Process MeNtOR 3.o

Uni-SEP

<Project Name>

Design Document

Version:	
Print Date:	
Release Date:	
Release State:	
Approval State:	
Approved by:	
Prepared by:	
Reviewed by:	
Path Name:	
File Name:	
Document No:	

object oriented pty. ltd.

Document Change Control

Version	Date	Authors	Summary of Changes
0.0	Mar 10th	Caiya Zhang	Create the document
0.1	Mar 17th	Caiya Zhang	Component Diagram and Tables
0.1	Mar 17th	Vivian Liang	Detailed Class Diagrams
0.1	Mar 19th	Jiawei Li	Deployment Diagram
0.2	Mar 21st	Caiya Zhang	Activities Plans
0.2	Mar 22nd	Vivian Liang	Test Driven Development
0.2	Mar 23rd	Liu Fan	Detailed Class Diagrams
0.3	Mar 24th	Liu and Vivian	Use of Design Pattern
0.4	Mar 24th	Caiya Zhang	Introduction
0.4	Mar 24th	Jiawei Li	Major Design Decision
1.0	Mar 24th	Li, Liang, Fan, and Zhang	Final Check

Document Sign-Off

Name (Position)	Signature	Date
Member	LF	Jan 18th
Member	VL	Jan 18th
Member	CYZ	Jan 18th
Member	JWL	Jan 18th

Contents

1 Introduction

1.1 Purpose

This document details the requirements of the system DAGraph.

For small projects this document may contain the complete design model of a system. In larger projects this document will cover only a particular subject area.

1.2 Overview

The programming tasks for this project are summarized as follows:

1. To modify the Location server to be able to query the Service Registry, and pass the list of services or NULL to the Web Server for further processing.
2. To modify the database to be able to accept service descriptions from the registry and to verify strings.
3. To modify the Client Agent to reference a Wrapper.
4. To create a Wrapper class that references the Client Agent. This Wrapper class should use the api connecting the world bank to invoke the Web Service that has been selected using the Client Agent UI.
5. To modify the conditional loop to invoke a new service using the Client Agent UI.

The SDD document contains the following information:

1. Component Diagram of the system (Architecture). The level of detail and granularity will be at the Java Package level of details;
2. Deployment Diagram of the system assuming current run-time configuration.
3. Detailed Class Diagram for all classes created or modified in the system with only one level of associations. The detailed class diagram contains the classes in UML notation and a table for each class with its data members and methods with the appropriate signatures. Design patterns are identified.
4. State Diagram for the ***Call*** class.

1.3 Resources - References

References for different technologies used in the project are listed below:

Eclipse: http://www.eclipse.org/downloads/index.php

JfreeChart: https://www.jfree.org/jfreechart/

WSDP: http://java.sun.com/webservices/jwsdp/index.jsp

JAXR: http://java.sun.com/webservices/jaxr/index.jsp

JAXRPC: http://java.sun.com/webservices/jaxrpc/index.jsp

Maven: http://maven.apache.org/

2 Major Design Decisions

Most of the software code has been provided. Only WSEvent needs to be implemented to complete the operation of the Web Service Extension to GUI

However there are assumptions that were made regarding the operation of the software:

1. The paser in the WSEvent takes a float of the form <Service-description:Float>--<Service-Wsdlurl::URL> <Service-Wsolap::OLAP> <Service-Wsgraphml::GRAPHML> <Service-Endpoint> and it is assumed that no where in the service description, wsdl URL, or OLAP, or GraphML, or endpoint fields that there will be a control character of the form "--" that the parser needs to parse the data correctly to create a WSListData object. Also, wsdl, OLAP, GraphML and endpoints are assumed not to contain any spaces.
2. The user must be validated by the registry server for the Web service invocation to function.

3 Architecture

The software architecture pattern employed in this system is the **Client Server architecture**. The Client and Server (Data Visualization Calculator and Web Server) interact via calculation subsystems through user operations and a conditional visualization loop.

Below is the package level component diagram of the system [Figure 4.1]. Please note that due to space constraints only a highlight of interface methods is shown. The rest of the methods can be found in Tables 1 and 2.

