<FINAL VERSION>

FINAL REFACTOR OF THE CODE

VERSION <1.0>

Ιορδάνης-Ραφαήλ Φυδανάκης AM:420

Ευάγγελος Δημουλής AM:421

-

TABLE OF CONTENTS

TABLE OF CONTENTS

INTRODUCTION

Under Construction

You can find the source code of the project in GitHub at:

https://github.com/JordanRaphael/l1-software-data-evolution

REFACTORING

REFACTORING

Final Changes

Populated GlobalDataManager with logic methods, removed DataManipulator, now everything goes through GlobalDataManager.

<TBD> Created Interface/Factory at DataKeeper

We followed the these ideas as we moved on with the refactoring of the code:

The project needs test to verify the integrity of the code.

There is a lot of code duplication, some factory classes would help.

The Gui class is a god class and does a lot of things, logic needs to be moved out.

The GlobalDataKeeper class has no logic, the logic needs to move closer to Data.

The code has unnecessary coupling and complicated logic, it needs to be fixed.

The refactorings applied to the different stages of the project involved:

- Moving the gui.treeElements package to the data hyper-package, renaming it to data.treeElements.

- Moving the gui.tableElements.tableConstructors package to the data hyper-package, renaming it to data.tableConstructors.

- Moving all listeners related to the General Table to a separate class called GeneralTableListenerHandler that handles Action Listeners initialization.

- Moving all listeners related to the Zoom Table to a separate class called ZoomTableListenerHandler that handles Action Listeners initialization.

- Moving data logic from BusinessLogic infoAction() to GlobalDataKeeper printInfo().

- Moving data logic from BusinessLogic showClusterSelectionZoomArea() to GlobalDataKeeper showClusterSelectionZoomArea().

- Moving Constructors of tableConstruction inside GlobalDataKeeper (e.g. TableConstructionPhases).

- Moving two population methods (populateWithPhases, populateWithClusters) inside GlobalDataKeeper.

- Extracted the Action Listeners to separate classes
- Added class JItemsCreator which is responsible for the creation and initialization of JItems

- The methods that were handling the functionality Create Project, Edit Project, Load Project, Show PLD, Show Phases PLD, Show Phases PLD with Clusters, Show Full Detailed Lifetime Table, Zoom In, Zoom out, Set Data

- Extracted code from the forementioned methods which depended on GlobalDataKeeper e.g. pupulateWithPhases, populatewithClusters

- Moved TableConstructors inside the GlobalDataKeeper to avoid passing the GlobalDataKeeper object every time as a parameter to these constructors

- Renamed the BusinessLogic class to GuiController

- Broke the Cycles at the Data Package Diagram and Gui Class Diagram
- Split GlobalDataKeeper to 6 classes. 4 classes that store data, 1 class that handles the processing of the data and 1 GlobalDataManager.

PATTERNS USED

A variation of Factor out state at Gui Package

Facade at GlobalDataManager for the data container methods

TESTING

TESTS

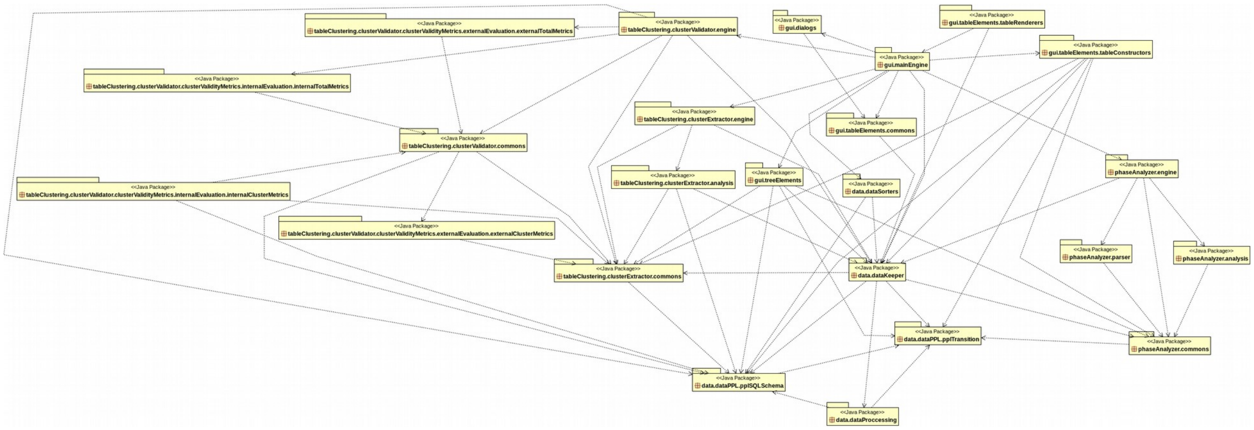
Tests were developed in order to test that changes to the project, won’t affect it’s functionality. The logic behind the tests were, find out the expected result in a specific scenario and try to recreate it with the new code changes. If it matches the functionality was intact, else something needs to be checked over.

