Pivot Table

COMP5541
Group C

Aiken (Chen-Fang) Chung
Christopher Birkbeck
Daniel Szwimmer
Eden Almakias
Ernesto Garsiya Melikhov
Rodrigo Pessoa

Introduction

This presentation outlines the scope and features of Release 2 of the Pivot Table Generator software and provides a high-level picture of the future course of its development. All diagrams and information included in this presentation reflect the refactored code and features implemented up until April 17, 2017.

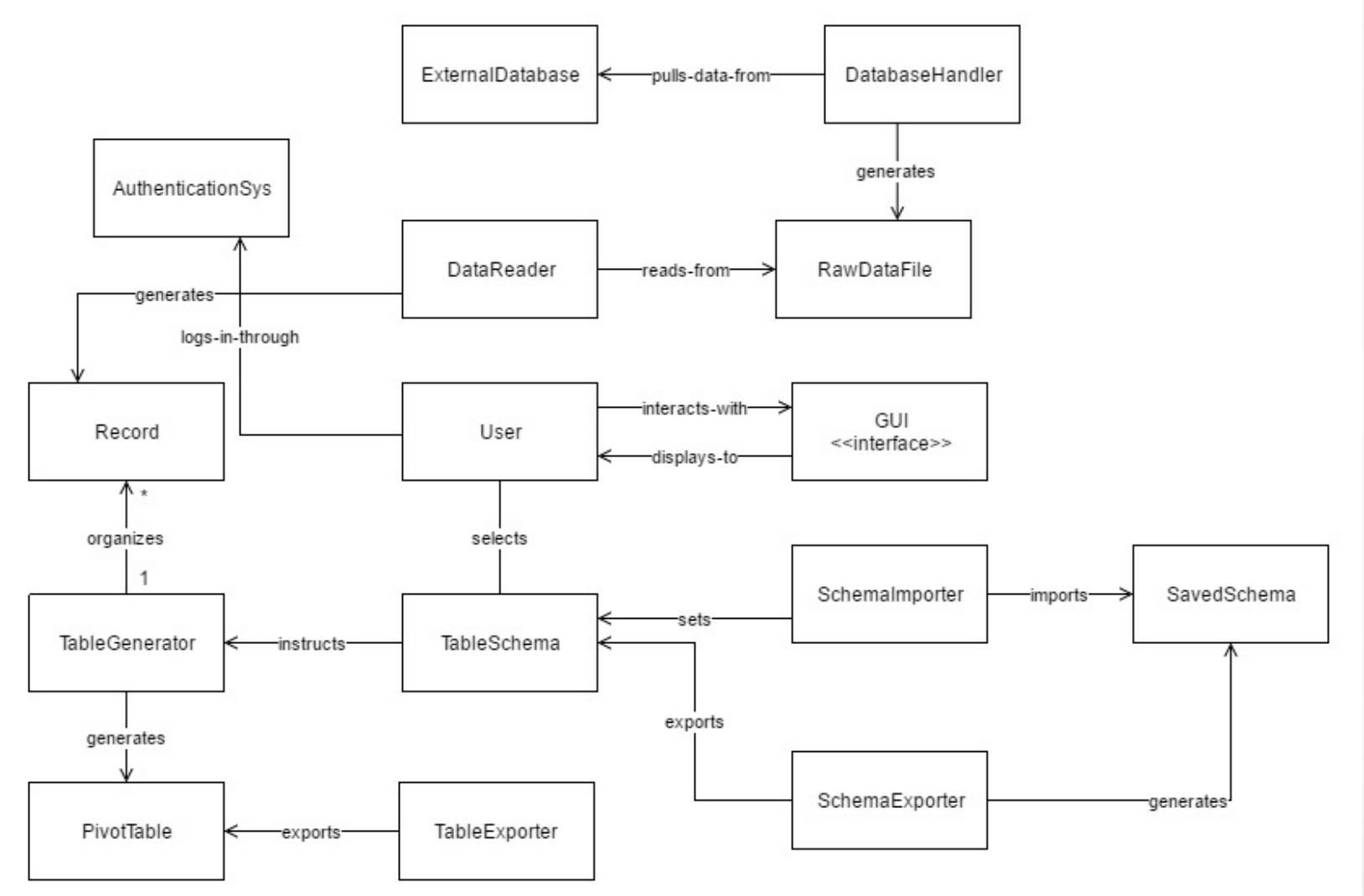
Index

- Domain Model 4
- Use Case Diagram 6
- Class Diagrams 8
  - Package Diagram 9
  - DatabaseHandler 10
  - TableExporter 11
  - TableMaker 12
- Architecture Patterns 13
- Design Patterns and Refactoring 14
- Future Implementation 16
- Timesheet (Logs) 17
- Roles 18

Domain Model

The following Domain Model was updated to reflect the new features and changes made to the first version of the Pivot Table Generator tool.