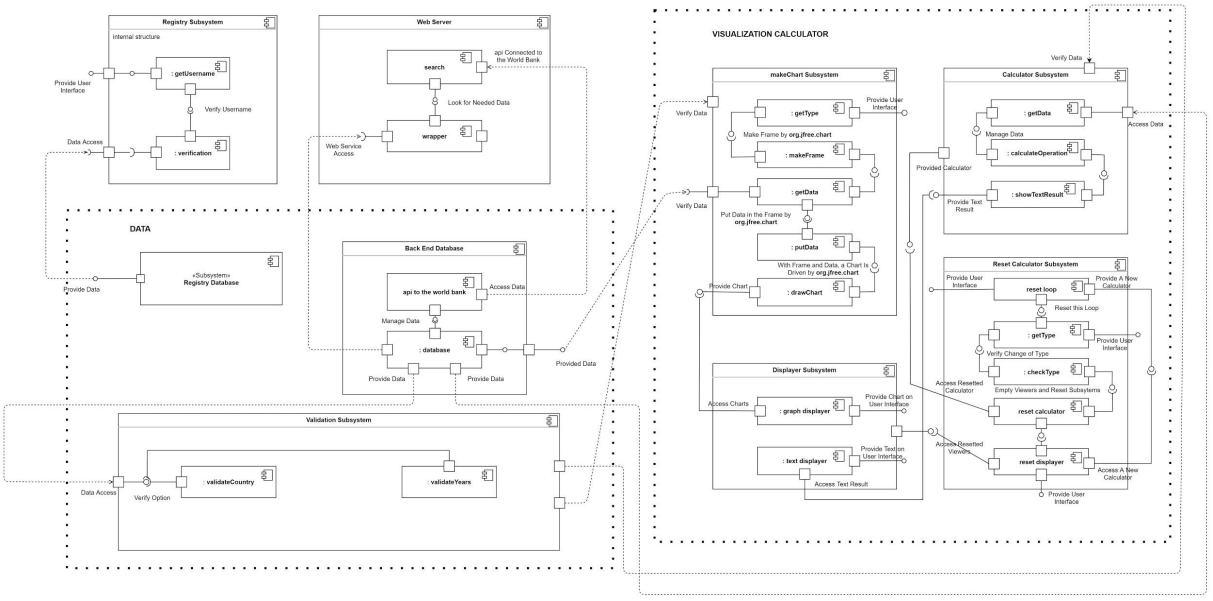


Figure 4.1

Table 1: Interface Methods for Calculation Server and Web Server

Calculation Server	INVOKE LOCAL SERVICE	Call invoke
Web Server	INVOKE WEB SERVICE	Call invoke
Data Access (api)	SEND REQUEST	api.sendRequest
	REQUEST CREATION	api.searchData
		api.getData
	GET RESPONSE	api.storeData

Please note that these methods are still a subset of the total methods that allows the interaction between calculator and the web server. These methods are methods that are of some significance and are on the main path of code assuming the Web Services query will be successful and the caller issues an invite to the callee who have registered Web Services.

Table 2: Interface Methods for Service Registry and Calculation Server

registry	User Log In	readNameFile
		getUsername
		getPassword
		findUsername
		checkPassword
calculate	User Make Choice	Type.getType
		Type.getCountry
		Type.checkCountry
		Type.countryException
		Type.getYears

		Type.checkYears
		Type.yearException
	User Add/delete Viewers	addViewer
		deleteViewer
		Type.getType
		Viewer.checkGraph
		emptyViewers
	System Calculate	getData
		calculateOperations
		showText
	System Draw Charts	Graph.getType
		Graph.makeFrame
		Graph.getData
		Graph.putData
		Graph.drawChart
	System Display	Graph.showChart
		Text.showText
	System Reset (Loop)	Type.getType
		Type.checkType
		Viewer.emptyViewers
		Viewer.addViewer
		Loop.calculate
		Calculate.Graph.getType
		Calculate.Graph.getData
		Graph.drawChart
		Graph.showChart
		Text.showText

Please note that these methods are still a subset of the total methods that allows the interaction between Registry Server and the Calculator assuming the Web Services query will be successful and the caller issues an invite to the callee who have registered Web Services.

The following figure [Figure 4.2] illustrates the deployment diagram of the system with the current run-time configurations involving the client and server, along with the application server all on one node.

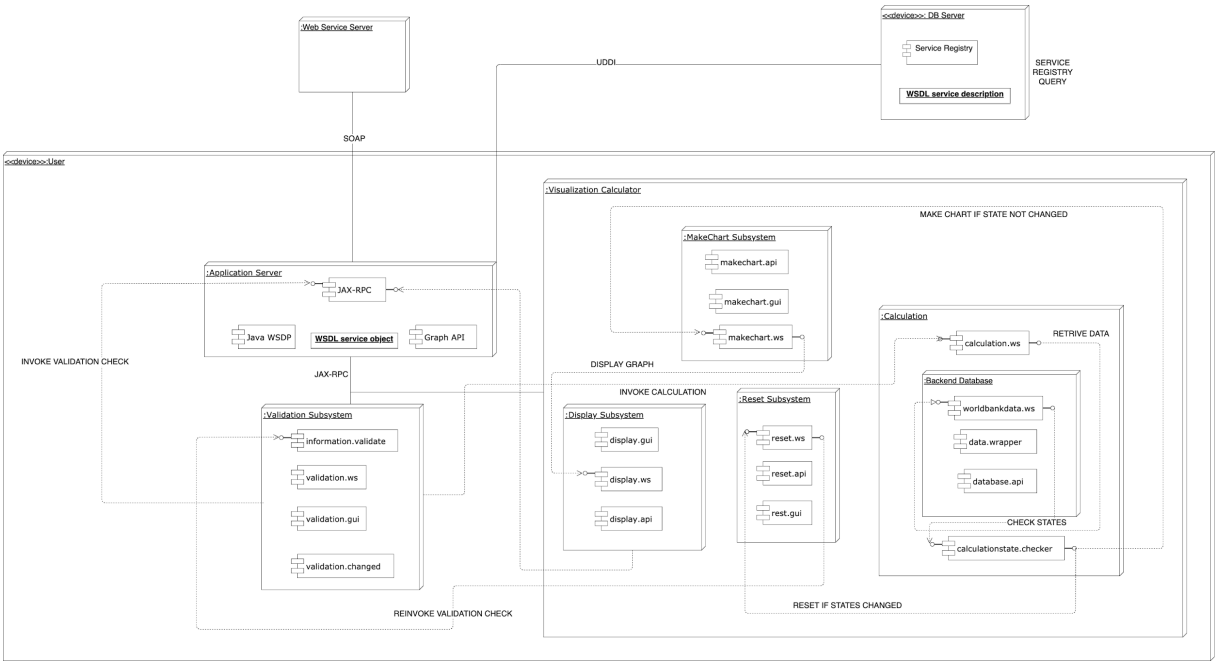
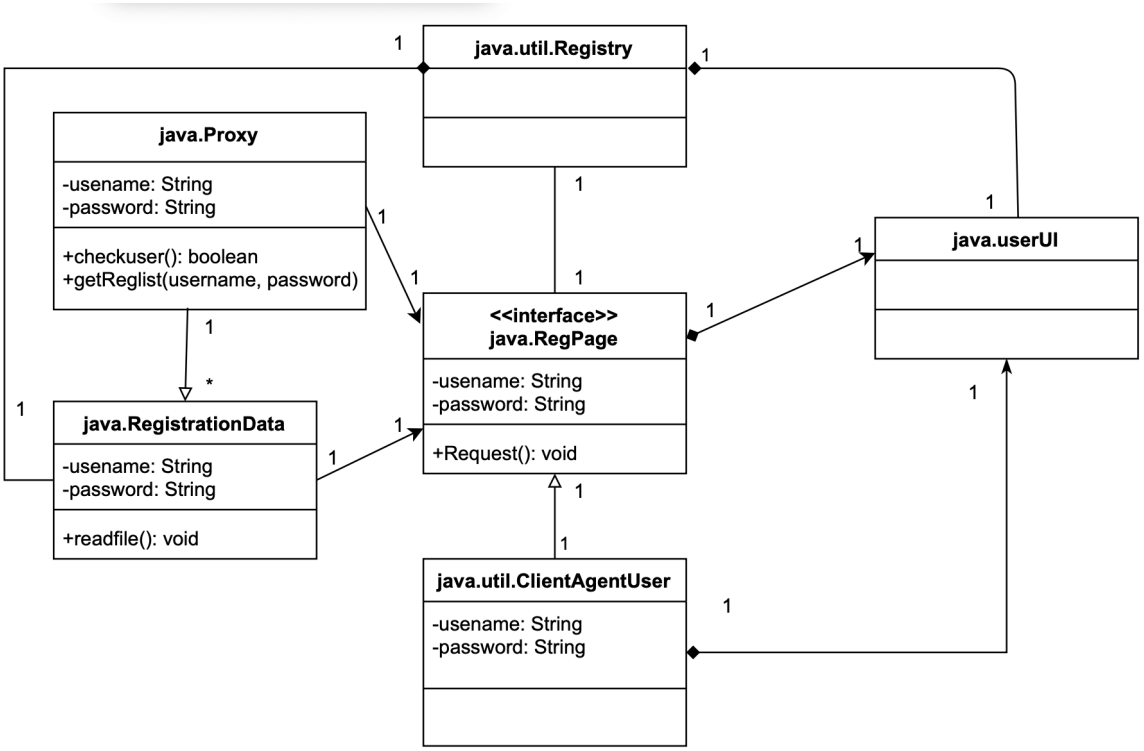


