MINNESOTA INCOME TAX CALCULATOR REFACTORING

OVERALL REPORT

VERSION <1.0>

GRIGORIOS TSOPOURIDIS, AM:3358

IOANNIS TARASIDIS, AM:3342

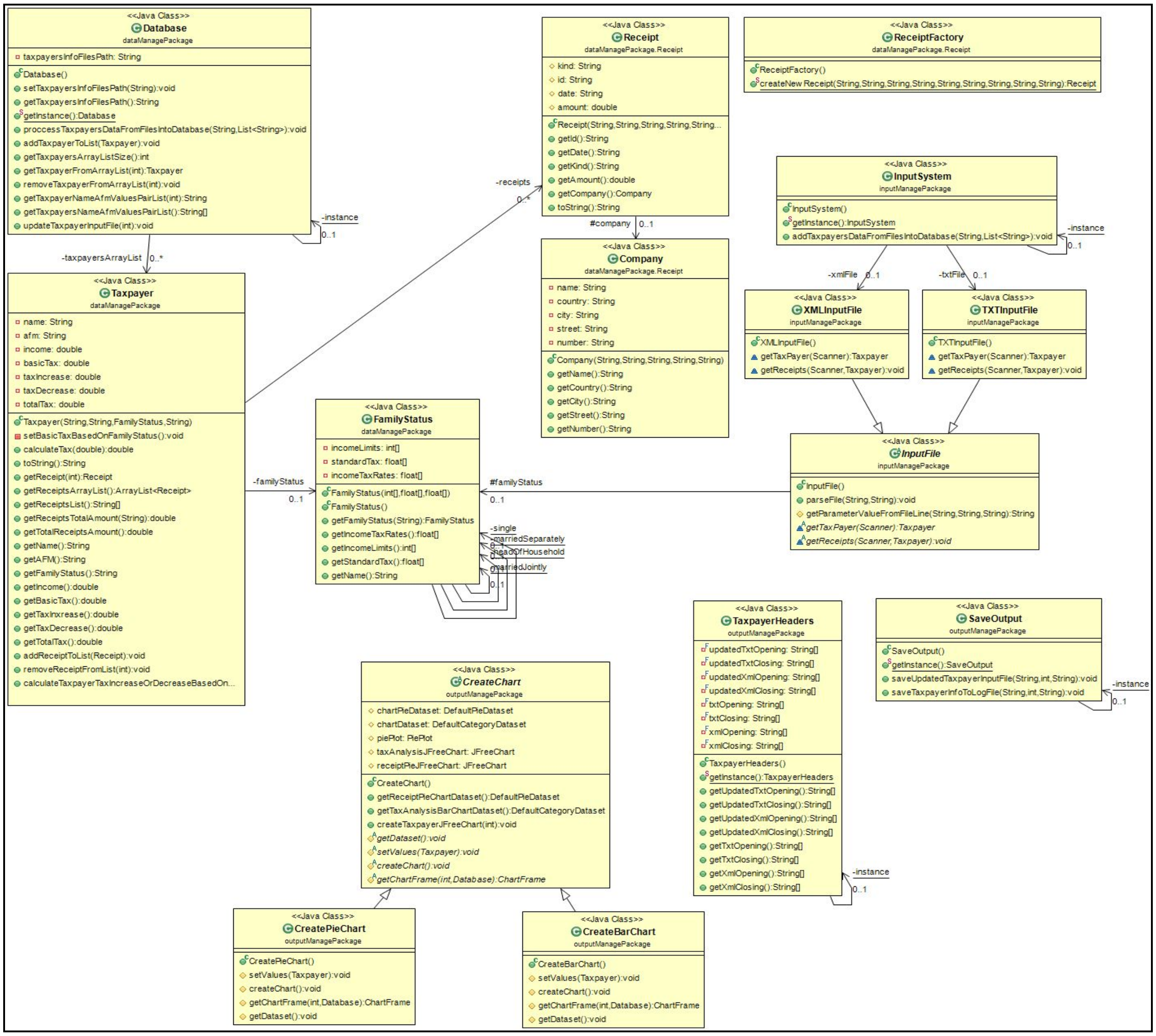
TABLE OF CONTENTS

INTRODUCTION

The goal of this phase is to refactor the Java application. At first we create a couple of unit tests for the basic classes, Database, Taxpayer and InputSystem which we later refactor. Our aim is to remove code duplication, primitive obsession, trivial subclasses and any other smell that may appear.

