# JDBC REPORT

CS221 (Programming - 2)

Hossam Fawzy elsafty (24)
Saeed Hamdy Mahmoud Hassan (31)
Amr Mohamed Fathy Mohamed (49)
Arsanuos essa Attia (18)

Including User Guide

# TABLE OF CONTENTS

## PROBLEM STATMENT

SQL:

A Computer Database is a structured collection of records or data that is stored in a computer system. On the other hand, a Database Management System (DBMS) is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. DBMS are categorized according to their data structures or types. The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data. On the other hand, Extensible Markup Language (XML) is a set of rules for encoding documents in machine readable form. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all gratis open standards.

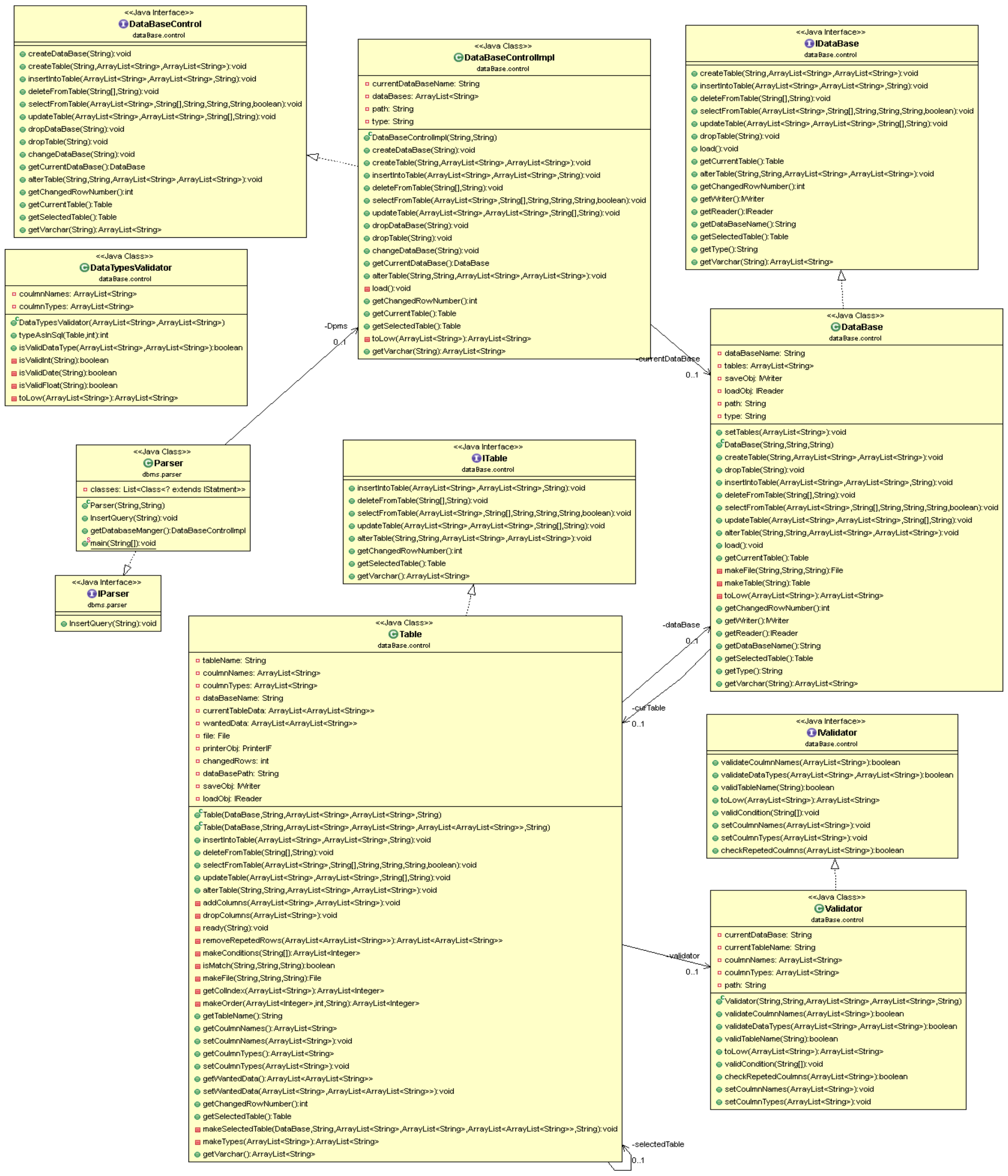
JDBC**:**

Java Database Connectivity (JDBC) provides Java developers with a standard API that is used to access databases, regardless of the driver and database product. JDBC presents a uniform interface to databases - change vendors and your applications only need to change their driver.