Figure 4.2

4 Detailed Class Diagrams

4.1 UML Class Diagrams

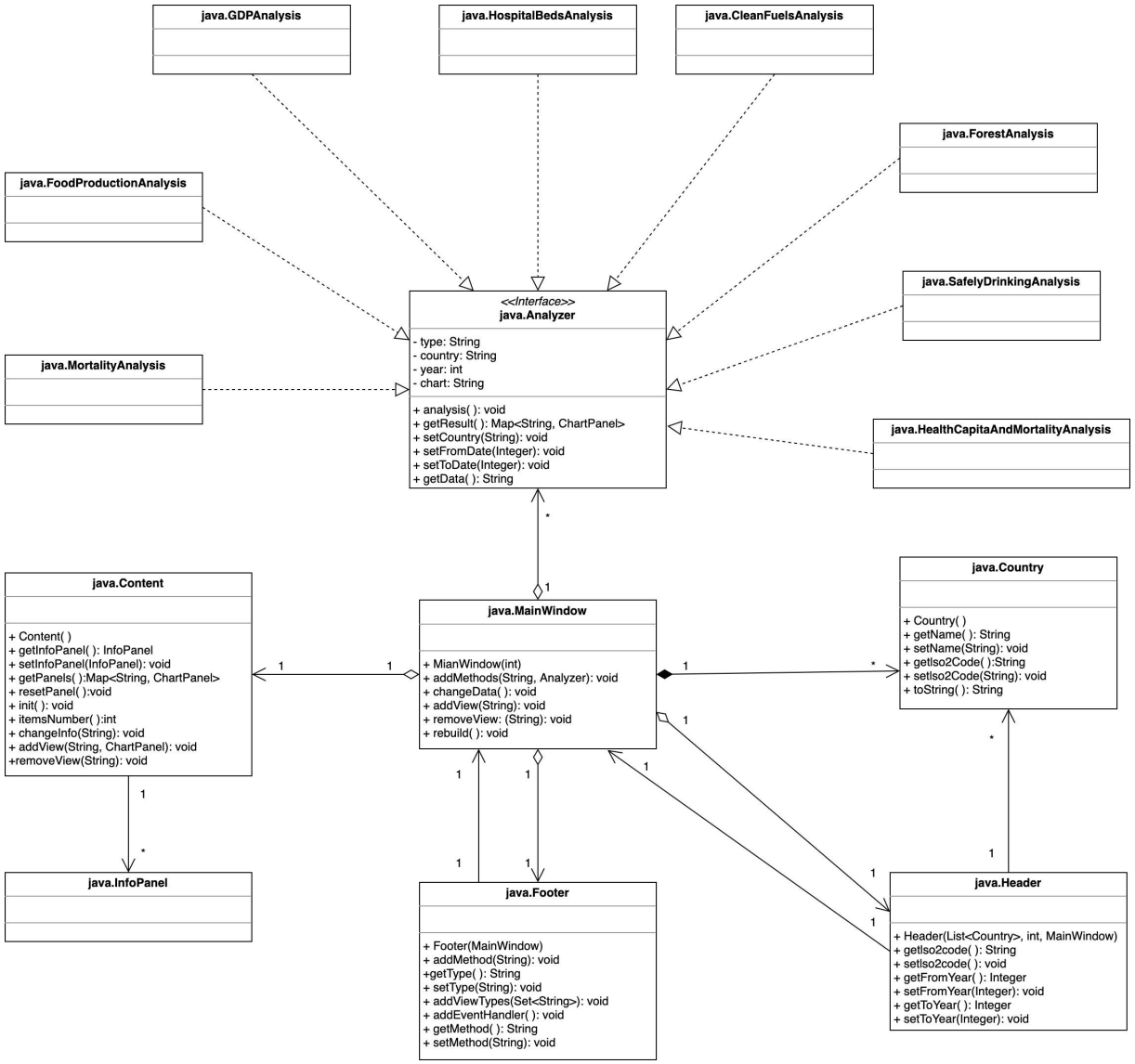
The following is the UML class diagram for the Client Agent User that pertains to the log-in process in the class Registry. The diagram shows that Registry would store the username-password pairs file and validate the users' input combination. For this case, we will use Proxy Design Pattern to handle the process.