REFACTORED DESIGN

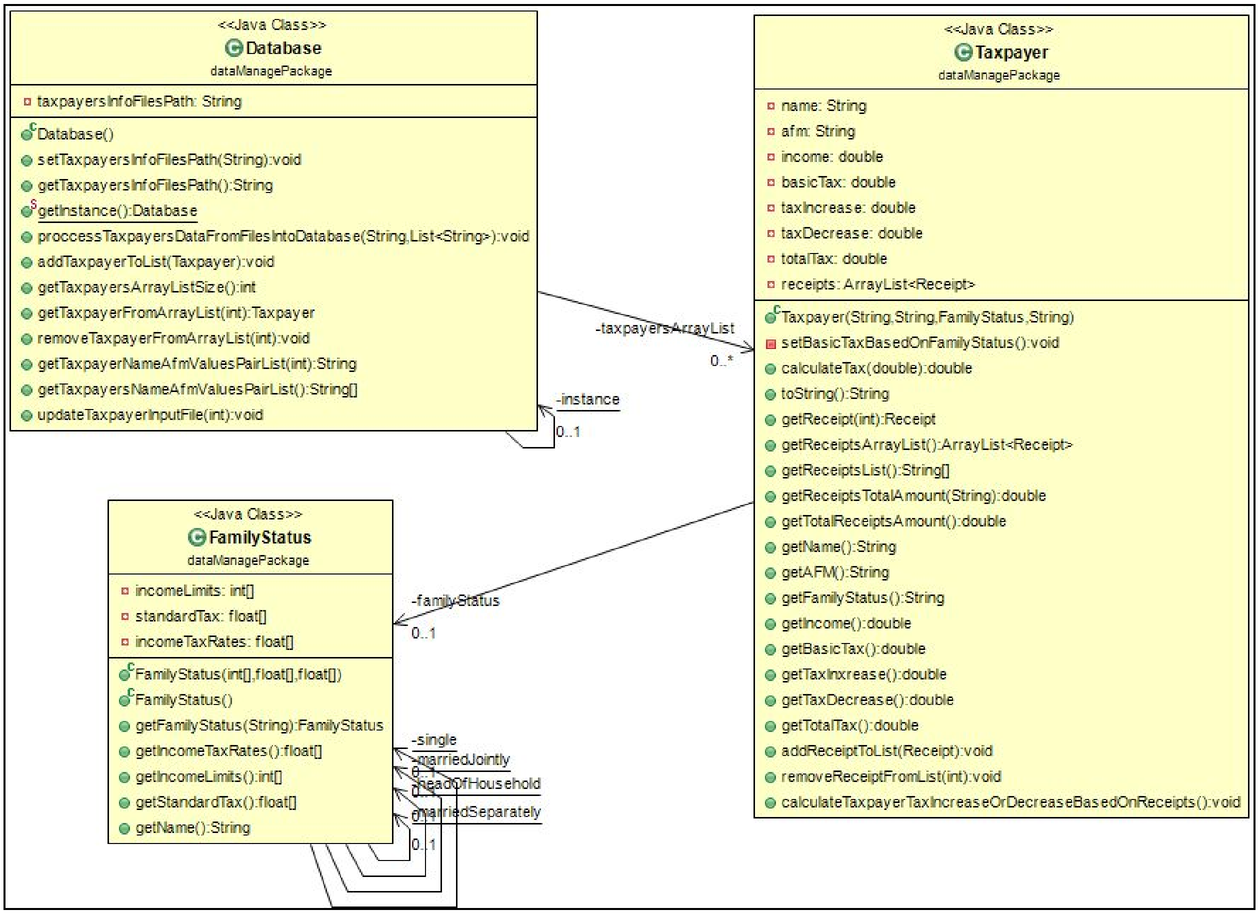
ARCHITECTURE



<<Java Class>>
Database
dataManagePackage
taxpayersInfoFilesPath: String
Database()
setTaxpayersInfoFilesPath(String):void
getTaxpayersInfoFilesPath():String
getInstance():Database
proccessTaxpayersDataFromFilesIntoDatabase(String,List<String>):void
addTaxpayerToList(Taxpayer):void
getTaxpayersArrayListSize():int
getTaxpayerFromArrayList(int):Taxpayer
removeTaxpayerFromArrayList(int):void
getTaxpayerNameAfmValuesPairList(int):String
getTaxpayersNameAfmValuesPairList():String[]
updateTaxpayerInputFile(int):void
-instance
0..1
-taxpayersArrayList
0..*
<<Java Class>>
Taxpayer
dataManagePackage
name: String
afm: String
income: double
basicTax: double
taxIncrease: double
taxDecrease: double
totalTax: double
Taxpayer(String,String,FamilyStatus,String)
setBasicTaxBasedOnFamilyStatus():void
calculateTax(double):double
toString():String
getReceipt(int):Receipt
getReceiptsArrayList():ArrayList<Receipt>
getReceiptsList():String[]
getReceiptsTotalAmount(String):double
getTotalReceiptsAmount():double
getName():String
getAFM():String
getFamilyStatus():String
getIncome():double
getBasicTax():double
getTaxInxrease():double
getTaxDecrease():double
getTotalTax():double
addReceiptToList(Receipt):void
removeReceiptFromList(int):void
calculateTaxpayerTaxIncreaseOrDecreaseBasedOn...
-receipts
0..*
<<Java Class>>
Receipt
dataManagePackage.Receipt
kind: String
id: String
date: String
amount: double
Receipt(String,String,String,String,String...
getId():String
getDate():String
getKind():String
getAmount():double
getCompany():Company
toString():String
#company
0..1
<<Java Class>>
Company
dataManagePackage.Receipt
name: String
country: String
city: String
street: String
number: String
Company(String,String,String,String,String)
getName():String
getCountry():String
getCity():String
getStreet():String
getNumber():String
<<Java Class>>
ReceiptFactory
dataManagePackage.Receipt
ReceiptFactory()
createNew Receipt(String,String,String,String,String,String,String,String,String):Receipt
<<Java Class>>
InputSystem
inputManagePackage
InputSystem()
getInstance():InputSystem
addTaxpayersDataFromFilesIntoDatabase(String,List<String>):void
-instance
0..1
-xmlFile
0..1
-txtFile
0..1
<<Java Class>>
XMLInputFile
inputManagePackage
XMLInputFile()