# DESIGN

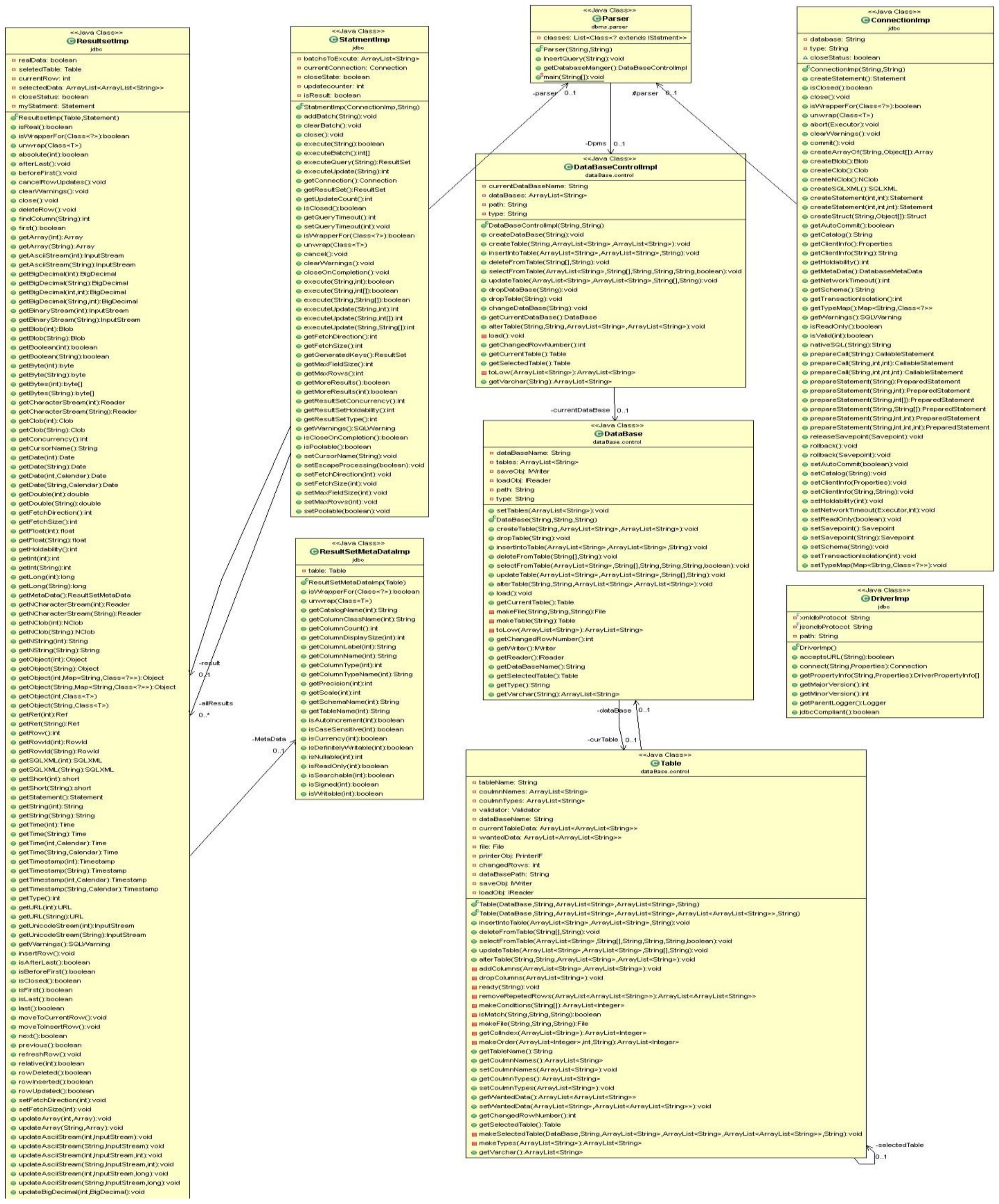
## UML

UML diagram for the DBMS.