List of tests implemented for the project:

- testCreateProject
- testEditProject
- testLoadProject
- testZoomIn
- testZoomOut
- testSetData
- testShowPLD
- testShowPhasesPLD
- testShowPhasesWithClusters
- testShowFullDetailedLifetimeTable

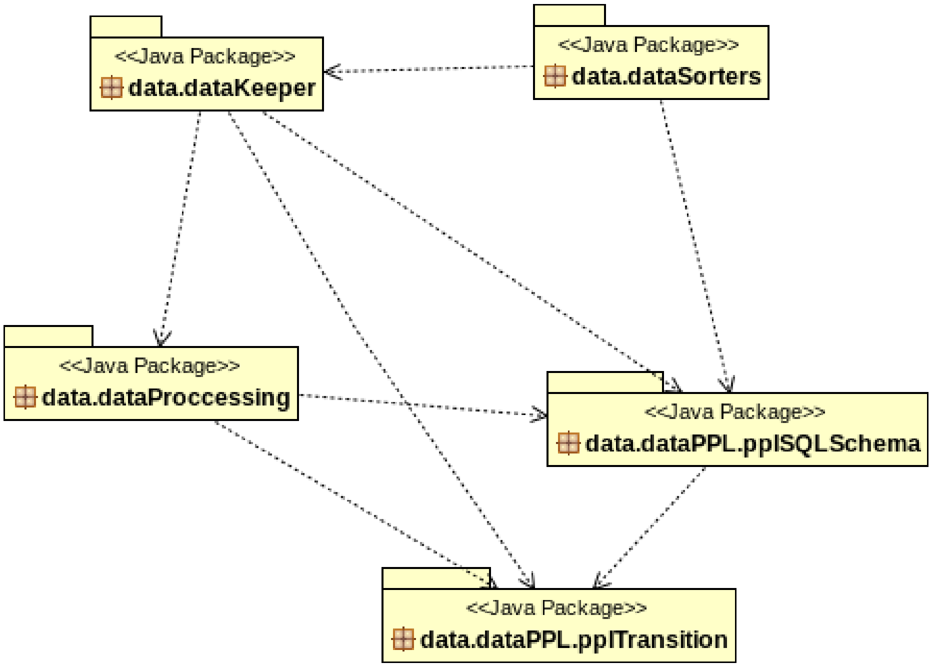
UML DIAGRAMS

Project hyperpackages

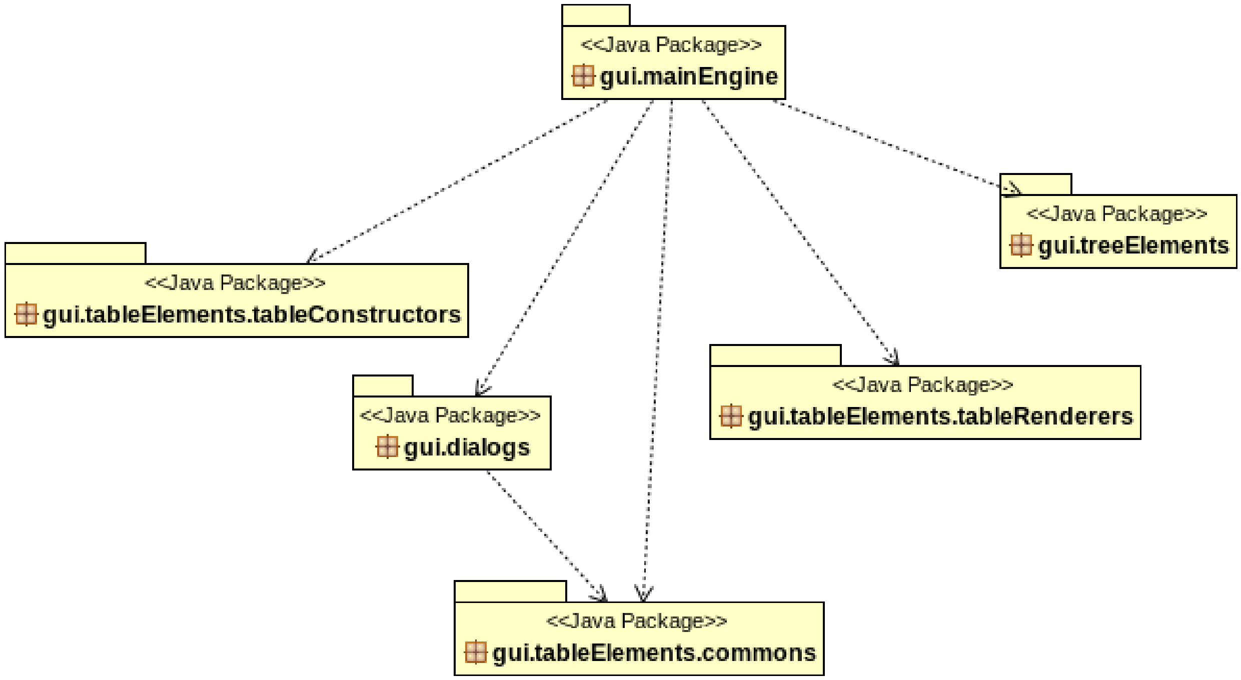


<<Java Package>>
tableClustering.clusterValidator.clusterValidityMetrics.externalEvaluation.externalTotalMetrics
<<Java Package>>
tableClustering.clusterValidator.engine
<<Java Package>>
gui.dialogs
<<Java Package>>
gui.tableElements.tableRenderers
<<Java Package>>
gui.mainEngine
<<Java Package>>
gui.tableElements.tableConstructors
<<Java Package>>
tableClustering.clusterValidator.clusterValidityMetrics.internalEvaluation.internalTotalMetrics
<<Java Package>>
tableClustering.clusterExtractor.engine
<<Java Package>>
gui.tableElements.commons
<<Java Package>>
tableClustering.clusterValidator.commons
<<Java Package>>
phaseAnalyzer.engine
<<Java Package>>
tableClustering.clusterExtractor.analysis
<<Java Package>>
gui.treeElements
<<Java Package>>
data.dataSorters
<<Java Package>>
tableClustering.clusterValidator.clusterValidityMetrics.internalEvaluation.internalClusterMetrics
<<Java Package>>
tableClustering.clusterValidator.clusterValidityMetrics.externalEvaluation.externalClusterMetrics
<<Java Package>>
phaseAnalyzer.parser
<<Java Package>>
phaseAnalyzer.analysis
<<Java Package>>
tableClustering.clusterExtractor.commons
<<Java Package>>
data.dataKeeper
<<Java Package>>
data.dataPPL.pplTransition
<<Java Package>>
phaseAnalyzer.commons
<<Java Package>>
data.dataPPL.pplSQLSchema
<<Java Package>>
data.dataProccessing