Domain Model

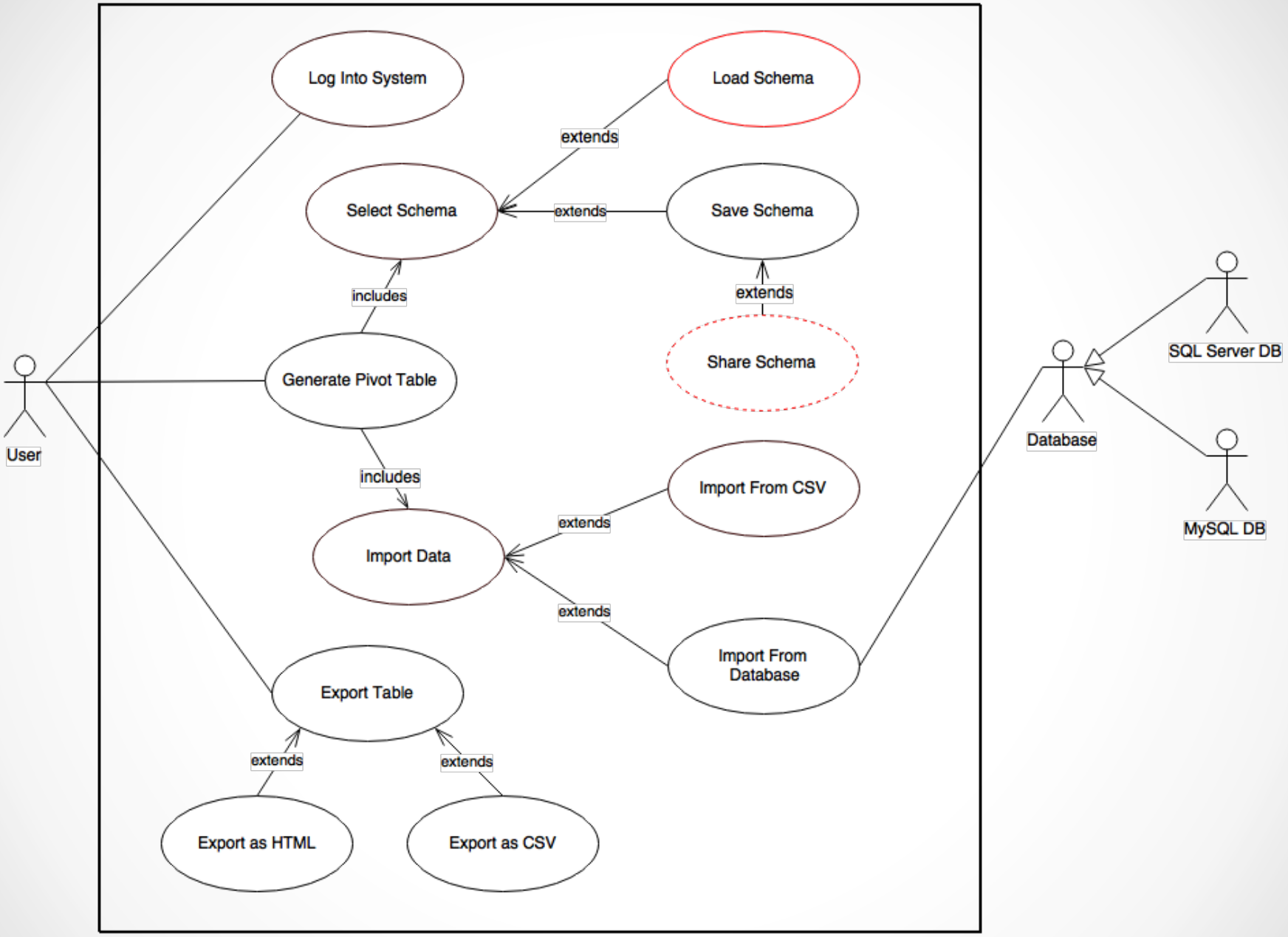


Use Case Diagram

- The following Use Case Diagram presents the major use cases of Pivot Table Generator.