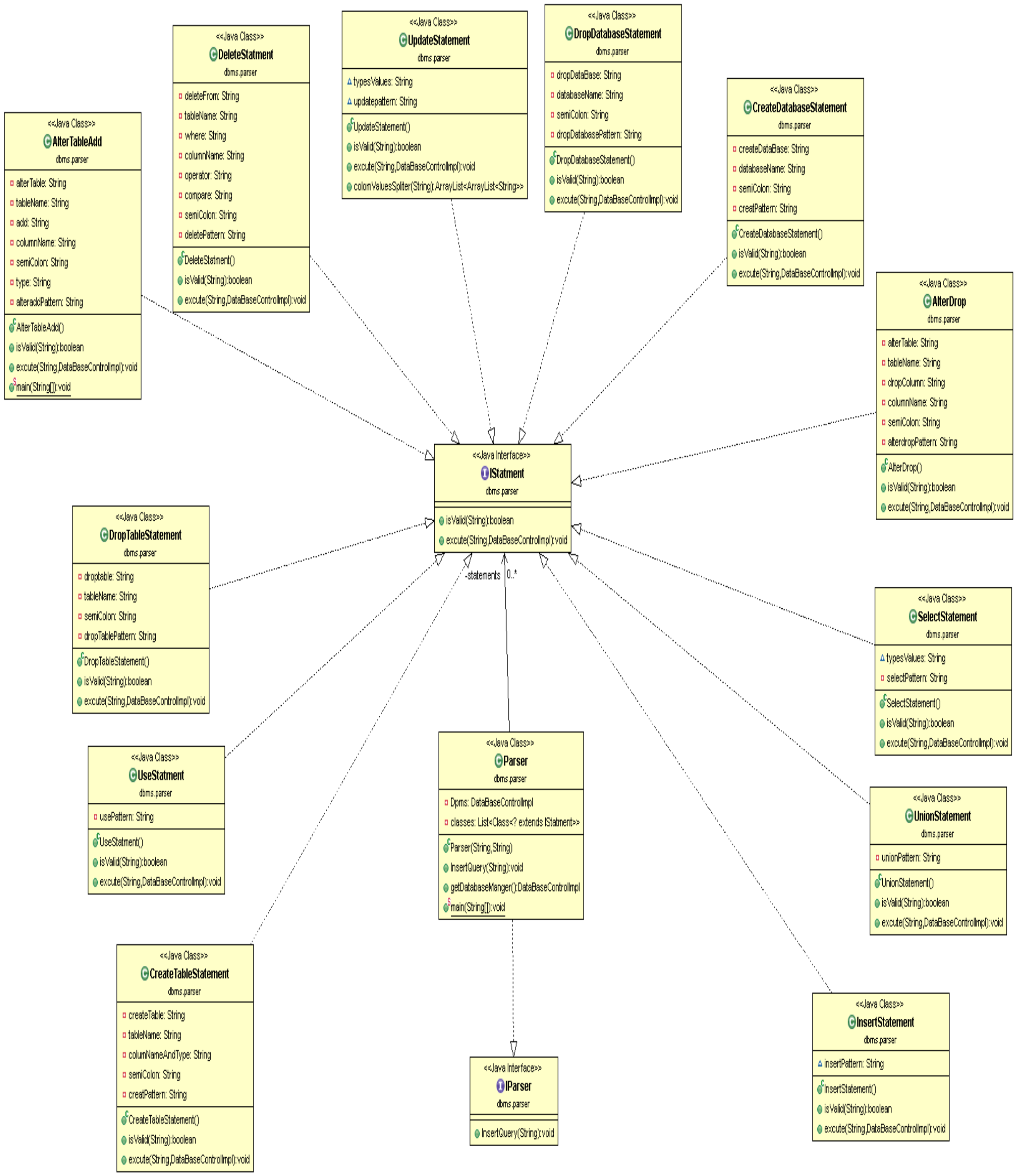
<<Java Interface>>
DataBaseControl
dataBase.control
createDataBase(String):void
createTable(String,ArrayList<String>,ArrayList<String>):void
insertIntoTable(ArrayList<String>,ArrayList<String>,String):void
deleteFromTable(String[],String):void
selectFromTable(ArrayList<String>,String[],String,String,String,boolean):void
updateTable(ArrayList<String>,ArrayList<String>,String[],String):void
dropDataBase(String):void
dropTable(String):void
changeDataBase(String):void
getCurrentDataBase():DataBase
alterTable(String,String,ArrayList<String>,ArrayList<String>):void
getChangedRowNumber():int
getCurrentTable():Table
getSelectedTable():Table
getVarchar(String):ArrayList<String>
<<Java Class>>
DataBaseControlImpl
dataBase.control
currentDataBaseName: String
dataBases: ArrayList<String>
path: String
type: String
DataBaseControlImpl(String,String)
createDataBase(String):void
createTable(String,ArrayList<String>,ArrayList<String>):void
insertIntoTable(ArrayList<String>,ArrayList<String>,String):void
deleteFromTable(String[],String):void
selectFromTable(ArrayList<String>,String[],String,String,String,boolean):void
updateTable(ArrayList<String>,ArrayList<String>,String[],String):void
dropDataBase(String):void
dropTable(String):void
changeDataBase(String):void
getCurrentDataBase():DataBase
alterTable(String,String,ArrayList<String>,ArrayList<String>):void
load():void
getChangedRowNumber():int
getCurrentTable():Table
getSelectedTable():Table
toLow(ArrayList<String>):ArrayList<String>
getVarchar(String):ArrayList<String>
<<Java Interface>>
IDataBase
dataBase.control
createTable(String,ArrayList<String>,ArrayList<String>):void
insertIntoTable(ArrayList<String>,ArrayList<String>,String):void
deleteFromTable(String[],String):void
selectFromTable(ArrayList<String>,String[],String,String,String,boolean):void
updateTable(ArrayList<String>,ArrayList<String>,String[],String):void
dropTable(String):void
load():void
getCurrentTable():Table
alterTable(String,String,ArrayList<String>,ArrayList<String>):void
getChangedRowNumber():int
getWriter():IWriter
getReader():IReader
getDataBaseName():String
getSelectedTable():Table
getType():String
getVarchar(String):ArrayList<String>
<<Java Class>>
DataTypesValidator
dataBase.control
coulmnNames: ArrayList<String>
coulmnTypes: ArrayList<String>
DataTypesValidator(ArrayList<String>,ArrayList<String>)
typeAsInSql(Table,int):int
isValidDataType(ArrayList<String>,ArrayList<String>):boolean
isValidInt(String):boolean
isValidDate(String):boolean
isValidFloat(String):boolean
toLow(ArrayList<String>):ArrayList<String>
-Dpms
0..1
-currentDataBase
0..1
<<Java Class>>
DataBase
dataBase.control
dataBaseName: String
tables: ArrayList<String>
saveObj: IWriter
loadObj: IReader
path: String
type: String
setTables(ArrayList<String>):void
DataBase(String,String,String)
createTable(String,ArrayList<String>,ArrayList<String>):void
dropTable(String):void
insertIntoTable(ArrayList<String>,ArrayList<String>,String):void
deleteFromTable(String[],String):void
selectFromTable(ArrayList<String>,String[],String,String,String,boolean):void
updateTable(ArrayList<String>,ArrayList<String>,String[],String):void
alterTable(String,String,ArrayList<String>,ArrayList<String>):void
load():void
getCurrentTable():Table
makeFile(String,String,String):File
makeTable(String):Table
toLow(ArrayList<String>):ArrayList<String>
getChangedRowNumber():int
getWriter():IWriter
getReader():IReader
getDataBaseName():String
getSelectedTable():Table
getType():String
getVarchar(String):ArrayList<String>
<<Java Class>>
Parser
dbms.parser
classes: List<Class<? extends IStatment>>
Parser(String,String)
InsertQuery(String):void
getDatabaseManger():DataBaseControlImpl
main(String[]):void
<<Java Interface>>
ITable
dataBase.control
insertIntoTable(ArrayList<String>,ArrayList<String>,String):void
deleteFromTable(String[],String):void
selectFromTable(ArrayList<String>,String[],String,String,String,boolean):void
updateTable(ArrayList<String>,ArrayList<String>,String[],String):void
alterTable(String,String,ArrayList<String>,ArrayList<String>):void
getChangedRowNumber():int
getSelectedTable():Table
getVarchar():ArrayList<String>
<<Java Interface>>
IParser
dbms.parser
InsertQuery(String):void
<<Java Class>>
Table
dataBase.control
tableName: String
coulmnNames: ArrayList<String>
coulmnTypes: ArrayList<String>
dataBaseName: String
currentTableData: ArrayList<ArrayList<String>>
wantedData: ArrayList<ArrayList<String>>
file: File
printerObj: PrinterIF
changedRows: int
dataBasePath: String
saveObj: IWriter
loadObj: IReader
Table(DataBase,String,ArrayList<String>,ArrayList<String>,String)
Table(DataBase,String,ArrayList<String>,ArrayList<String>,ArrayList<ArrayList<String>>,String)
insertIntoTable(ArrayList<String>,ArrayList<String>,String):void
deleteFromTable(String[],String):void
selectFromTable(ArrayList<String>,String[],String,String,String,boolean):void
updateTable(ArrayList<String>,ArrayList<String>,String[],String):void
alterTable(String,String,ArrayList<String>,ArrayList<String>):void
addColumns(ArrayList<String>,ArrayList<String>):void
dropColumns(ArrayList<String>):void
ready(String):void
removeRepetedRows(ArrayList<ArrayList<String>>):ArrayList<ArrayList<String>>
makeConditions(String[]):ArrayList<Integer>
isMatch(String,String,String):boolean
makeFile(String,String,String):File
getColIndex(ArrayList<String>):ArrayList<Integer>
makeOrder(ArrayList<Integer>,int,String):ArrayList<Integer>
getTableName():String
getCoulmnNames():ArrayList<String>
setCoulmnNames(ArrayList<String>):void
getCoulmnTypes():ArrayList<String>
setCoulmnTypes(ArrayList<String>):void
getWantedData():ArrayList<ArrayList<String>>
setWantedData(ArrayList<String>,ArrayList<ArrayList<String>>):void
getChangedRowNumber():int
getSelectedTable():Table
makeSelectedTable(DataBase,String,ArrayList<String>,ArrayList<String>,ArrayList<ArrayList<String>>,String):void
makeTypes(ArrayList<String>):ArrayList<String>
getVarchar():ArrayList<String>
-dataBase
0..1
-curTable
0..1
<<Java Interface>>
IValidator
dataBase.control
validateCoulmnNames(ArrayList<String>):boolean
validateDataTypes(ArrayList<String>,ArrayList<String>):boolean
validTableName(String):boolean
toLow(ArrayList<String>):ArrayList<String>
validCondition(String[]):void
setCoulmnNames(ArrayList<String>):void
setCoulmnTypes(ArrayList<String>):void
checkRepetedCoulmns(ArrayList<String>):boolean
<<Java Class>>
Validator
dataBase.control
currentDataBase: String
currentTableName: String
coulmnNames: ArrayList<String>
coulmnTypes: ArrayList<String>
path: String
Validator(String,String,ArrayList<String>,ArrayList<String>,String)
validateCoulmnNames(ArrayList<String>):boolean
validateDataTypes(ArrayList<String>,ArrayList<String>):boolean
validTableName(String):boolean
toLow(ArrayList<String>):ArrayList<String>
validCondition(String[]):void
checkRepetedCoulmns(ArrayList<String>):boolean
setCoulmnNames(ArrayList<String>):void
setCoulmnTypes(ArrayList<String>):void
-validator
0..1
-selectedTable
0..1