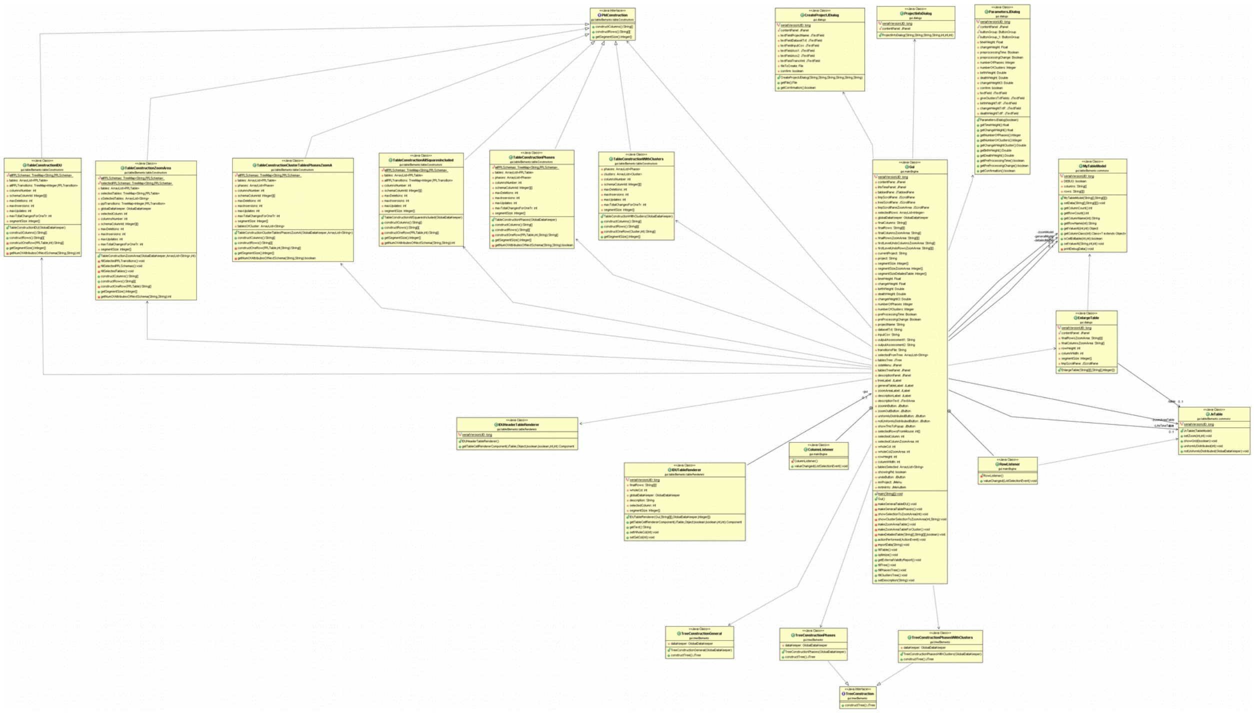
Data Hyperpackage (moved 2 more packages)