- The "Load Schema" use case (loading an earlier saved schema and automatically recreating the original pivot table with its help) has been implemented, but is limited to local CSV files only and is, therefore, circled with a red line.

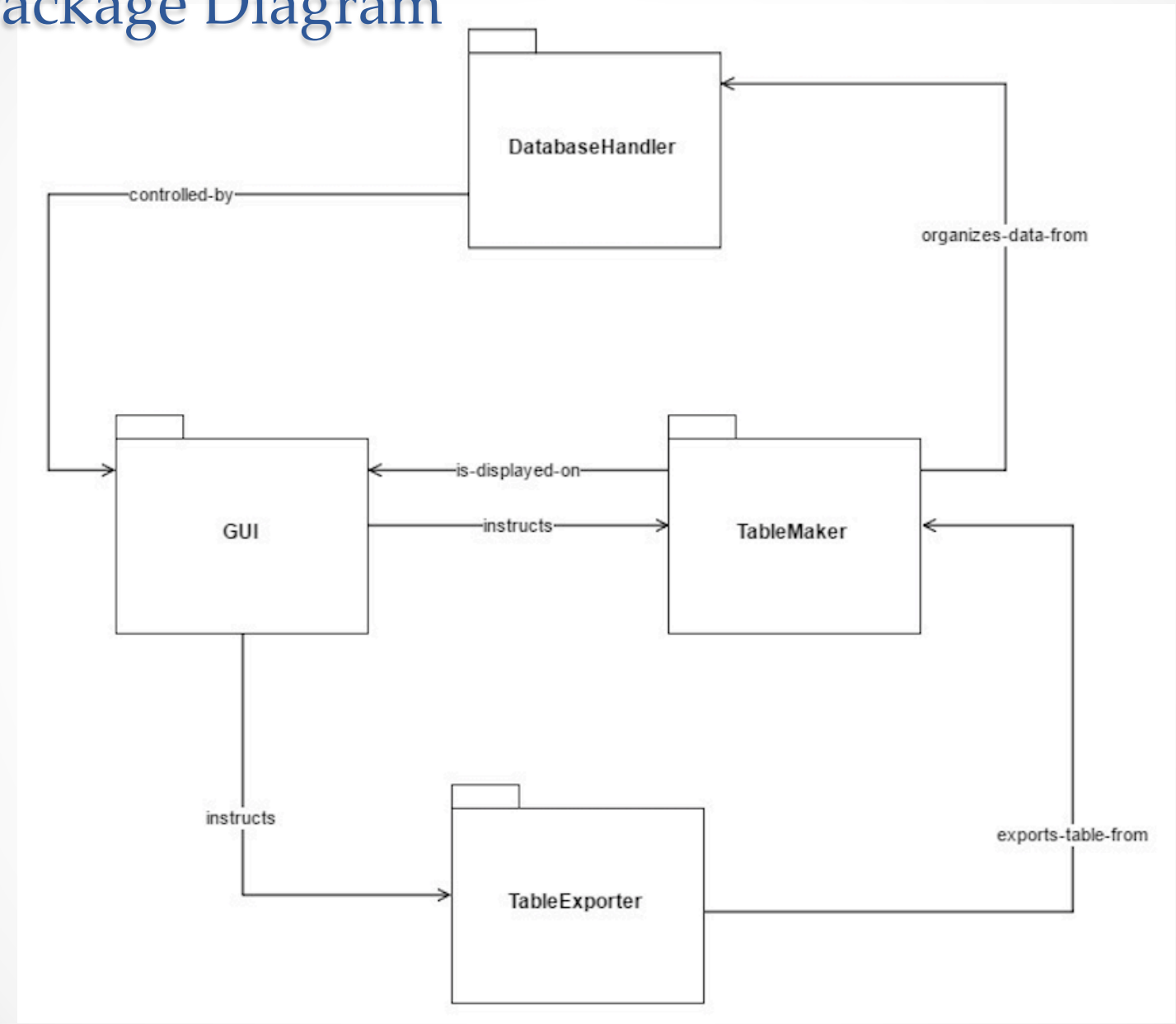
- The "Share Schema" use case is not implemented in this final version. Thus, it is circled with a red broken line. The future implementation of the user-sharing functionality is outlined in Annex I.