UML diagram for JDBC:

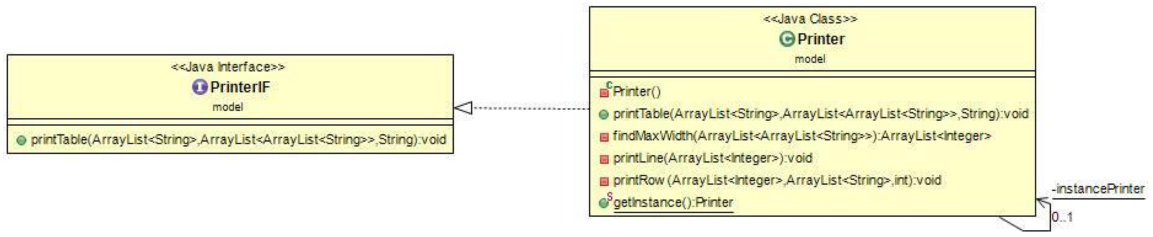


<<Java Class>> ResultsetImp jdbc
realData: boolean
seletedTable: Table
currentRow: int
selectedData: ArrayList<ArrayList<String>>
closeStatus: boolean
myStatment: Statement
ResultsetImp(Table,Statement)
isReal():boolean
isWrapperFor(Class<?>):boolean
unwrap(Class<T>)
absolute(int):boolean
afterLast():void
beforeFirst():void
cancelRowUpdates():void
clearWarnings():void
close():void
deleteRow():void
findColumn(String):int
first():boolean
getArray(int):Array
getArray(String):Array
getAsciiStream(int):InputStream
getAsciiStream(String):InputStream
getBigDecimal(int):BigDecimal
getBigDecimal(String):BigDecimal
getBigDecimal(int,int):BigDecimal
getBigDecimal(String,int):BigDecimal
getBinaryStream(int):InputStream
getBinaryStream(String):InputStream
getBlob(int):Blob
getBlob(String):Blob
getBoolean(int):boolean
getBoolean(String):boolean
getByte(int):byte
getByte(String):byte
getBytes(int):byte[]
getBytes(String):byte[]
getCharacterStream(int):Reader
getCharacterStream(String):Reader
getClob(int):Clob
getClob(String):Clob
getConcurrency():int
getCursorName():String
getDate(int):Date
getDate(String):Date
getDate(int,Calendar):Date
getDate(String,Calendar):Date
getDouble(int):double
getDouble(String):double
getFetchDirection():int
getFetchSize():int
getFloat(int):float
getFloat(String):float
getHoldability():int
getInt(int):int
getInt(String):int
getLong(int):long
getLong(String):long
getMetaData():ResultSetMetaData
getNCharacterStream(int):Reader
getNCharacterStream(String):Reader
getNClob(int):NClob
getNClob(String):NClob
getNString(int):String
getNString(String):String
getObject(int):Object
getObject(String):Object
getObject(int,Map<String,Class<?>>):Object
getObject(String,Map<String,Class<?>>):Object
getObject(int,Class<T>)
getObject(String,Class<T>)
getRef(int):Ref
getRef(String):Ref
getRow():int
getRowId(int):RowId
getRowId(String):RowId
getSQLXML(int):SQLXML
getSQLXML(String):SQLXML
getShort(int):short
getShort(String):short
getStatement():Statement
getString(int):String
getString(String):String
getTime(int):Time
getTime(String):Time
getTime(int,Calendar):Time
getTime(String,Calendar):Time
getTimestamp(int):Timestamp
getTimestamp(String):Timestamp
getTimestamp(int,Calendar):Timestamp
getTimestamp(String,Calendar):Timestamp
getType():int
getURL(int):URL
getURL(String):URL
getUnicodeStream(int):InputStream
getUnicodeStream(String):InputStream
getWarnings():SQLWarning
insertRow():void
isAfterLast():boolean
isBeforeFirst():boolean
isClosed():boolean
isFirst():boolean
isLast():boolean
last():boolean
moveToCurrentRow():void
moveToInsertRow():void
next():boolean
previous():boolean
refreshRow():void
relative(int):boolean
rowDeleted():boolean
rowInserted():boolean
rowUpdated():boolean
setFetchDirection(int):void
setFetchSize(int):void
updateArray(int,Array):void
updateArray(String,Array):void
updateAsciiStream(int,InputStream):void
updateAsciiStream(String,InputStream):void
updateAsciiStream(int,InputStream,int):void
updateAsciiStream(String,InputStream,int):void
updateAsciiStream(int,InputStream,long):void
updateAsciiStream(String,InputStream,long):void
updateBigDecimal(int,BigDecimal):void
<<Java Class>> StatmentImp jdbc
batchsToExcute: ArrayList<String>
currentConnection: Connection
closeState: boolean
updatecounter: int
isResult: boolean
StatmentImp(ConnectionImp,String)
addBatch(String):void
clearBatch():void
close():void
execute(String):boolean
executeBatch():int[]
executeQuery(String):ResultSet
executeUpdate(String):int
getConnection():Connection
getResultSet():ResultSet
getUpdateCount():int
isClosed():boolean
getQueryTimeout():int
setQueryTimeout(int):void
isWrapperFor(Class<?>):boolean
unwrap(Class<T>)
cancel():void
clearWarnings():void
closeOnCompletion():void
execute(String,int):boolean
execute(String,int[]):boolean
execute(String,String[]):boolean
executeUpdate(String,int):int
executeUpdate(String,int[]):int
executeUpdate(String,String[]):int
getFetchDirection():int
getFetchSize():int
getGeneratedKeys():ResultSet
getMaxFieldSize():int
getMaxRows():int
getMoreResults():boolean
getMoreResults(int):boolean
getResultSetConcurrency():int
getResultSetHoldability():int
getResultSetType():int
getWarnings():SQLWarning
isCloseOnCompletion():boolean
isPoolable():boolean
setCursorName(String):void
setEscapeProcessing(boolean):void
setFetchDirection(int):void
setFetchSize(int):void
setMaxFieldSize(int):void
setMaxRows(int):void
setPoolable(boolean):void
<<Java Class>> ResultSetMetaDataImp jdbc
table: Table
ResultSetMetaDataImp(Table)
isWrapperFor(Class<?>):boolean
unwrap(Class<T>)
getCatalogName(int):String
getColumnClassName(int):String
getColumnCount():int
getColumnDisplaySize(int):int
getColumnLabel(int):String
getColumnName(int):String
getColumnType(int):int
getColumnTypeName(int):String
getPrecision(int):int
getScale(int):int
getSchemaName(int):String
getTableName(int):String
isAutoIncrement(int):boolean
isCaseSensitive(int):boolean
isCurrency(int):boolean
isDefinitelyWritable(int):boolean
isNullable(int):int
isReadOnly(int):boolean
isSearchable(int):boolean
isSigned(int):boolean
isWritable(int):boolean
-result
0..1
-allResults
0..*
-MetaData
0..1
<<Java Class>> Parser dbms.parser
classes: List<Class<? extends IStatment>>
Parser(String,String)
InsertQuery(String):void
getDatabaseManger():DataBaseControlImpl
main(String[]):void
-parser 0..1
#parser 0..1
-Dpms 0..1
<<Java Class>> DataBaseControlImpl dataBase.control
currentDataBaseName: String
dataBases: ArrayList<String>