Below is a figure showing the attributes and methods of the modified class ***Registry***

	java.util.Registry		
Attribute	-username: String		
	-password: String		
Method	+readfile():	java.Registry.namefile	Read the username-password pairs file
	+getRegList():	java.Reistry.RegistrationData	Get the pairs from the input file
	+checkuser():	java.Registry.proxy.check	Check the user input if it's correct

The following is the UML class diagram for the Analyzer and the associated Main Window that pertains to the modification in the class ***Analyzer.***

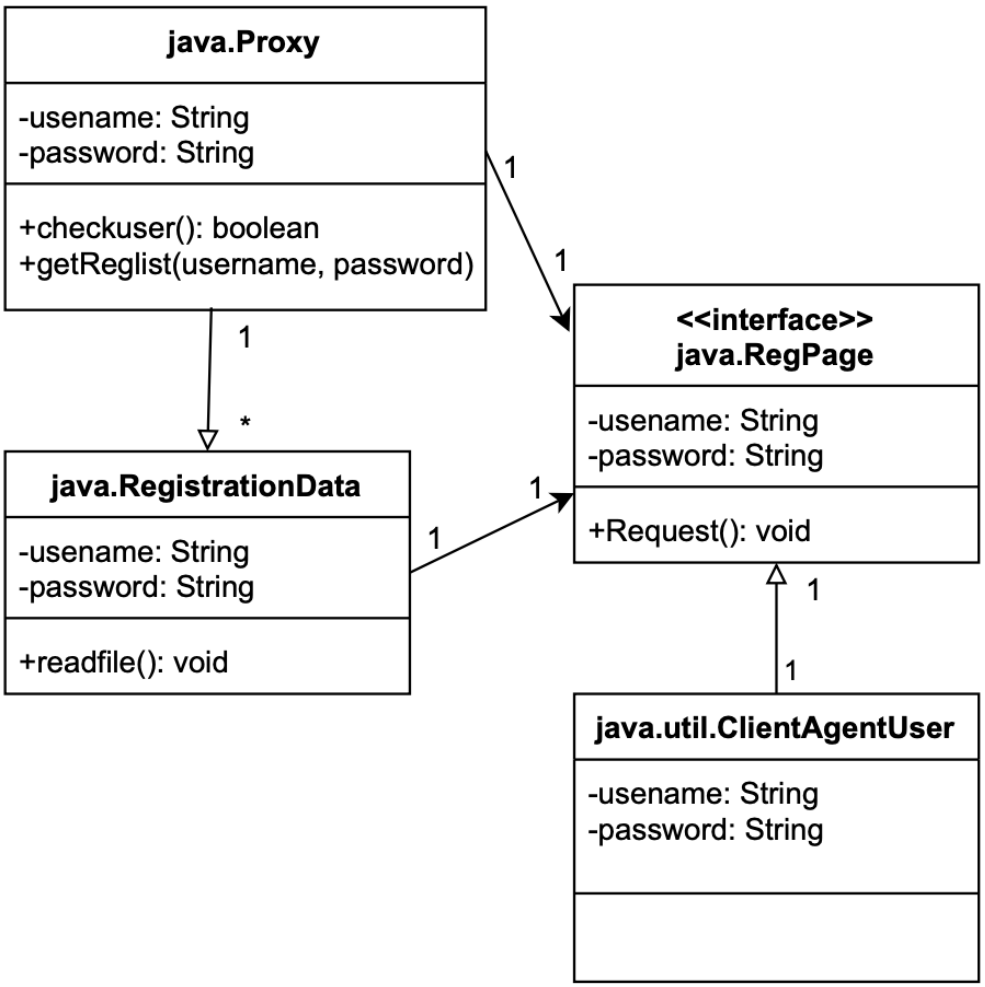


Below is a figure showing the attributes and methods of the modified class ***Analyzer***

	java.Analyzer		
Attribute	-type: String		
	-country: String		
	-year: int		
	-chart: String		
Method	+getResult(): Map<String, ChartPanel>	java.Analyzer	Get the result of analysis
	+analysis():	Void	Do analysis algorithm
	+jfreechart():	java.awt.Analyzer.Graph	Import jfree chart from java
	+getInfoPanel():	org.jfree.chart.Analyzer.Graph	Get the chart which the user chooses
	+exit () throws java.lang.Exception	void	
	+main (args: java.lang.String):	Void	
	+checkYear():	boolean	Check the year if it matches the country which the user chooses
	+checkCountry():	boolean	Check the country if it matches the type which the user chooses
	+checkChart():	boolean	Check the chart if it is able to display the data which the user chooses
	+addView(string, chartPanel):	Void	
	+removeView(string, chartPanel):	Void	
	+checkView():	Boolean	After user delete/add views, check if it exits the same one
	+setInfoPanel(string)	Void	Set the chart which the user chooses
	+rebuild():	Void	Clear all the data
	+resetPanel():	Void	If the user does not change the “type”, then only reset “year”,”country” and “chart”
	+setCountry(string):	Void	Set the country which the user chooses
	+getCountry()	java.Analyzer.Country	Get the country which the user chooses
	+setType(string):	Void	Set the type which the user chooses
	+getType()	java.Analyzer.Type	Get the type which the user chooses
	+setFromYear(int):	Void	Set the start year which the user chooses
	+getFromYear()	java.Analyzer.Year	Get the start year which the user chooses
	+setToYear(int):	Void	Set the end year which the user chooses
	+getToYear()	java.Analyzer.Year	Get the end year which the user chooses