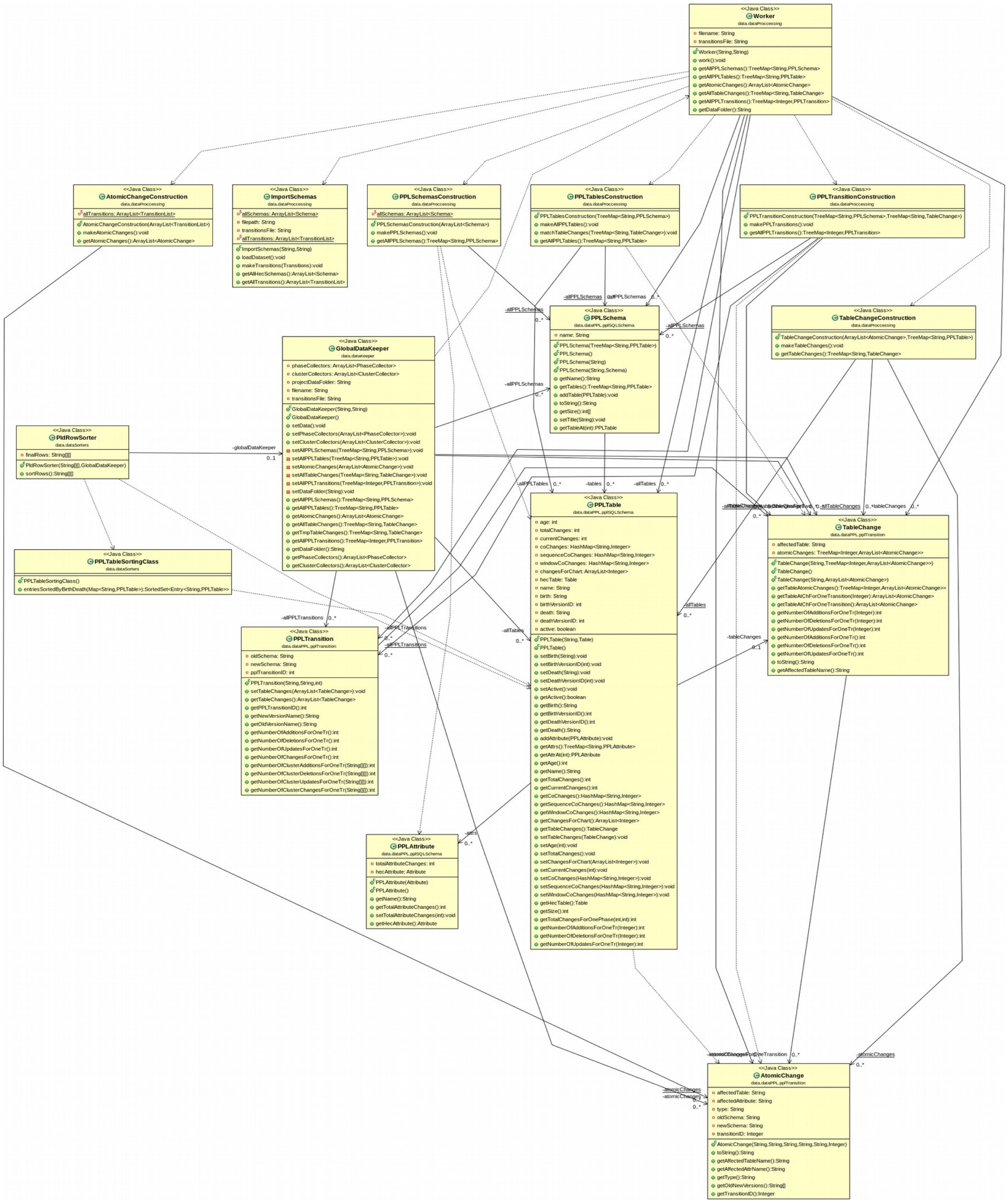
Gui Hyperpackage



Gui class diagram



Data class diagram



<<Java Class>>
Worker
data.dataProccessing
filename: String
transitionsFile: String
Worker(String,String)
work():void
getAllPPLSchemas():TreeMap<String,PPLSchema>
getAllPPLTables():TreeMap<String,PPLTable>
getAtomicChanges():ArrayList<AtomicChange>
getAllTableChanges():TreeMap<String,TableChange>
getAllPPLTransitions():TreeMap<Integer,PPLTransition>
getDataFolder():String
<<Java Class>>
AtomicChangeConstruction
data.dataProccessing
allTransitions: ArrayList<TransitionList>
AtomicChangeConstruction(ArrayList<TransitionList>)
makeAtomicChanges():void
getAtomicChanges():ArrayList<AtomicChange>
<<Java Class>>
ImportSchemas
data.dataProccessing
allSchemas: ArrayList<Schema>
filepath: String
transitionsFile: String
allTransitions: ArrayList<TransitionList>
ImportSchemas(String,String)
loadDataset():void
makeTransitions(Transitions):void
getAllHecSchemas():ArrayList<Schema>
getAllTransitions():ArrayList<TransitionList>
<<Java Class>>
PPLSchemasConstruction
data.dataProccessing
allSchemas: ArrayList<Schema>
PPLSchemasConstruction(ArrayList<Schema>)
makePPLSchemas():void
getAllPPLSchemas():TreeMap<String,PPLSchema>
<<Java Class>>
PPLTablesConstruction
data.dataProccessing
PPLTablesConstruction(TreeMap<String,PPLSchema>)
makeAllPPLTables():void
matchTableChanges(TreeMap<String,TableChange>):void
getAllPPLTables():TreeMap<String,PPLTable>
<<Java Class>>
PPLTransitionConstruction
data.dataProccessing
PPLTransitionConstruction(TreeMap<String,PPLSchema>,TreeMap<String,TableChange>)
makePPLTransitions():void
getAllPPLTransitions():TreeMap<Integer,PPLTransition>
-allPPLSchemas
0..*
-allPPLSchemas
0..*
-allPPLSchemas
0..*
-allPPLSchemas
0..*
-allPPLSchemas
0..*
<<Java Class>>
PPLSchema
data.dataPPL.pplSQLSchema
name: String
PPLSchema(TreeMap<String,PPLTable>)
PPLSchema()
PPLSchema(String)
PPLSchema(String,Schema)
getName():String
getTables():TreeMap<String,PPLTable>
addTable(PPLTable):void
toString():String
getSize():int[]
setTitle(String):void
getTableAt(int):PPLTable
<<Java Class>>
TableChangeConstruction
data.dataProccessing
TableChangeConstruction(ArrayList<AtomicChange>,TreeMap<String,PPLTable>)
makeTableChanges():void
getTableChanges():TreeMap<String,TableChange>
<<Java Class>>
GlobalDataKeeper
data.dataKeeper
phaseCollectors: ArrayList<PhaseCollector>
clusterCollectors: ArrayList<ClusterCollector>
projectDataFolder: String
filename: String
transitionsFile: String
GlobalDataKeeper(String,String)
GlobalDataKeeper()
setData():void
setPhaseCollectors(ArrayList<PhaseCollector>):void
setClusterCollectors(ArrayList<ClusterCollector>):void