path: String
type: String
DataBaseControlImpl(String,String)
createDataBase(String):void
createTable(String,ArrayList<String>,ArrayList<String>):void
insertIntoTable(ArrayList<String>,ArrayList<String>,String):void
deleteFromTable(String[],String):void
selectFromTable(ArrayList<String>,String[],String,String,String,boolean):void
updateTable(ArrayList<String>,ArrayList<String>,String[],String):void
dropDataBase(String):void
dropTable(String):void
changeDataBase(String):void
getCurrentDataBase():DataBase
alterTable(String,String,ArrayList<String>,ArrayList<String>):void
load():void
getChangedRowNumber():int
getCurrentTable():Table
getSelectedTable():Table
toLow(ArrayList<String>):ArrayList<String>
getVarchar(String):ArrayList<String>
-currentDataBase 0..1
<<Java Class>> DataBase dataBase.control
dataBaseName: String
tables: ArrayList<String>
saveObj: IWriter
loadObj: IReader
path: String
type: String
setTables(ArrayList<String>):void
DataBase(String,String,String)
createTable(String,ArrayList<String>,ArrayList<String>):void
dropTable(String):void
insertIntoTable(ArrayList<String>,ArrayList<String>,String):void
deleteFromTable(String[],String):void
selectFromTable(ArrayList<String>,String[],String,String,String,boolean):void
updateTable(ArrayList<String>,ArrayList<String>,String[],String):void
alterTable(String,String,ArrayList<String>,ArrayList<String>):void
load():void
getCurrentTable():Table
makeFile(String,String,String):File
makeTable(String):Table
toLow(ArrayList<String>):ArrayList<String>
getChangedRowNumber():int
getWriter():IWriter
getReader():IReader
getDataBaseName():String
getSelectedTable():Table
getType():String
getVarchar(String):ArrayList<String>
-dataBase 0..1
-curTable 0..1
<<Java Class>> Table dataBase.control
tableName: String
coulmnNames: ArrayList<String>
coulmnTypes: ArrayList<String>
validator: Validator
dataBaseName: String
currentTableData: ArrayList<ArrayList<String>>
wantedData: ArrayList<ArrayList<String>>
file: File
printerObj: PrinterIF
changedRows: int
dataBasePath: String
saveObj: IWriter
loadObj: IReader
Table(DataBase,String,ArrayList<String>,ArrayList<String>,String)
Table(DataBase,String,ArrayList<String>,ArrayList<String>,ArrayList<ArrayList<String>>,String)
insertIntoTable(ArrayList<String>,ArrayList<String>,String):void
deleteFromTable(String[],String):void
selectFromTable(ArrayList<String>,String[],String,String,String,boolean):void
updateTable(ArrayList<String>,ArrayList<String>,String[],String):void
alterTable(String,String,ArrayList<String>,ArrayList<String>):void
addColumns(ArrayList<String>,ArrayList<String>):void
dropColumns(ArrayList<String>):void
ready(String):void
removeRepetedRows(ArrayList<ArrayList<String>>):ArrayList<ArrayList<String>>
makeConditions(String[]):ArrayList<Integer>
isMatch(String,String,String):boolean
makeFile(String,String,String):File
getColIndex(ArrayList<String>):ArrayList<Integer>
makeOrder(ArrayList<Integer>,int,String):ArrayList<Integer>
getTableName():String
getCoulmnNames():ArrayList<String>
setCoulmnNames(ArrayList<String>):void
getCoulmnTypes():ArrayList<String>
setCoulmnTypes(ArrayList<String>):void
getWantedData():ArrayList<ArrayList<String>>
setWantedData(ArrayList<String>,ArrayList<ArrayList<String>>):void
getChangedRowNumber():int
getSelectedTable():Table
makeSelectedTable(DataBase,String,ArrayList<String>,ArrayList<String>,ArrayList<ArrayList<String>>,String):void
makeTypes(ArrayList<String>):ArrayList<String>
getVarchar():ArrayList<String>
-selectedTable
0..1
<<Java Class>> ConnectionImp jdbc
database: String
type: String
closeStatus: boolean
ConnectionImp(String,String)
createStatement():Statement
isClosed():boolean
close():void
isWrapperFor(Class<?>):boolean
unwrap(Class<T>)
abort(Executor):void
clearWarnings():void
commit():void
createArrayOf(String,Object[]):Array
createBlob():Blob
createClob():Clob
createNClob():NClob
createSQLXML():SQLXML
createStatement(int,int):Statement
createStatement(int,int,int):Statement
createStruct(String,Object[]):Struct
getAutoCommit():boolean
getCatalog():String
getClientInfo():Properties
getClientInfo(String):String
getHoldability():int
getMetaData():DatabaseMetaData
getNetworkTimeout():int
getSchema():String
getTransactionIsolation():int
getTypeMap():Map<String,Class<?>>
getWarnings():SQLWarning
isReadOnly():boolean
isValid(int):boolean
nativeSQL(String):String
prepareCall(String):CallableStatement
prepareCall(String,int,int):CallableStatement
prepareCall(String,int,int,int):CallableStatement
prepareStatement(String):PreparedStatement
prepareStatement(String,int):PreparedStatement
prepareStatement(String,int[]):PreparedStatement
prepareStatement(String,String[]):PreparedStatement
prepareStatement(String,int,int):PreparedStatement
prepareStatement(String,int,int,int):PreparedStatement
releaseSavepoint(Savepoint):void
rollback():void
rollback(Savepoint):void
setAutoCommit(boolean):void
setCatalog(String):void
setClientInfo(Properties):void
setClientInfo(String,String):void
setHoldability(int):void
setNetworkTimeout(Executor,int):void
setReadOnly(boolean):void
setSavepoint():Savepoint
setSavepoint(String):Savepoint
setSchema(String):void
setTransactionIsolation(int):void
setTypeMap(Map<String,Class<?>>):void
<<Java Class>> DriverImp jdbc
xmldbProtocol: String
jsondbProtocol: String
path: String
DriverImp()
acceptsURL(String):boolean
connect(String,Properties):Connection
getPropertyInfo(String,Properties):DriverPropertyInfo[]
getMajorVersion():int
getMinorVersion():int
getParentLogger():Logger
jdbcCompliant():boolean