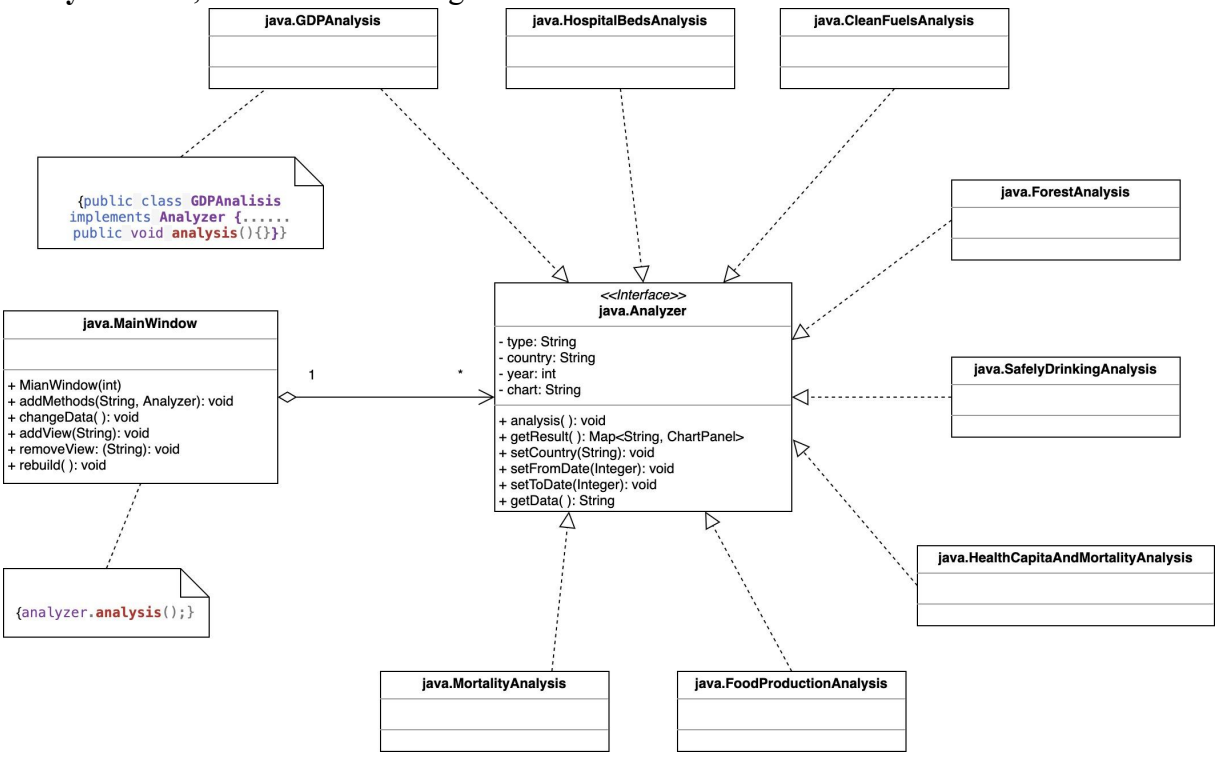
5 Use of Design Patterns

The following diagram shows a **Proxy Design Pattern** which encodes the username and password for checking the validation of the combination. In our case, only the correct combination can access the registry. We create a **Proxy** to control the access, when the user provides their username and password on the Registration Page, the **RegPage**(interface) will automatically request the **Proxy** to verify the combination. Then the **Proxy** will encode the users' input and invoke the username-password file to verify. Once the combination is verified successfully, the **Proxy** will ask the **RegPage**(interface) to allow the user to access the **Registry** and the **userUI**. Otherwise, the proxy will deny the user request.