setAllPPLSchemas(TreeMap<String,PPLSchema>):void
setAllPPLTables(TreeMap<String,PPLTable>):void
setAtomicChanges(ArrayList<AtomicChange>):void
setAllTableChanges(TreeMap<String,TableChange>):void
setAllPPLTransitions(TreeMap<Integer,PPLTransition>):void
setDataFolder(String):void
getAllPPLSchemas():TreeMap<String,PPLSchema>
getAllPPLTables():TreeMap<String,PPLTable>
getAtomicChanges():ArrayList<AtomicChange>
getAllTableChanges():TreeMap<String,TableChange>
getTmpTableChanges():TreeMap<String,TableChange>
getAllPPLTransitions():TreeMap<Integer,PPLTransition>
getDataFolder():String
getPhaseCollectors():ArrayList<PhaseCollector>
getClusterCollectors():ArrayList<ClusterCollector>
<<Java Class>>
PldRowSorter
data.dataSorters
finalRows: String[][]
PldRowSorter(String[][],GlobalDataKeeper)
sortRows():String[][]
-globalDataKeeper
0..1
<<Java Class>>
PPLTableSortingClass
data.dataSorters
PPLTableSortingClass()
entriesSortedByBirthDeath(Map<String,PPLTable>):SortedSet<Entry<String,PPLTable>>
-allPPLTables
0..*
-tables
0..*
-allTables
0..*
<<Java Class>>
PPLTable
data.dataPPL.pplSQLSchema
age: int
totalChanges: int
currentChanges: int
coChanges: HashMap<String,Integer>
sequenceCoChanges: HashMap<String,Integer>
windowCoChanges: HashMap<String,Integer>
changesForChart: ArrayList<Integer>
hecTable: Table
name: String
birth: String
birthVersionID: int
death: String
deathVersionID: int
active: boolean
PPLTable(String,Table)
PPLTable()
setBirth(String):void
setBirthVersionID(int):void
setDeath(String):void
setDeathVersionID(int):void
setActive():void
getActive():boolean
getBirth():String
getBirthVersionID():int
getDeathVersionID():int
getDeath():String
addAttribute(PPLAttribute):void
getAttrs():TreeMap<String,PPLAttribute>
getAttrAt(int):PPLAttribute
getAge():int
getName():String
getTotalChanges():int
getCurrentChanges():int
getCoChanges():HashMap<String,Integer>
getSequenceCoChanges():HashMap<String,Integer>
getWindowCoChanges():HashMap<String,Integer>
getChangesForChart():ArrayList<Integer>
getTableChanges():TableChange
setTableChanges(TableChange):void
setAge(int):void
setTotalChanges():void
setChangesForChart(ArrayList<Integer>):void
setCurrentChanges(int):void
setCoChanges(HashMap<String,Integer>):void
setSequenceCoChanges(HashMap<String,Integer>):void
setWindowCoChanges(HashMap<String,Integer>):void
getHecTable():Table
getSize():int
getTotalChangesForOnePhase(int,int):int
getNumberOfAdditionsForOneTr(Integer):int
getNumberOfDeletionsForOneTr(Integer):int
getNumberOfUpdatesForOneTr(Integer):int
-allTableChanges
0..*
-tableChanges
0..*
0..*
<<Java Class>>
TableChange
data.dataPPL.pplTransition
affectedTable: String