UML diagram for the Parser.

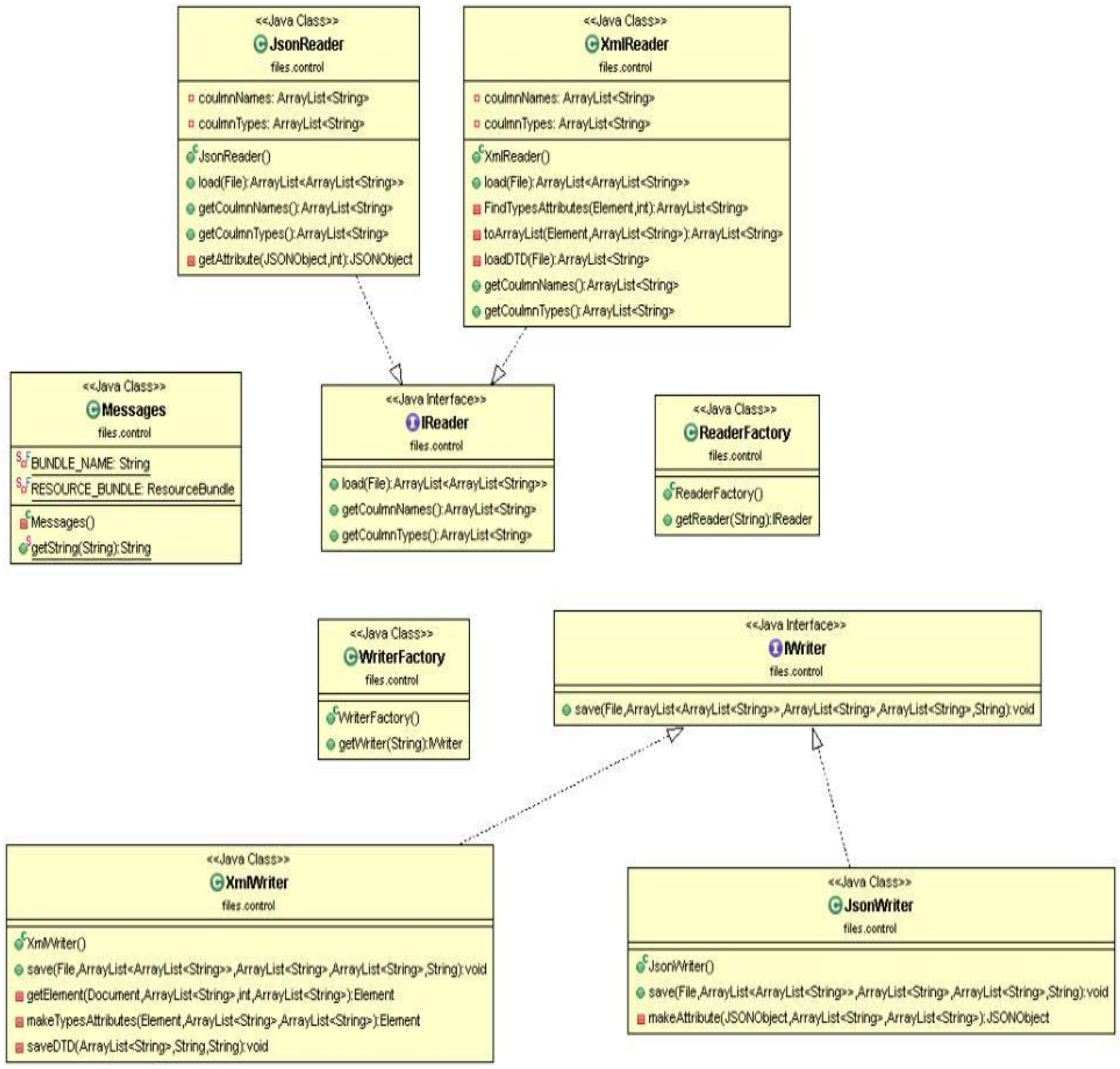


UML diagram for the Printer.