Log Into System
Load Schema
extends
Select Schema
extends
Save Schema
includes
extends
Share Schema
Generate Pivot Table
User
includes
Import From CSV
extends
Import Data
extends
Import From Database
Export Table
extends
extends
Export as HTML
Export as CSV
Database
SQL Server DB
MySQL DB

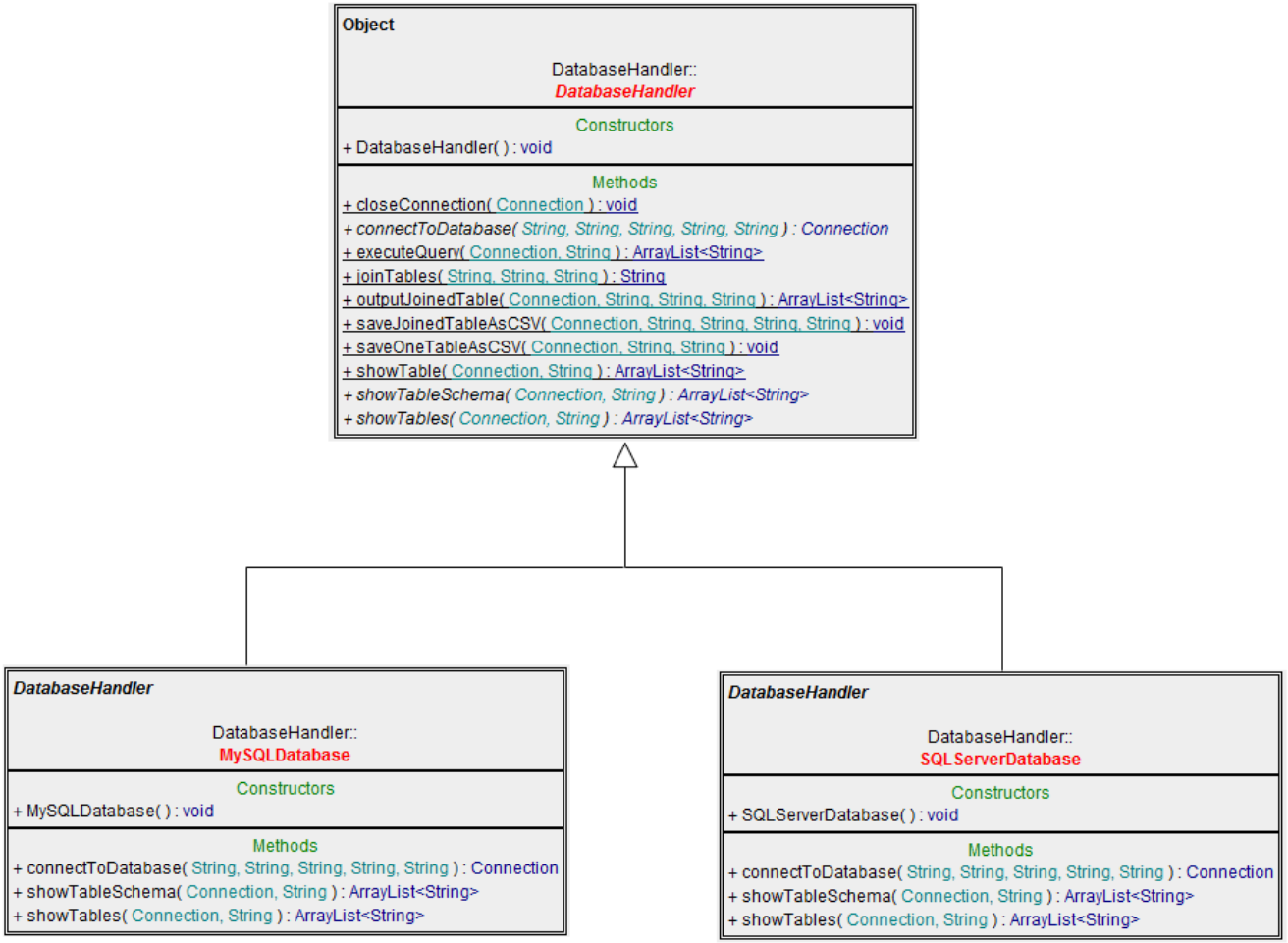
Class Diagrams

- The following slides will present three main class diagrams of the Pivot Table Generator tool:
  - DatabaseHandler
  - TableExporter
  - TableMaker
- In order to facilitate the understanding of the class diagrams, we included a "Package Diagram" that shows how the above modules and the GUI interact.
- All class diagrams can be found in the folder "Class_Diagrams" included in the Project's zip file.