getTaxPayer(Scanner):Taxpayer
getReceipts(Scanner,Taxpayer):void
<<Java Class>>
TXTInputFile
inputManagePackage
TXTInputFile()
getTaxPayer(Scanner):Taxpayer
getReceipts(Scanner,Taxpayer):void
-familyStatus
0..1
<<Java Class>>
FamilyStatus
dataManagePackage
incomeLimits: int[]
standardTax: float[]
incomeTaxRates: float[]
FamilyStatus(int[],float[],float[])
FamilyStatus()
getFamilyStatus(String):FamilyStatus
getIncomeTaxRates():float[]
getIncomeLimits():int[]
getStandardTax():float[]
getName():String
-single
-marriedSeparately
-headOfHousehold
-marriedJointly
0..1
0..1
0..1
0..1
#familyStatus
0..1
<<Java Class>>
InputFile
inputManagePackage
InputFile()
parseFile(String,String):void
getParameterValueFromFileLine(String,String,String):String
getTaxPayer(Scanner):Taxpayer
getReceipts(Scanner,Taxpayer):void
<<Java Class>>
TaxpayerHeaders
outputManagePackage
updatedTxtOpening: String[]
updatedTxtClosing: String[]
updatedXmlOpening: String[]
updatedXmlClosing: String[]
txtOpening: String[]
txtClosing: String[]
xmlOpening: String[]
xmlClosing: String[]
TaxpayerHeaders()
getInstance():TaxpayerHeaders
getUpdatedTxtOpening():String[]
getUpdatedTxtClosing():String[]
getUpdatedXmlOpening():String[]
getUpdatedXmlClosing():String[]
getTxtOpening():String[]
getTxtClosing():String[]
getXmlOpening():String[]
getXmlClosing():String[]
-instance
0..1
<<Java Class>>
SaveOutput
outputManagePackage
SaveOutput()
getInstance():SaveOutput
saveUpdatedTaxpayerInputFile(String,int,String):void
saveTaxpayerInfoToLogFile(String,int,String):void
-instance
0..1
<<Java Class>>
CreateChart
outputManagePackage
chartPieDataset: DefaultPieDataset
chartDataset: DefaultCategoryDataset
piePlot: PiePlot
taxAnalysisJFreeChart: JFreeChart
receiptPieJFreeChart: JFreeChart
CreateChart()
getReceiptPieChartDataset():DefaultPieDataset
getTaxAnalysisBarChartDataset():DefaultCategoryDataset
createTaxpayerJFreeChart(int):void
getDataset():void
setValues(Taxpayer):void
createChart():void
getChartFrame(int,Database):ChartFrame
<<Java Class>>
CreatePieChart
outputManagePackage
CreatePieChart()
setValues(Taxpayer):void
createChart():void
getChartFrame(int,Database):ChartFrame
getDataset():void
<<Java Class>>
CreateBarChart
outputManagePackage
CreateBarChart()
setValues(Taxpayer):void
createChart():void
getChartFrame(int,Database):ChartFrame
getDataset():void