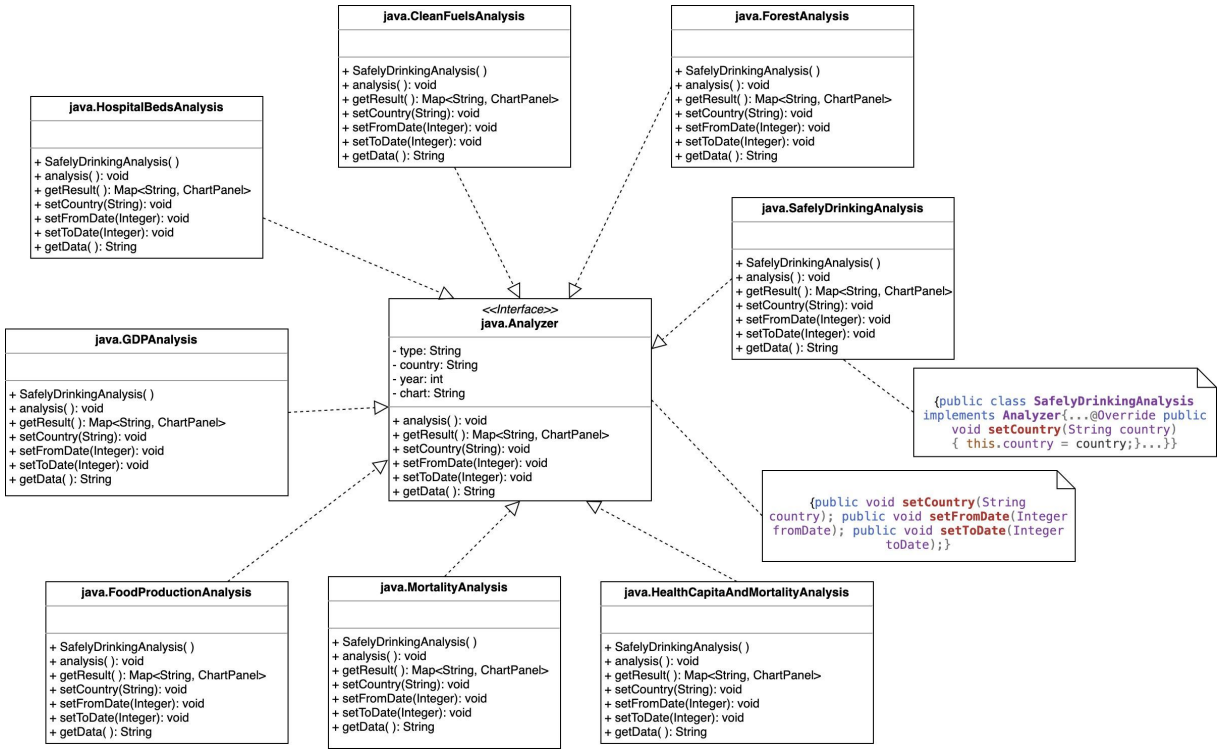


Several design patterns are evident in the **Analyzer** and the **MainWindow**.

The following diagram shows a **Strategy Software Design Pattern** that is evident in the classes of the **Analyzer**. A family of algorithms in the form of the specific type of **Analyzer** is encapsulated and can be used by the user interchangeably. Various implementations for the **Analyzer** exist, in our case 4 strategies are used.

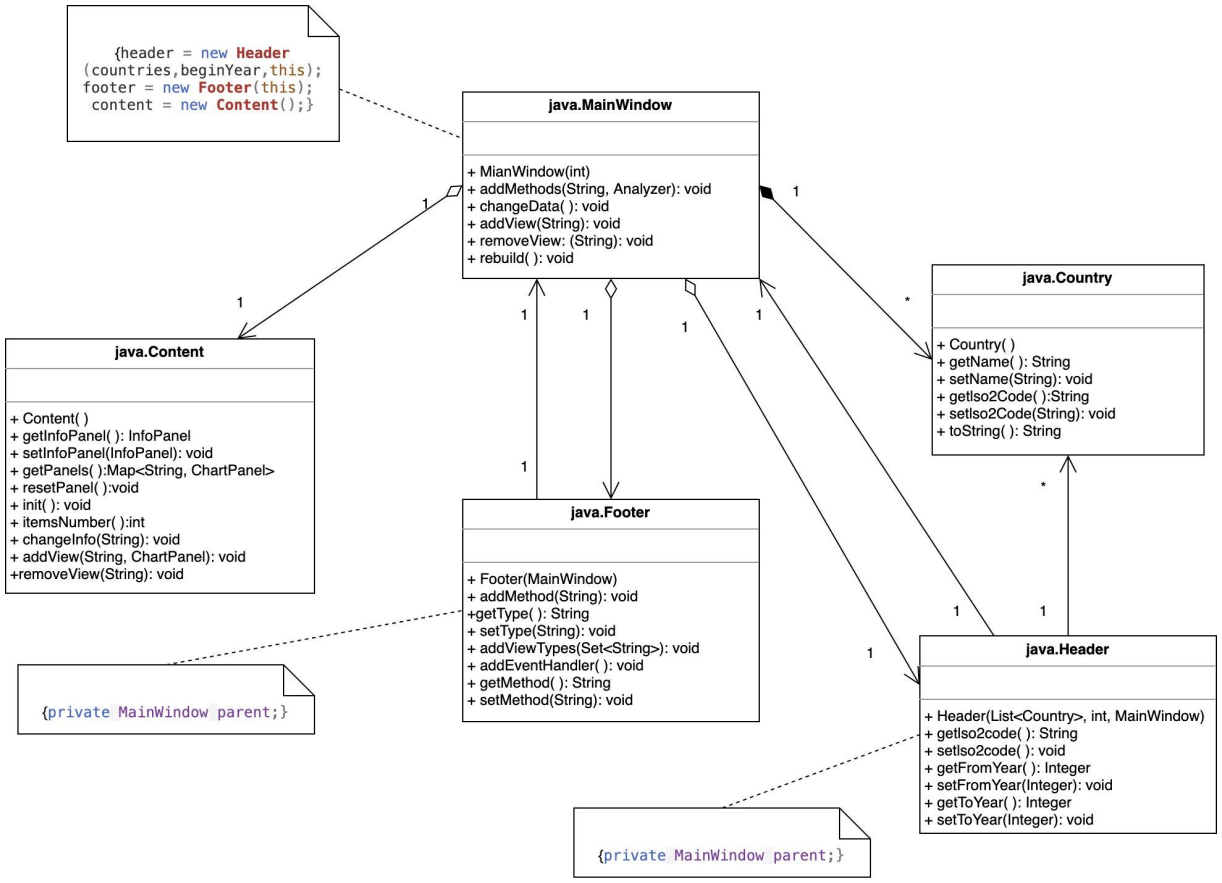


The following diagram shows an example of the **Builder Software Design Pattern** that is evident in the **Analyzer**. This design pattern separates the construction of a complex object from its representation so that the same construction process can create different representations. For example, in the case of the **SafelyDrinkingAnalysis** in the **Analyzer**, the interface declares steps of analysis that are common to the **SafelyDrinkingAnalysis** and also to all the other analysis.