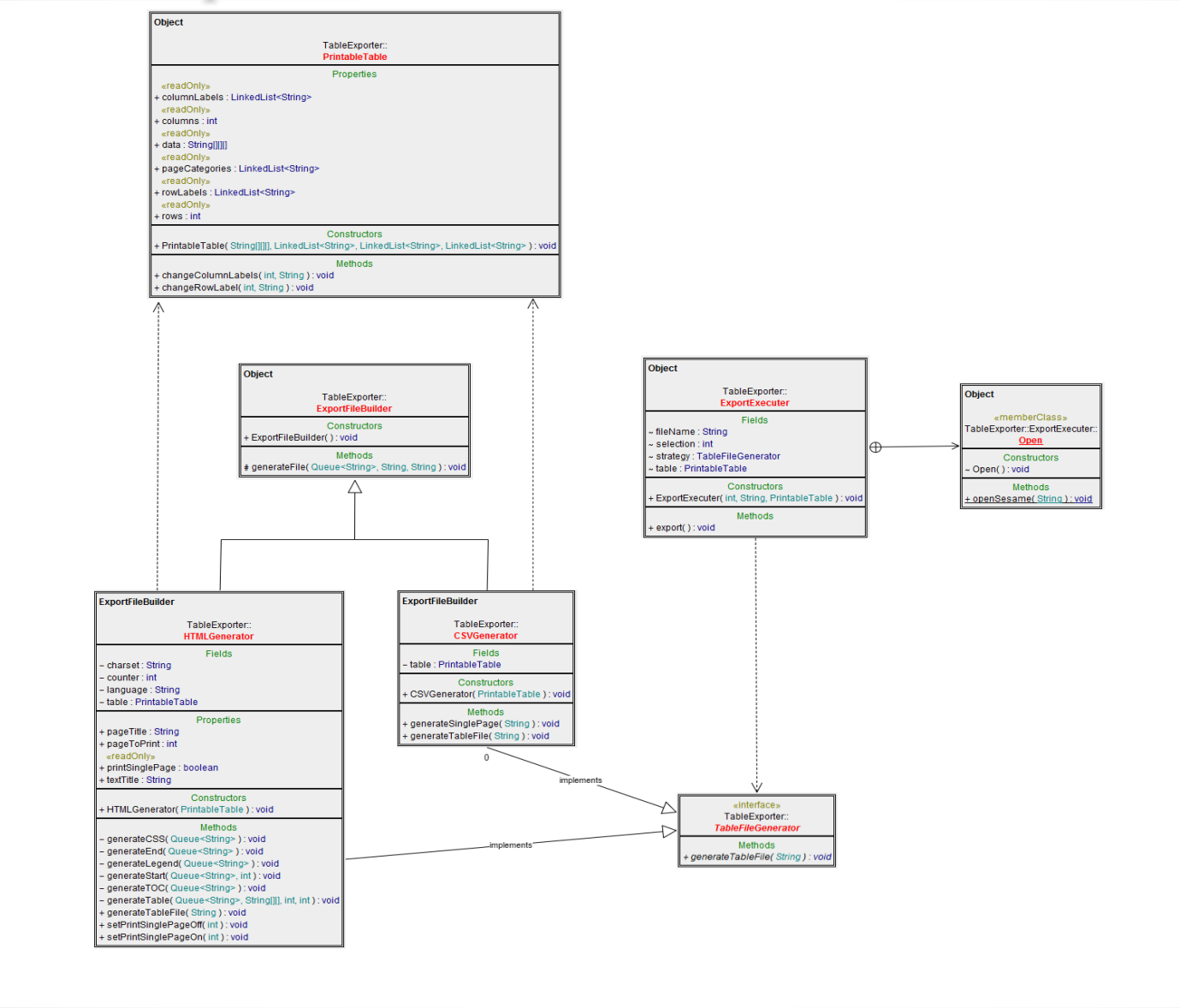
Package Diagram



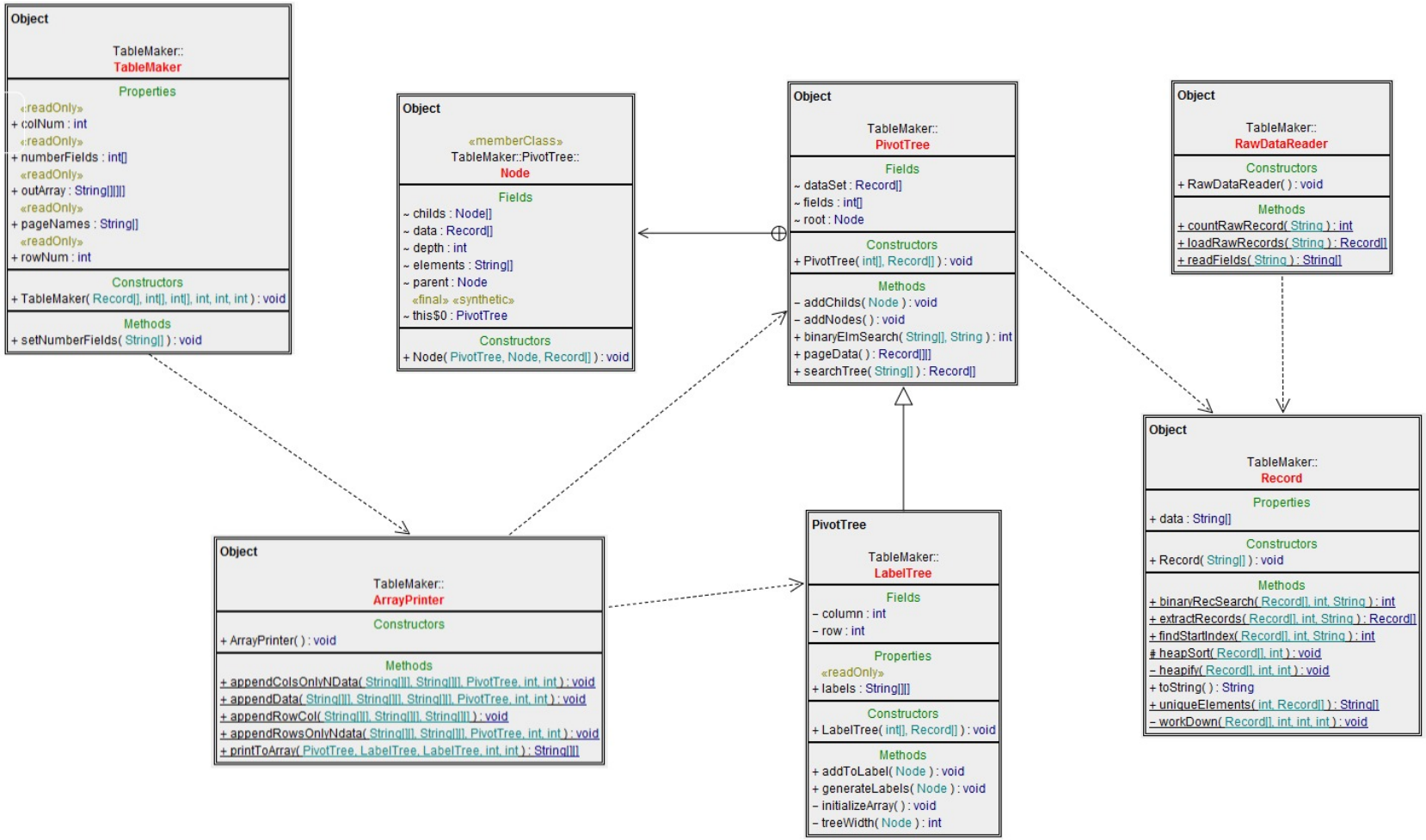
DatabaseHandler



TableExporter



TableMaker


Object
TableMaker::
TableMaker
Properties
«readOnly»
+ colNum : int
«readOnly»
+ numberFields : int[]
«readOnly»
+ outArray : String[][][]
«readOnly»
+ pageNames : String[]
«readOnly»
+ rowNum : int
Constructors
+ TableMaker(Record[], int[], int[], int, int, int) : void
Methods
+ setNumberFields(String[]) : void
Object
«memberClass»
TableMaker::PivotTree::
Node
Fields
~ childs : Node[]
~ data : Record[]
~ depth : int
~ elements : String[]
~ parent : Node
«final» «synthetic»
~ this$0 : PivotTree
Constructors
+ Node(PivotTree, Node, Record[]) : void
Object
TableMaker::
PivotTree
Fields
~ dataSet : Record[]
~ fields : int[]
~ root : Node
Constructors
+ PivotTree(int[], Record[]) : void
Methods
- addChilds(Node) : void
- addNodes( ) : void
+ binaryElmSearch(String[], String) : int
+ pageData( ) : Record[][]
+ searchTree(String[]) : Record[]
Object
TableMaker::
RawDataReader
Constructors
+ RawDataReader( ) : void
Methods
+ countRawRecord(String) : int
+ loadRawRecords(String) : Record[]
+ readFields(String) : String[]
Object
TableMaker::
Record
Properties
+ data : String[]
Constructors
+ Record(String[]) : void
Methods
+ binaryRecSearch(Record[], int, String) : int
+ extractRecords(Record[], int, String) : Record[]
+ findStartIndex(Record[], int, String) : int
# heapSort(Record[], int) : void
- heapify(Record[], int, int) : void