DETAILED DESIGN

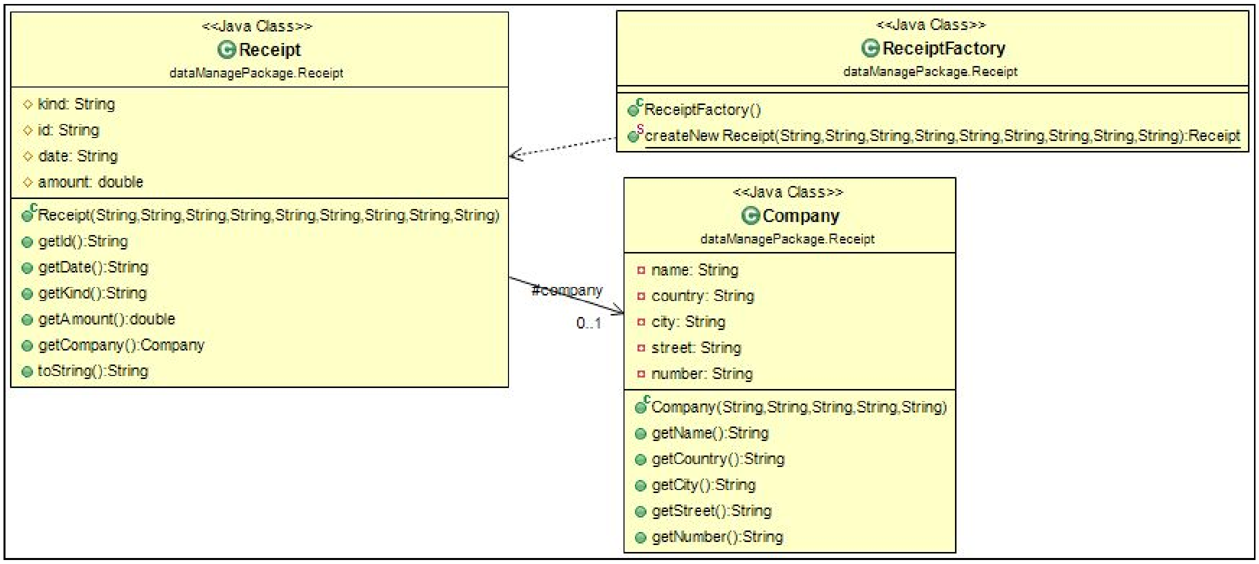
- dataManagePackage:


<<Java Class>>
Database
dataManagePackage
taxpayersInfoFilesPath: String
Database()
setTaxpayersInfoFilesPath(String):void
getTaxpayersInfoFilesPath():String
getInstance():Database
proccessTaxpayersDataFromFilesIntoDatabase(String,List<String>):void
addTaxpayerToList(Taxpayer):void
getTaxpayersArrayListSize():int
getTaxpayerFromArrayList(int):Taxpayer
removeTaxpayerFromArrayList(int):void
getTaxpayerNameAfmValuesPairList(int):String
getTaxpayersNameAfmValuesPairList():String[]
updateTaxpayerInputFile(int):void
-instance
0..1
-taxpayersArrayList
0..*
<<Java Class>>
Taxpayer
dataManagePackage
name: String
afm: String
income: double
basicTax: double
taxIncrease: double
taxDecrease: double
totalTax: double
receipts: ArrayList<Receipt>
Taxpayer(String,String,FamilyStatus,String)
setBasicTaxBasedOnFamilyStatus():void
calculateTax(double):double
toString():String
getReceipt(int):Receipt
getReceiptsArrayList():ArrayList<Receipt>
getReceiptsList():String[]
getReceiptsTotalAmount(String):double
getTotalReceiptsAmount():double
getName():String
getAFM():String
getFamilyStatus():String
getIncome():double
getBasicTax():double
getTaxInxrease():double
getTaxDecrease():double
getTotalTax():double
addReceiptToList(Receipt):void
removeReceiptFromList(int):void
calculateTaxpayerTaxIncreaseOrDecreaseBasedOnReceipts():void
-familyStatus
0..1
<<Java Class>>
FamilyStatus
dataManagePackage
incomeLimits: int[]
standardTax: float[]
incomeTaxRates: float[]
FamilyStatus(int[],float[],float[])
FamilyStatus()
getFamilyStatus(String):FamilyStatus
getIncomeTaxRates():float[]
getIncomeLimits():int[]
getStandardTax():float[]
getName():String
-single
-marriedJointly
-headOfHousehold
-marriedSeparately
0..1
0..1
0..1
0..1

We removed code duplication from Taxpayer using parameterize-method. We removed primitive obsession from Taxpayer ,for family status specifically, using “Replace Typecode with Class” by creating a new FamilyStatus class containing all the details about each tax family status.