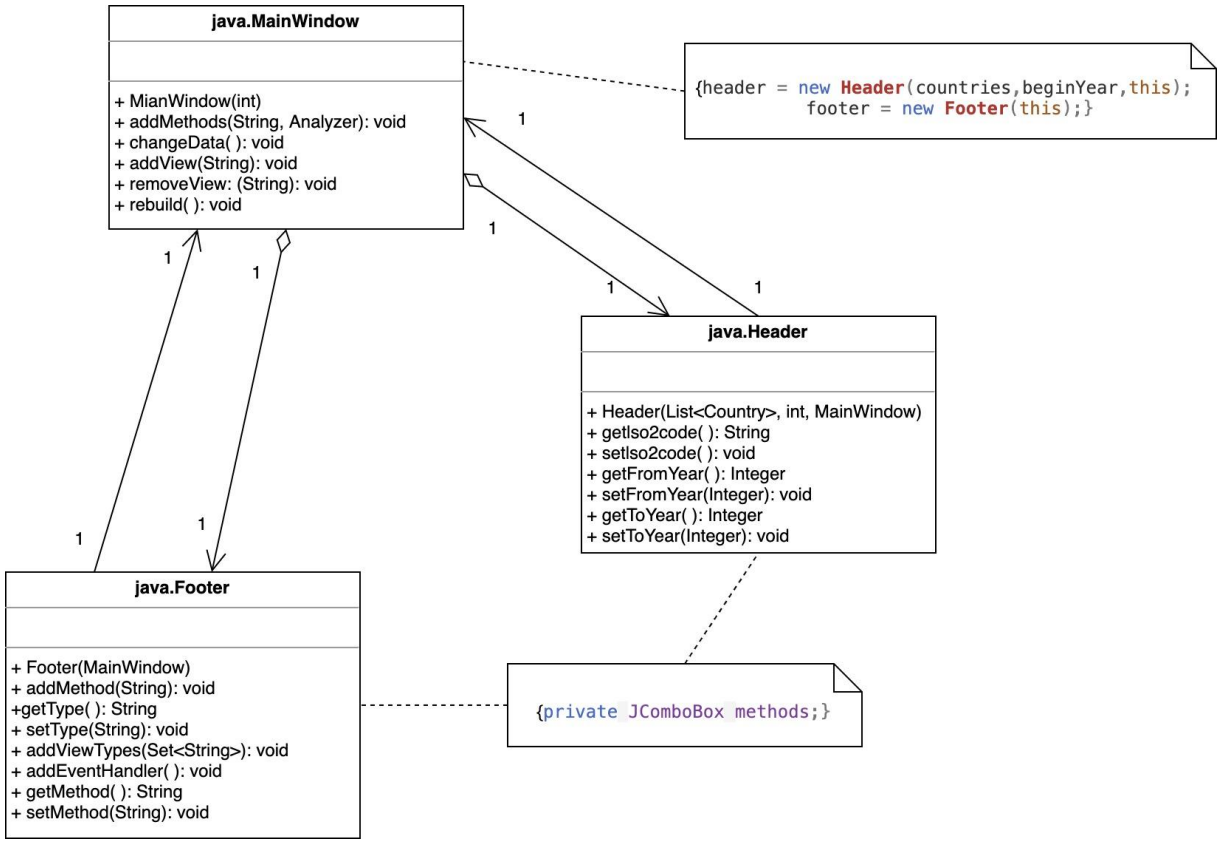


The following diagram shows a **Composite Design Pattern** that is evident in the classes of the **MainWindow**. This design pattern is used in the design of the main window, and the whole window is divided into three parts: **Header**, **Footer** and the **Content** area. The **Header** is responsible for displaying the information including country, time, and providing monitoring and response for the user's choice. The **Footer** is to display views and analyzers, and provide services when users make changes. The middle **Content** panel is to display various charts and analyze data information.

The following diagram also shows a **Mediator Design Pattern**. When the **Header** user makes a selection, the content area needs to be changed. There is no direct dependency between the **Content** area and the **Header** panel. The main window acts as a mediator to communicate. In the same way, when the foot changes, the main window act as a mediator to notify the **Content** panel to make changes.



The following diagram shows an **Observer Software Design Pattern** that is evident in the class **MainWindow**. This design pattern is reflected in the operation of the mouse or the drop-down box in the main window. When we click the drop-down box to change the option, the windows respond according to the operation we make. This is a typical listener mode.



6 Activities Plan

1.1 Project Backlog and Sprint Backlog

1. Introduction
    a. not urgent;
    b. need to be written before due date;
    c. content of purpose and overview part can be confirmed after first meeting;
    d. resources and references can be confirmed after part 3-5 are written
2. major design decisions
    a. significant, and not very urgent;
    b. need to be confirmed after first meeting;
    c. should be written before due date;
3. architecture
    a. significant and urgent
    b. should be discussed in the first meeting;
    c. Caiya needs to draw the component diagram and provide the two tables in two days;
    d. Klopp needs to draw the deployment diagram after component diagram is confirmed;
    e. diagrams should be modified after the second meeting;
    f. supplementary words should be written before due date;
4. detailed class diagrams
    a. significant and urgent;
    b. should be discussed in the first meeting;
    c. Vivian needs to draw detailed class diagrams after Caiya giving a component diagram;
    d. Liu should help Vivian complete detailed explanation and tables together;
    e. diagrams and tables should be modified after the second meeting;
    f. final check;
5. use of design patterns
    a. significant, and not very urgent;
    b. should be discussed in the second meeting;
    c. should be written before due date;
6. activities plan
    a. plans should be complete before the first meeting;
    b. Caiya need to keep on recording regularly;
7. test driven development
    a. should be discussed in the second meeting;
    b. should be completed before due date;

1.2 Group Meeting Logs

Present Group Members	Meeting Date	Issues Discussed / Resolved
Vivian Liang, Jiawei Li, Liu Fan, Caiya Zhang	Mar 5th	Topic: Generally Identify Tasks for SDD (160mins) 1. Schedule 5 main parts; 2. Caiya and Klopp were assigned Architecture part; 3. Vivian and Liu were assigned Detailed Class Diagrams and use of Design Patterns parst; 4. Talk about Component level structure and main methods should be used in real code.
Vivian Liang, Jiawei Li, Caiya Zhang	Mar 12th	Topic: Discussion about modifying Part3-4 (50mins) 1. Discuss details needed to modified in Part 3 and 4; 2. Discuss details in Introduction part; 3. Talk about the detailed classes and methods in real code;
Vivian Liang, Jiawei Li, Liu Fan, Caiya Zhang	Mar 19th	Topic: Final Check (50mins) 1. Confirm each diagram; 2. Complete test driven development part; 3. Check text details and format;