+ toString( ) : String
+ uniqueElements(int, Record[]) : String[]
- workDown(Record[], int, int, int) : void
Object
TableMaker::
ArrayPrinter
Constructors
+ ArrayPrinter( ) : void
Methods
+ appendColsOnlyNData(String[][], String[][], PivotTree, int, int) : void
+ appendData(String[][], String[][], String[][], PivotTree, int, int) : void
+ appendRowCol(String[][], String[][], String[][]) : void
+ appendRowsOnlyNdata(String[][], String[][], PivotTree, int, int) : void
+ printToArray(PivotTree, LabelTree, LabelTree, int, int) : String[][]
PivotTree
TableMaker::
LabelTree
Fields
- column : int
- row : int
Properties
«readOnly»
+ labels : String[][]
Constructors
+ LabelTree(int[], Record[]) : void
Methods
+ addToLabel(Node) : void
+ generateLabels(Node) : void
- initializeArray( ) : void
- treeWidth(Node) : int

Architecture Patterns

We started our project without using a specific architectural pattern. However, the most natural one for our Pivot Table Generator would be the MVC.

Model = data organization/processing

Viewer = GUI display

Controller = GUI buttons/input

Design Patterns and Refactoring

Strategy	Class	Reasoning
Façade	TableMaker	Instead of manipulating multiple objects to make a pivot table we manipulate one class that contains all the other objects that will be used to create the pivot table.
Strategy	FileGenerator	Depending on user selection at run time, a different concrete class will be assigned to the interface instance in the executing classes.
