometaxcalculator.data.io
generateFile(int):void
<<Java Class>>
TXTLogWriter
incometaxcalculator.data.io
TXTLogWriter()
getWriterStatus():String
createBeginTag(String):String
createEndTag(String):String
getReceiptsBeginTag():String
<<Java Class>>
XMLLogWriter
incometaxcalculator.data.io
XMLLogWriter()
getWriterStatus():String
createBeginTag(String):String
createEndTag(String):String
splitName(String):String
getReceiptsBeginTag():String
<<Java Class>>
XMLFileReader
incometaxcalculator.data.io
XMLFileReader()
checkForReceipt(String[]):boolean
getValueNumberForParsing():int
processValues(String[]):String
<<Java Class>>
TXTFileReader
incometaxcalculator.data.io
TXTFileReader()
checkForReceipt(String[]):boolean
getValueNumberForParsing():int
processValues(String[]):String
<<Java Class>>
InfoWriterFactory
incometaxcalculator.data.io
InfoWriterFactory()
createInfoWriter(int):void
XMLfileExists(int):boolean
<<Java Class>>
MyFileReaderFactory
incometaxcalculator.data.io
MyFileReaderFactory()
createFileReader(String):MyFileReader
<<Java Class>>
MyFileReader
incometaxcalculator.data.io
MyFileReader()
checkForReceipt(String[]):boolean
getValueNumberForParsing():int
processValues(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
checkForReceipt(BufferedReader):int
getValueOfField(String):String
<<Java Class>>
LogWriterFactory
incometaxcalculator.data.io
LogWriterFactory()
createLogWriter(int,String):MyFileWriter

- Package **incometaxcalculator.data.management**


<<Java Class>>
TaxpayerFactory
incometaxcalculator.data.management
calculationIncomeLimitsMap: HashMap<String,int[]>
calculationIncomeRatesMap: HashMap<String,double[]>
calculationIncomeAdditionsMap: HashMap<String,double[]>
TaxpayerFactory()
createTaxpayer(String,int,double,String):Taxpayer
<<Java Class>>
TaxpayerManager
incometaxcalculator.data.management
receiptOwnerTRN: HashMap<Integer,Integer>
TaxpayerManager()
createTaxpayer(String,int,String,double):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
isValidFormat(String):boolean
containsTaxpayer(int):boolean
containsTaxpayer():boolean
containsReceipt(int):boolean
getTaxpayer(int):Taxpayer
loadTaxpayer(String):void
getTypeOfFile(String):String
checkIfTypeIsValid(String):boolean
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>
<<Java Class>>
Taxpayer
incometaxcalculator.data.management
fullname: String
taxRegistrationNumber: int
income: double
amountPerReceiptsKind: float[]
totalReceiptsGathered: int
kindOfReceipt: String[]
limitOfIncome: double[]
rateOfIncome: double[]
BIGGEST_RATE: double
calculationIncomeLimits: int[]
calculationIncomeRates: double[]
calculationIncomeAdditions: double[]
CALCULATION_INCOME_ARRAYS_SIZE: int
taxpayerStatus: String
Taxpayer(String,int,double,String,int[],double[],double[])
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
getTaxpayerStatus():String
calculateBasicTax():double
-taxpayerHashMap
0..*
-receiptHashMap
0..*
<<Java Class>>
Receipt
incometaxcalculator.data.management
id: int
issueDate: SimpleDateFormat
amount: float
kind: String
company: String
Receipt(int,String,float,String,String,Address)
createDate(String):SimpleDateFormat
getId():int
getIssueDate():String
getAmount():float
getKind():String
getCompany():String
getCompanyAddress():Address
-companyAddress
0..1
<<Java Class>>
Address
incometaxcalculator.data.management
country: String
city: String
street: String
number: int
Address(String,String,String,int)
getCountry():String
getCity():String
getStreet():String
getNumber():int
toString():String

Class Name: Receipt	
Responsibilities	**Collaborations**
▪ As previously this class is responsible for the creation of a receipt. ▪ As previously this class contains methods that return the data of its fields. ▪ This class now contains an SimpleDateFormat Object, implemented from java source code, instead of an user-implemented Date Class. ▪ This class now contains a company, not as an object, but as a String. Also Company class has been removed. ▪ This class now contains an Address object that was previously stored in a Company class.	▪ Taxpayer

Class Name: Taxpayer

Responsibilities	**Collaborations**
▪ This class is no more an abstract class, as calculateBasicTax() method is implemented. ▪ This class now contains three arrays , that are used in parametrization of calculateBasicTax() algorithm. These arrays are defined in a Factory. ▪ This class now contains the taxpayerStatus field that defines the type of Taxpayer, because Taxpayer subclasses have been removed. ▪ This class now has more simplified addRecceipt(), removeReceipt() and getVariationTaxOnReceipts() methods. In these methods, a fixed array is used for parametrization. ▪ As previously this class contains a Hashmap in which the receipts are stored. ▪ As previously this class is responsible for adding or removing a receipt for a	▪ Receipt ▪ TaxpayerManager ▪ TaxpayerFactory

Taxpayer. ▪ As previously this class contains methods that return the data of its fields and also a method which calculates the tax based on the receipts.	