7. Test Driven Development

Test ID	Test1: Login
Category	*evaluation of user credentials stored on file or DB*
Requirements Coverage	*UC1-Successful-User-Login*
Initial Condition	*the system has been initiated and runs*
Procedure	*1. The user selects login* *2. The user provides a user name* *3. The user provides a password* *4. The user logs-in into the system and is presented with the main UI window*
Expected Outcome	*the login form closes, and the user is presented with the main UI window*
Notes	*the user should provide only alphanumeric usernames and passwords without any special characters*

Test ID	Test2: Analysis Selection
Category	evaluation of the system to retrieve the correct data from the users' selection of analysis
Requirements Coverage	*UC2-Successful-Run-Analysis*
Initial Condition	Initial analysis of the system
Procedure	*1. The user selects type* *2. System restrive data and display in modular way*
Expected Outcome	System performs correct analysis data in modular way
Notes	

Test ID	Test 3: Empty
Category	*evaluation of system empties the data correctly without exception*
Requirements Coverage	*UC2-Successful-Empties-Analysis*
Initial Condition	After performing analysis, system can initialize all the data
Procedure	The list of steps required for this test case (*e.g.* *1. The user selects type* *2. System retrieve data*

	2. The user re-select the type *3. System empties the type, analysis data*
Expected Outcome	The analysis data will be emptied and the user is performed with the main UI windows.
Notes	The user changes the analysis type, the system empties the analysis.

Test ID	Test 4: Country Test
Category	Evaluation of fetching proper country
Requirements Coverage	UC3-Successful-Country
Initial Condition	User select the type to analysis
Procedure	*1. The user selects country* *2. System fetching the corresponding data*
Expected Outcome	the corresponding country data is fetching
Notes	System only processes the proper country with the analysis chosen.

Test ID	Test 5: Country Failed Test
Category	Evaluation of fetching improper country
Requirements Coverage	UC3-Unsuccessful-Country
Initial Condition	User select the type to analysis
Procedure	*1. The user selects country which is not on the list* *2. System pops up a notification and ask the user for another selection*
Expected Outcome	a small window with notification about the wrong country data selected

Test ID	Test 6: Year Test
Category	Evaluation of fetching proper year
Requirements Coverage	UC4-Successful-Year
Initial Condition	User select the type and corresponding country to analysis
Procedure	*1. The user selects start year* *2. The user selects end year*

	3. System fetching the corresponding data
Expected Outcome	the corresponding year data is fetching
Notes	System only processes the proper year range with the analysis and country chosen.

Test ID	Test 7: Year Failed Test
Category	Evaluation of fetching inproper year range
Requirements Coverage	UC4-Unsuccessful-Year
Initial Condition	User select the type and corresponding country to analysis
Procedure	*1. The user selects start year* *2. The user selects end year* *3. System notices that the year of the data is not available* *4. System pops up a notification and ask the user for another selection*
Expected Outcome	a small window with notification about the wrong data selected

Test ID	Test 8: Add Viewer
Category	Evaluation of the new added viewer
Requirements Coverage	UC5-Successful-Add-Viewer
Initial Condition	The user selects the type with corresponding country and year, the system is ready to present the analysis
Procedure	*1. The user add a new viewer* *2. System checks for the compatible* *3. System added the new viewer*
Expected Outcome	The added viewer with proper data is present
Notes	System only performs the compatible viewers with the analysis chosen.

Test ID	Test 9: Failed Add Viewer
Category	Evaluation of the failure to add viewer
Requirements Coverage	UC5-Unsuccessful-Add-Viewer
Initial Condition	The user selects the type with corresponding country and year, the system is ready to present the analysis
Procedure	*1. The user add a new viewer*

	2. System checks for the compatible *3. The added viewer is not compatible to the analysis data, then system pop up a notification window*
Expected Outcome	A pop up window to notify user
Notes	

Test ID	Test 10: Delete Viewer
Category	Evaluation of the deleted viewer
Requirements Coverage	UC6-Successful-Delete-Viewer
Initial Condition	The user selects the type with corresponding country, year, and viewers, the system is ready to present the analysis
Procedure	*1. The user selects a viewer to delete* *2. System checks for the target viewer if exists* *3. System deleted the viewer*
Expected Outcome	Data is present with selected viewers after deletion
Notes	

Test ID	Test 11: Failed Delete Viewer
Category	Evaluation of the failure to delete viewer
Requirements Coverage	UC6-Unsuccessful-Delete-Viewer
Initial Condition	The user selects the type with corresponding country, year, and viewers, the system is ready to present the analysis
Procedure	*1. The user selects a viewer to delete* *2. System checks for the target viewer if exists* *3. System pops up a notification window*
Expected Outcome	A pop up window to notify user the selective viewer doesn't exit
Notes	

Test ID	Test 12: Recalculate
Category	Evaluation of recalculation of the data
Requirements Coverage	*UC7-Successful-Recalculate*
Initial Condition	System remains the previous analysis data and user recalculate the data with the same analysis type
Procedure	*1. The user selects country*

	2. The user selects start and end year *3. System initiate the algorithm* *4. System retrieves corresponding data*
Expected Outcome	Display the new selected data and viewers
Notes	The user cannot change the type if they recalculate

Test ID	Test 13: Display Result
Category	Evaluation of display
Requirements Coverage	*UC8-Successful-Display-Result*
Initial Condition	All the data, algorithm, views are ready to display.
Procedure	*1. System do algorithm* *2. System display the result with corresponding viewers*
Expected Outcome	The selected data with corresponding viewers are displayed
Notes	