Refactoring: **Replace Conditional With Polymorphism**	DatabaseHandler	The conditional statements for handling MySQL vs MS SQL Server were replaced with two subclasses (MySQLDatabase and SQLServer, respectively), whereas the original class was abstracted and is now extended to the above concrete subclasses, instances of which are created based on GUI input.

Future Implementation

- Sharing System is foreseen for the implementation at the next stage in the development of the Pivot Table Generator tool.
- System must allow the user to share any created schemas with other registered software users.
- In order to develop, test and implement the new Sharing System, the team will require ca. 60 hours of work.
- Full description of the Sharing System is provided in Annex I.

Timesheet (Logs)

Name	Total Hours per Developer
Aiken Chung	86
Christopher Birkbeck	78.00
Daniel Szwimmer	71.50
Eden Almakias	75.50
Ernesto Garsiya Melikhov	83.50
Rodrigo Pessoa	76.50
Total:	471

Detailed logs are provided in Annex II

Roles

Name	1st Phase	2nd Phase	3rd Phase	4th Phase	5th Phase	6th Phase
Aiken Chung	Designer	Designer	Project Management	Coder	Coder/ Tester	Coder
Christopher Birkbeck	Designer	Tester	Coder	Coder	Coder	Project Management
Daniel Szwimmer	Coder	Coder	Designer	Tester	Project Management	Coder
Eden Almakias	Coder	Designer	Tester	Project Management	Coder	Coder
Ernesto Melikhov	Project Management	Coder	Coder	Tester	Coder	Coder/Tester
Rodrigo Pessoa	Tester	Project Management	Designer	Coder	Coder/ Tester	Coder/Tester