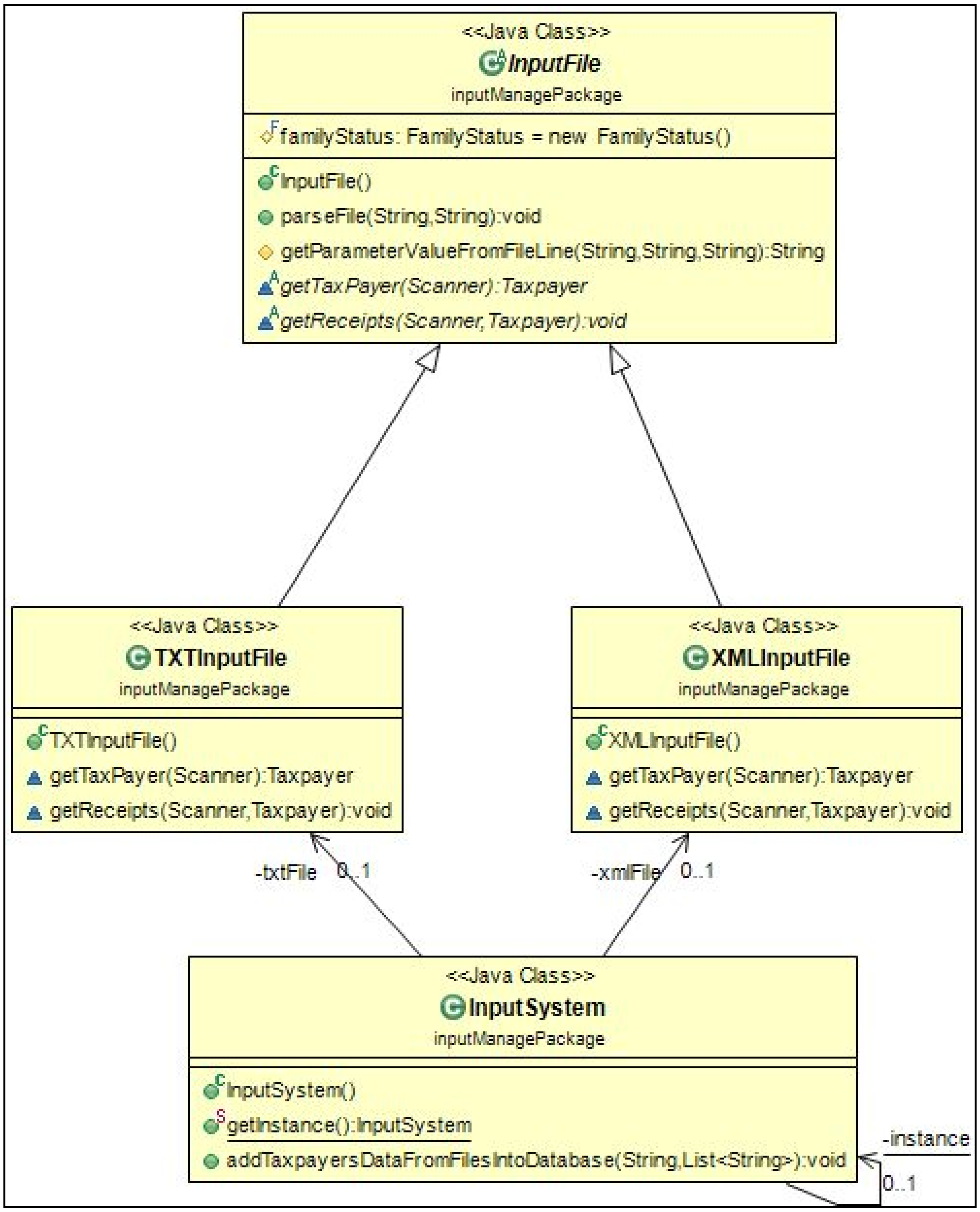
We refactored Database class into OOP style by refactoring using singleton pattern instead of using static methods.

- dataManagePackage.Receipt:



Instead of many trivial subclasses of receipt (lazy classes) we moved all their features into Receipt class thus generalising the design for every class (in-line class). Additionally, we created a Factory Pattern for generating different receipt types.