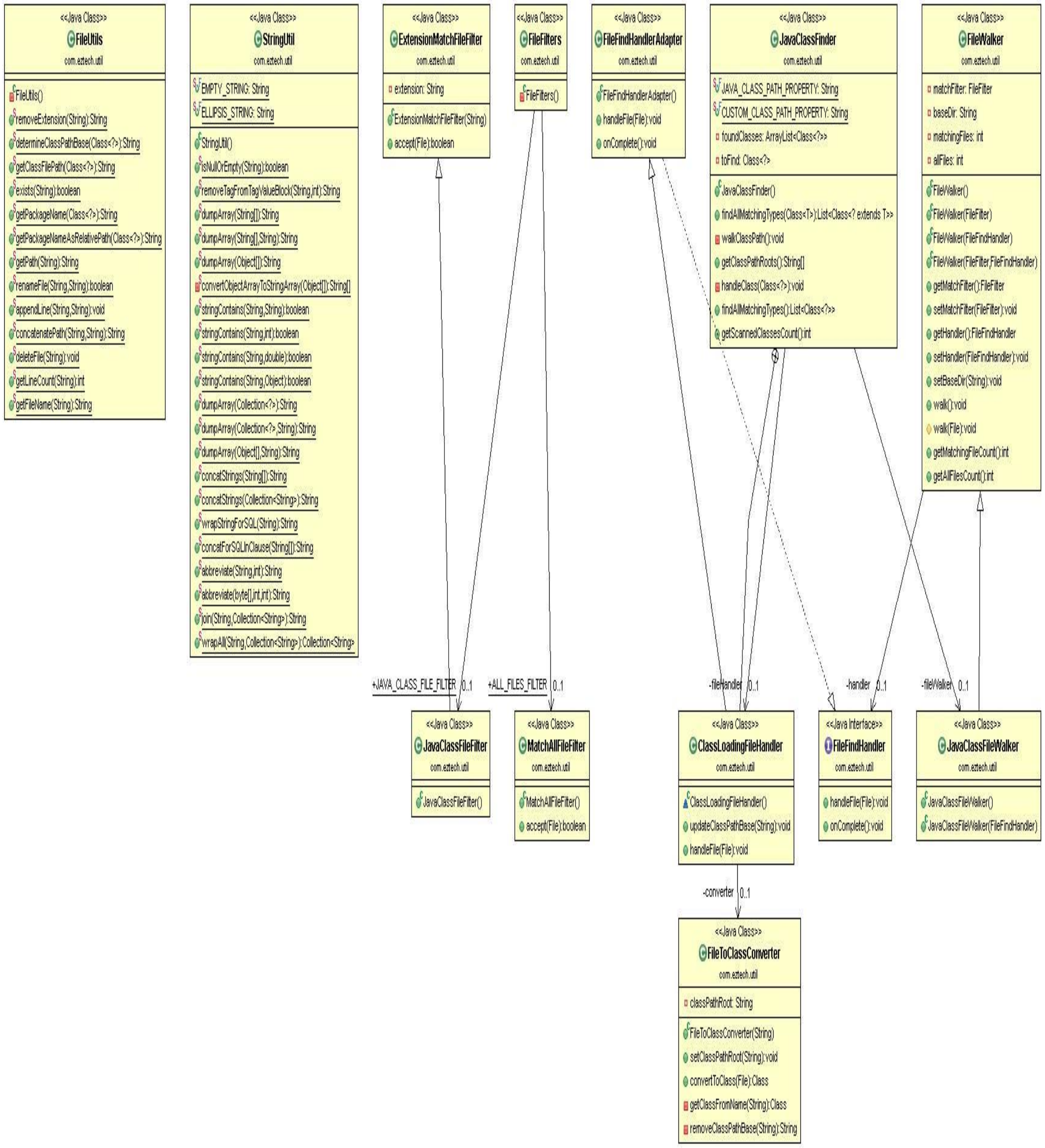
<<Java Interface>>
PrinterIF
model
printTable(ArrayList<String>,ArrayList<ArrayList<String>>,String):void
<<Java Class>>
Printer
model
Printer()
printTable(ArrayList<String>,ArrayList<ArrayList<String>>,String):void
findMaxWidth(ArrayList<ArrayList<String>>):ArrayList<Integer>
printLine(ArrayList<Integer>):void
printRow (ArrayList<Integer>,ArrayList<String>,int):void
getInstance():Printer
-instancePrinter
0..1

UML diagram for the Save and load.



<<Java Class>>
JsonReader
files.control
coulmnNames: ArrayList<String>
coulmnTypes: ArrayList<String>
JsonReader()
load(File):ArrayList<ArrayList<String>>
getCoulmnNames():ArrayList<String>
getCoulmnTypes():ArrayList<String>
getAttribute(JSONObject,int):JSONObject
<<Java Class>>
XmlReader
files.control
coulmnNames: ArrayList<String>
coulmnTypes: ArrayList<String>
XmlReader()
load(File):ArrayList<ArrayList<String>>
FindTypesAttributes(Element,int):ArrayList<String>
toArrayList(Element,ArrayList<String>):ArrayList<String>
loadDTD(File):ArrayList<String>
getCoulmnNames():ArrayList<String>
getCoulmnTypes():ArrayList<String>
<<Java Class>>
Messages
files.control
BUNDLE_NAME: String
RESOURCE_BUNDLE: ResourceBundle
Messages()
getString(String):String
<<Java Interface>>
IReader
files.control
load(File):ArrayList<ArrayList<String>>
getCoulmnNames():ArrayList<String>
getCoulmnTypes():ArrayList<String>
<<Java Class>>
ReaderFactory
files.control
ReaderFactory()
getReader(String):IReader
<<Java Class>>
WriterFactory
files.control
WriterFactory()
getWriter(String):IWriter
<<Java Interface>>
IWriter
files.control
save(File,ArrayList<ArrayList<String>>,ArrayList<String>,ArrayList<String>,String):void
<<Java Class>>
XmlWriter
files.control
XmlWriter()
save(File,ArrayList<ArrayList<String>>,ArrayList<String>,ArrayList<String>,String):void
getElement(Document,ArrayList<String>,int,ArrayList<String>):Element
makeTypesAttributes(Element,ArrayList<String>,ArrayList<String>):Element
saveDTD(ArrayList<String>,String,String):void
<<Java Class>>
JsonWriter
files.control
JsonWriter()
save(File,ArrayList<ArrayList<String>>,ArrayList<String>,ArrayList<String>,String):void
makeAttribute(JSONObject,ArrayList<String>,ArrayList<String>):JSONObject

UML diagram for the Class Finder.