atomicChanges: TreeMap<Integer,ArrayList<AtomicChange>>
TableChange(String,TreeMap<Integer,ArrayList<AtomicChange>>)
TableChange()
TableChange(String,ArrayList<AtomicChange>)
getTableAtomicChanges():TreeMap<Integer,ArrayList<AtomicChange>>
getTableAtChForOneTransition(Integer):ArrayList<AtomicChange>
getTableAtChForOneTransition():ArrayList<AtomicChange>
getNumberOfAdditionsForOneTr(Integer):int
getNumberOfDeletionsForOneTr(Integer):int
getNumberOfUpdatesForOneTr(Integer):int
getNumberOfAdditionsForOneTr():int
getNumberOfDeletionsForOneTr():int
getNumberOfUpdatesForOneTr():int
toString():String
getAffectedTableName():String
-allTables
0..*
-allTables
0..*
-tableChanges
0..1
-allPPLTransitions
0..*
-allPPLTransitions
0..*
-allPPLTransitions
0..*
<<Java Class>>
PPLTransition
data.dataPPL.pplTransition
oldSchema: String
newSchema: String
pplTransitionID: int
PPLTransition(String,String,int)
setTableChanges(ArrayList<TableChange>):void
getTableChanges():ArrayList<TableChange>
getPPLTransitionID():int
getNewVersionName():String
getOldVersionName():String
getNumberOfAdditionsForOneTr():int
getNumberOfDeletionsForOneTr():int
getNumberOfUpdatesForOneTr():int
getNumberOfChangesForOneTr():int
getNumberOfClusterAdditionsForOneTr(String[][]):int
getNumberOfClusterDeletionsForOneTr(String[][]):int
getNumberOfClusterUpdatesForOneTr(String[][]):int
getNumberOfClusterChangesForOneTr(String[][]):int
-attrs
0..*
<<Java Class>>
PPLAttribute
data.dataPPL.pplSQLSchema
totalAttributeChanges: int
hecAttribute: Attribute
PPLAttribute(Attribute)
PPLAttribute()
getName():String
getTotalAttributeChanges():int
setTotalAttributeChanges(int):void
getHecAttribute():Attribute
0..*
-atomicChanges
0..*
-atomicChanges
-atomicChanges
0..*
0..*
<<Java Class>>
AtomicChange
data.dataPPL.pplTransition
affectedTable: String
affectedAttribute: String
type: String
oldSchema: String
newSchema: String
transitionID: Integer
AtomicChange(String,String,String,String,String,Integer)
toString():String
getAffectedTableName():String
getAffectedAttrName():String
getType():String
getOldNewVersions():String[]
getTransitionID():Integer

CODE QUALITY

The code had complicated if statements which was extracted to methods.

The identation of the code was refactored.

Many variables and class names were misleading or just bad, they was name refactoring as well.

Also, some methods was extremely long, they were split into smaller methods or even classes.

Some methods could be moved to another class or package, so there was method extraction.

A LOT of code was duplicate, so quite some time was given to removing duplication code. (some of the duplications were hard to locate)