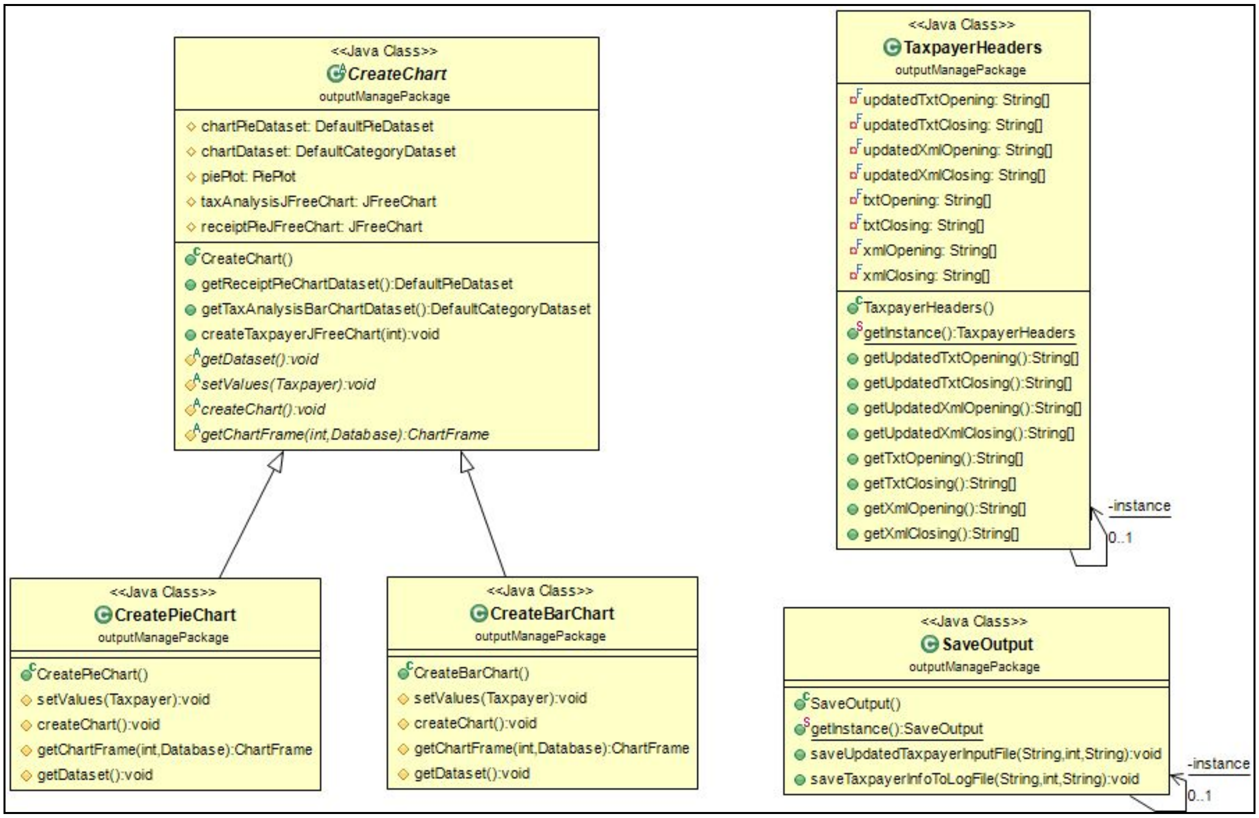
- inputManagePackage:



<<Java Class>>
InputFile
inputManagePackage
familyStatus: FamilyStatus = new FamilyStatus()
InputFile()
parseFile(String,String):void
getParameterValueFromFileLine(String,String,String):String
getTaxPayer(Scanner):Taxpayer
getReceipts(Scanner,Taxpayer):void
<<Java Class>>
TXTInputFile
inputManagePackage
TXTInputFile()
getTaxPayer(Scanner):Taxpayer
getReceipts(Scanner,Taxpayer):void
<<Java Class>>
XMLInputFile
inputManagePackage
XMLInputFile()
getTaxPayer(Scanner):Taxpayer
getReceipts(Scanner,Taxpayer):void
-txtFile
0..1
-xmlFile
0..1
<<Java Class>>
InputSystem
inputManagePackage
InputSystem()
getInstance():InputSystem
addTaxpayersDataFromFilesIntoDatabase(String,List<String>):void
-instance
0..1

We refactored InputSystem class into OOP style by refactoring using singleton pattern instead of using static methods. We split the parsing (txt or xml) in different classes as InputSystem had too many responsibilities. We used InputFile class as a template (Form Template method) and created a class for each file type, TXTInputFile and XMLInputFile, inheriting from our pattern InputFile.

- outPutManagePackage:


<<Java Class>>
CreateChart
outputManagePackage
chartPieDataset: DefaultPieDataset
chartDataset: DefaultCategoryDataset
piePlot: PiePlot
taxAnalysisJFreeChart: JFreeChart
receiptPieJFreeChart: JFreeChart
CreateChart()
getReceiptPieChartDataset():DefaultPieDataset
getTaxAnalysisBarChartDataset():DefaultCategoryDataset
createTaxpayerJFreeChart(int):void
getDataset():void
setValues(Taxpayer):void
createChart():void
getChartFrame(int,Database):ChartFrame
<<Java Class>>
TaxpayerHeaders
outputManagePackage
updatedTxtOpening: String[]
updatedTxtClosing: String[]
updatedXmlOpening: String[]
updatedXmlClosing: String[]
txtOpening: String[]
txtClosing: String[]
xmlOpening: String[]
xmlClosing: String[]
TaxpayerHeaders()
getInstance():TaxpayerHeaders
getUpdatedTxtOpening():String[]
getUpdatedTxtClosing():String[]
getUpdatedXmlOpening():String[]
getUpdatedXmlClosing():String[]
getTxtOpening():String[]
getTxtClosing():String[]
getXmlOpening():String[]
getXmlClosing():String[]
-instance
0..1
<<Java Class>>
CreatePieChart
outputManagePackage
CreatePieChart()
setValues(Taxpayer):void
createChart():void
getChartFrame(int,Database):ChartFrame
getDataset():void
<<Java Class>>
CreateBarChart
outputManagePackage
CreateBarChart()
setValues(Taxpayer):void
createChart():void
getChartFrame(int,Database):ChartFrame
getDataset():void
<<Java Class>>
SaveOutput
outputManagePackage
SaveOutput()
getInstance():SaveOutput
saveUpdatedTaxpayerInputFile(String,int,String):void
saveTaxpayerInfoToLogFile(String,int,String):void
-instance
0..1

We split OutputSystem class into 3 classes. TaxPayerHeaders uses "Replace Typecode with Class" and contains the different coding styles of each file. SaveOutput is responsible for file creation and Create is responsible works as a template for CreatePieChart and CreateBarChart. CreatePieChart and CreateBarChart are responsible for pie and bar charts respectively and inherit from CreateChart template.

ALL UMLS WILL BE INCLUDED IN THE FINAL DELIVERABLE!

IMPLEMENTATION

Class Name: Database	
Responsibilities	**Collaborations**
This class is responsible for Taxpayer data storage, keeping and getting.	Taxpayer Class

Class Name: FamilyStatus	
Responsibilities	**Collaborations**
This class is responsible for storing all the data related to each family status tax rates and income values.	

Class Name: Taxpayer	
Responsibilities	**Collaborations**
This class is responsible for creating all Taxpayer data, calculating taxes and managing receipts.	FamilyStatus

Class Name: Receipt	
Responsibilities	**Collaborations**
This class is responsible for handling all Receipt related data and their accessors.	Company

Class Name: ReceiptFactory	
Responsibilities	**Collaborations**
This class is responsible for creating different version of Receipt.	Receipt

Class Name: Company	
Responsibilities	**Collaborations**
This class is responsible for handling all company related data and their accessors.	

Class Name: InputFile	
Responsibilities	**Collaborations**
This class is a template, implementing only common code for file opening between its children and has a role as their specification.	

Class Name: InputSystem	
Responsibilities	**Collaborations**
This class is responsible for handling all input related functions, importing a Taxpayer from file and loading it into the Database.	TXTInputFile XMLInputFile

Class Name: XMLInputFile	
Responsibilities	**Collaborations**
This class inherits basic functions from InputFile and works as a parser for .xml taxpayer files.	InputFile

Class Name: TXTInputFile	
Responsibilities	**Collaborations**
This class inherits basic functions from InputFile and works as a parser for .txt taxpayer files.	InputFile

Class Name: TaxpayerHeaders	
Responsibilities	**Collaborations**
This class contains basic data (headers) for .txt and .xml file output.	

Class Name: SaveOutput	
Responsibilities	**Collaborations**
This class exports Taxpayer data to .xml or .txt file.	

Class Name: CreateChart	
Responsibilities	**Collaborations**
This class is a template, implementing only common code on Chart creation between its children and has a role as their specification.	

Class Name: CreatePieChart	
Responsibilities	**Collaborations**
This class inherits basic functions from CreateChart and creates Pie charts.	CreateChart

Class Name: CreateBarChart	
Responsibilities	**Collaborations**
This class inherits basic functions from CreateChart and creates Bar charts.	CreateChart