<<Java Class>>
FileUtils
com.eztech.util
FileUtils()
removeExtension(String):String
determineClassPathBase(Class<?>):String
getClassFilePath(Class<?>):String
exists(String):boolean
getPackageName(Class<?>):String
getPackageNameAsRelativePath(Class<?>):String
getPath(String):String
renameFile(String,String):boolean
appendLine(String,String):void
concatenatePath(String,String):String
deleteFile(String):void
getLineCount(String):int
getFileName(String):String
<<Java Class>>
StringUtil
com.eztech.util
EMPTY_STRING: String
ELLIPSIS_STRING: String
StringUtil()
isNullOrEmpty(String):boolean
removeTagFromTagValueBlock(String,int):String
dumpArray(String[]):String
dumpArray(String[],String):String
dumpArray(Object[]):String
convertObjectArrayToStringArray(Object[]):String[]
stringContains(String,String):boolean
stringContains(String,int):boolean
stringContains(String,double):boolean
stringContains(String,Object):boolean
dumpArray(Collection<?>):String
dumpArray(Collection<?>,String):String
dumpArray(Object[],String):String
concatStrings(String[]):String
concatStrings(Collection<String>):String
wrapStringForSQL(String):String
concatForSQLInClause(String[]):String
abbreviate(String,int):String
abbreviate(byte[],int,int):String
join(String,Collection<String>):String
wrapAll(String,Collection<String>):Collection<String>
<<Java Class>>
ExtensionMatchFileFilter
com.eztech.util
extension: String
ExtensionMatchFileFilter(String)
accept(File):boolean
<<Java Class>>
FileFilters
com.eztech.util
FileFilters()
<<Java Class>>
FileFindHandlerAdapter
com.eztech.util
FileFindHandlerAdapter()
handleFile(File):void
onComplete():void
<<Java Class>>
JavaClassFinder
com.eztech.util
JAVA_CLASS_PATH_PROPERTY: String
CUSTOM_CLASS_PATH_PROPERTY: String
foundClasses: ArrayList<Class<?>>
toFind: Class<?>
JavaClassFinder()
findAllMatchingTypes(Class<T>):List<Class<? extends T>>
walkClassPath():void
getClassPathRoots():String[]
handleClass(Class<?>):void
findAllMatchingTypes():List<Class<?>>
getScannedClassesCount():int
<<Java Class>>
FileWalker
com.eztech.util
matchFilter: FileFilter
baseDir: String
matchingFiles: int
allFiles: int
FileWalker()
FileWalker(FileFilter)
FileWalker(FileFindHandler)
FileWalker(FileFilter,FileFindHandler)
getMatchFilter():FileFilter
setMatchFilter(FileFilter):void
getHandler():FileFindHandler
setHandler(FileFindHandler):void
setBaseDir(String):void
walk():void
walk(File):void
getMatchingFileCount():int
getAllFilesCount():int
+JAVA_CLASS_FILE_FILTER
0..1
+ALL_FILES_FILTER
0..1
-fileHandler
0..1
-handler
0..1
-fileWalker
0..1
<<Java Class>>
JavaClassFileFilter
com.eztech.util
JavaClassFileFilter()
<<Java Class>>
MatchAllFileFilter
com.eztech.util
MatchAllFileFilter()
accept(File):boolean
<<Java Class>>
ClassLoadingFileHandler
com.eztech.util
ClassLoadingFileHandler()
updateClassPathBase(String):void
handleFile(File):void
<<Java Interface>>
FileFindHandler
com.eztech.util
handleFile(File):void
onComplete():void
<<Java Class>>
JavaClassFileWalker
com.eztech.util
JavaClassFileWalker()
JavaClassFileWalker(FileFindHandler)
-converter
0..1
<<Java Class>>
FileToClassConverter
com.eztech.util
classPathRoot: String
FileToClassConverter(String)
setClassPathRoot(String):void
convertToClass(File):Class
getClassFromName(String):Class
removeClassPathBase(String):String

## Parser

First we make arraylist that contains all statement and iterate on it and check the input is matches with which statement to execute  the query , All statement implement from one interface and each statement has 2 function :one for check matches and another to execute .

## Save and Load as Json or xml file:

### Save and Load using DOM

By using Document Builder Factory, Document Builder and Element objects.

### Save and Load using DTD

By using print writer object, we didn't use any jar, we implement it ourselves.

## Driver

Get the connection to database after accepting URL.

## Connection

Create statement object that execute queries.

## Statement

Execute queries and get result set.

## Result set

Is an object that containing the selected data from database.

## Result set metadata

Get information about data in result set.

## Database control

Control database base using driver.

**Printer**

to show the data in table.

**Table**

Control data in side tables like update ,insert ,select and alter.

**Database**

Control the  operations on the tables like creating and deleting table.

**MVC**

We have 2 package one of them to perform "control " the action (database control  ) and another package (model) to show the data in table.

**Design pattern**

Some of the design pattern used in this app:

- Delegation design pattern: We use object from data base control inside parser class to use function of SQL command, and object from printer inside database control class.
- Interface Design pattern: We provide interface for parser, Database control, printer , save and load and Statement.
- Factory Design Pattern: we have reader and writer factory classes to use them to have save and load concrete object depending on a parameter  which is string has value either "json" or "xml" .
- Builder Design pattern: we make IStatment interface and all statement implement from it, in parser class we loop on all statement to check which one of them that matches with then make object statement equal new object from statement that matches with.
- Chain of responsibility design pattern:  we make database control that control the database, database control on tables, table control on data inside the columns
- Singleton design pattern: we use this design pattern to get an instance of the printer class(model).

# FEATURES

**MVC Architecture**

The implementation is based on the famous MVC pattern.

**SQL command**

It is provided to use a lot  of order to manage your data like : insert , update , delete ,create and drop , also you can switch between database you want to use .

**User-friendly**

User can enter command insensitive word , also semicolon doesn't require , its provide to show data from table sorted by using "order by" , if he use incorrect command we show massage that detect the error he made , and we show all data if he change any things of data.

**Data Command**

1-Select: it's using to show data in table ,it should be in form "

```
SELECT column_name,column_name
FROM table_name;
```

"

2-Where :to detect special cells to perfrom action , it should be in form "

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

"

3-order by :to show data sorted by key , it should be in form "

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC|DESC, column_name ASC|DESC;
```

"

4-insert into : to make new data in database , it should be in form "

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);"
```

Or "

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

"

5-update : to change some data in our database , it should be in form "

```
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

"

6-Delete : to delete some data in our database , it should be in form "

```
DELETE FROM table_name
WHERE some_column=some_value;
```

"

**Note:** Be very careful when deleting records. You cannot undo this statement!

## Database command

-Create database: to create new database , it should be in form "

```
CREATE DATABASE dbname;
```

"

-Create table :to create new table in our database , it should be in form "

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);"
```

-Drop table : to delete table from database , it should be in form "

```
  Drop table table_name
```

"

-Drop database : to delete database , it should be in form "

```
  Drop database database_name
```

"

-Alter table : to add column in form

```
ALTER TABLE table_name
ADD column_name datatype
```

Or delete column in form

```
ALTER TABLE table_name
DROP COLUMN column_name
```

-Distinct: to remove duplicate values in form :

```
SELECT DISTINCT column_name,column_name
FROM table_name;
```