Class Name: TaxpayerFactory	
Responsibilities	**Collaborations**
▪ This class contains a HashMap, that stores the values of parametrization for the calculateBasicTax() method for every type of Taxpayer. ▪ This class creates a Taxpayer with the given name, number and income, and also the right arrays for parametrization.	▪ Taxpayer ▪ TaxpayerManager

Class Name: TaxpayerManager	
Responsibilities	**Collaborations**
▪ This class now has a more simplified version of updateFiles(), saveLogFile() and loadTaxpayer(). Factories are used for the creation	▪ Receipt

of every object these methods use. ▪ This class contains two HashMaps, one for the Taxpayers and one for the receipts. ▪ This class now uses a factory to create a Taxpayer. As previously this class includes a method that creates Taxpayer objects and adds them to the HashMap and also a method that removes a Taxpayer from the HashMap. ▪ This class includes a method that creates a receipt and adds it to the HashMap and a method that removes a receipt from the HashMap. ▪ This class contains a method for updating files when a change takes place and also a method that saves data in a file and a method that loads data from file. ▪ This class contains a method that checks if a Taxpayer exists, a method that checks if a receipt exists and some methods that return the data of	▪ Company ▪ MyFileReader ▪ MyFileWriter ▪ InfoWriterFactory ▪ LogWriter ▪ LogWriterFactory ▪ MyFileReaderFactory ▪ MyFileReader ▪ TaxpayerFactory ▪ Taxpayer

its fields.	

Class Name: LogWriterFactory	
Responsibilities	**Collaborations**
▪ This class creates a LogWriter in the right format and returns it.	▪ TXTLogWriter ▪ XMLLogWriter ▪ TaxpayerManager

Class Name: InfoWriterFactory	
Responsibilities	**Collaborations**
▪ This class creates an InfoWriter in the right format.	▪ XMLInfoWriter ▪ TXTInfoWriter ▪ TaxpayerManager

Class Name: MyFileReaderFactory	
Responsibilities	**Collaborations**
▪ This class creates an MyFileReader in the right format.	▪ TXTFileReader ▪ XMLFileReader ▪ TaxpayerManager

Class Name: TXTFileReader	
Responsibilities	**Collaborations**
▪ This class now has a simple logic. It just implements steps of template methods of the new MyFileReader class. ▪ Checks if a receipt is valid. ▪ Returns a value important for parsing. ▪ Returns some processed String.	▪ MyFileReader

Class Name: XMLFileReader	
Responsibilities	**Collaborations**
▪ Same as TXTFileReader, parametrized for xml type of files. ▪ Checks if a receipt is valid. ▪ Returns a value important for parsing. ▪ Returns some processed String.	▪ MyFileReader

Class Name: MyFileReader	
Responsibilities	**Collaborations**
▪ This class is responsible for reading some information from file for a Taxpayer and the receipts of these informations. ▪ This class now has implementation for checkForReceipt() and getValueOfField() methods. These methods use some abstract methods, implemented for various types of files.	▪ TaxpayerManager

Class Name: MyFileWriter	
Responsibilities	**Collaborations**
▪ This class now is just an interface, and it's previous methods are used with a TaxpayerManager object when	▪ TaxpayerManager

they are in need. ▪ This class now has an abstract method, for file creation.	

Class Name: TXTInfoWriter	
Responsibilities	**Collaborations**
▪ This class now has a simple logic. It just implements steps of template methods of the new InfoWriter class. ▪ Returns a type of file as a String. ▪ Returns the right begin tag of a file format. ▪ Returns the right end tag of a file format.	▪ InfoWriter

Class Name: XMLInfoWriter	
Responsibilities	**Collaborations**
▪ This class now has a simple	▪ InfoWriter

logic. It just implements steps of template methods of the new InfoWriter class. ▪ Returns a type of file as a String. ▪ Returns the right begin tag of a file format. ▪ Returns the right end tag of a file format.	

Class Name: InfoWriter	
Responsibilities	**Collaborations**
▪ This class now generates a file with the write tags. These tags are given by two simple abstract template methods form the right class. ▪ This class now calculates the receipts, that it uses for file creation. Again it uses tags that are given from the right class by some abstract methods.	▪ MyFileWriter ▪ TaxpayerManager

Class Name: TXTLogWriter	
Responsibilities	**Collaborations**
▪ This class now has a simple logic. It just implements steps of template methods of the new LogWriter class. ▪ Returns a type of file as a String. ▪ Returns the right begin tag of a file format. ▪ Returns the right end tag of a file format. ▪ Returns a ReceiptBegin tag, as it is a special case of tag.	▪ LogWriter

Class Name: XMLLogWriter	
Responsibilities	**Collaborations**
▪ This class now has a simple logic. It just implements steps of template methods of the new LogWriter class. ▪ Returns a type of file as a	▪ LogWriter

String. ▪ Returns the right begin tag of a file format. ▪ Returns the right end tag of a file format. ▪ Returns a ReceiptBegin tag, as it is a special case of tag.	

Class Name: LogWriter	
Responsibilities	**Collaborations**
▪ This class now generates a file with the right tags, given from of its subclasses. The subclasses implement some simple abstract methods, that this class stores.	▪ `MyFileWriter` ▪ `TaxpayerManager`

ΣΗΜΕΙΩΣΗ

Τα τεστ δεν περιλαμβάνονται στην αναφορά. Τροποποιήθηκαν με βάση τις αλλαγές που έγιναν και περιλαμβάνονται στο package tests στον κώδικα.