PROJECT REPORT ON

Summer Internship 2021

Submitted By:

Kabir Dureja
kabir19365@iiitd.ac.in
+91-7011223518

Under The Guidance Of:

Mr. Vipin Bhatia
(Asst. General Manager - Systems)

July, 2021

C-EDGE
A TCS - SBI Enterprise
Innovate. Excel. Empower

C-Edge Technologies Ltd., Mumbai 400093, India

ACKNOWLEDGEMENT

I deem it a pleasure to acknowledge my sense of gratitude to my project guide and mentor Mr. Vipin Bhatia (Asst. General Manager - Systems) under whom I carried out the project work. His incisive and objective guidance and timely advice encouraged me with constant flow of energy to continue the work. Moreover, his constructive criticism has contributed immensely to the evolution of my ideas on the project.

Date: 29.07.2021

Place: Dr. Mukherjee Nagar, Delhi

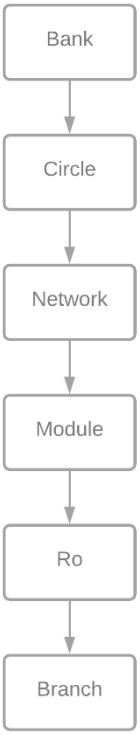
Kabir Dureja

Aspiring Software Engineer

CSSS Undergraduate | Batch of 2023, IIIT Delhi

Summary

The project was to design a software which is capable of generating reports and graphs in a matter of seconds to facilitate statistical analysis from millions of records in the hierarchy of a bank. The hierarchy of the bank is as follows:



The application supports two types of reports: summary report & detailed report; and two types of plots: bar chart & pie chart.

Requirements

Tools & Technologies:

The software is programmed in **Java (SDK 15 [version 15.0.2])** with the help of following tools: -

1. **JavaFX** (15): platform for creating rich applications.
2. **FXML**: for defining the UI of the JavaFX application.
3. **JDBC Driver** (ojdbc6.jar): to establish a connection between the Java application and Oracle Database.
4. **SQL**: to write and perform SQL queries in order to fetch the data from the database demanded by the user.
5. **Git**: for version control.
6. **Oracle Database** (Express Edition) : to create a dummy database for testing.

Modified build & run configurations by adding the following command to the VM options (for IntelliJ IDEA):

```
--module-path "path_to_javafx_jar_files" --add-modules javafx.controls,javafx.fxml
```

The idea was to provide separate interfaces for generating reports and generating graphs.

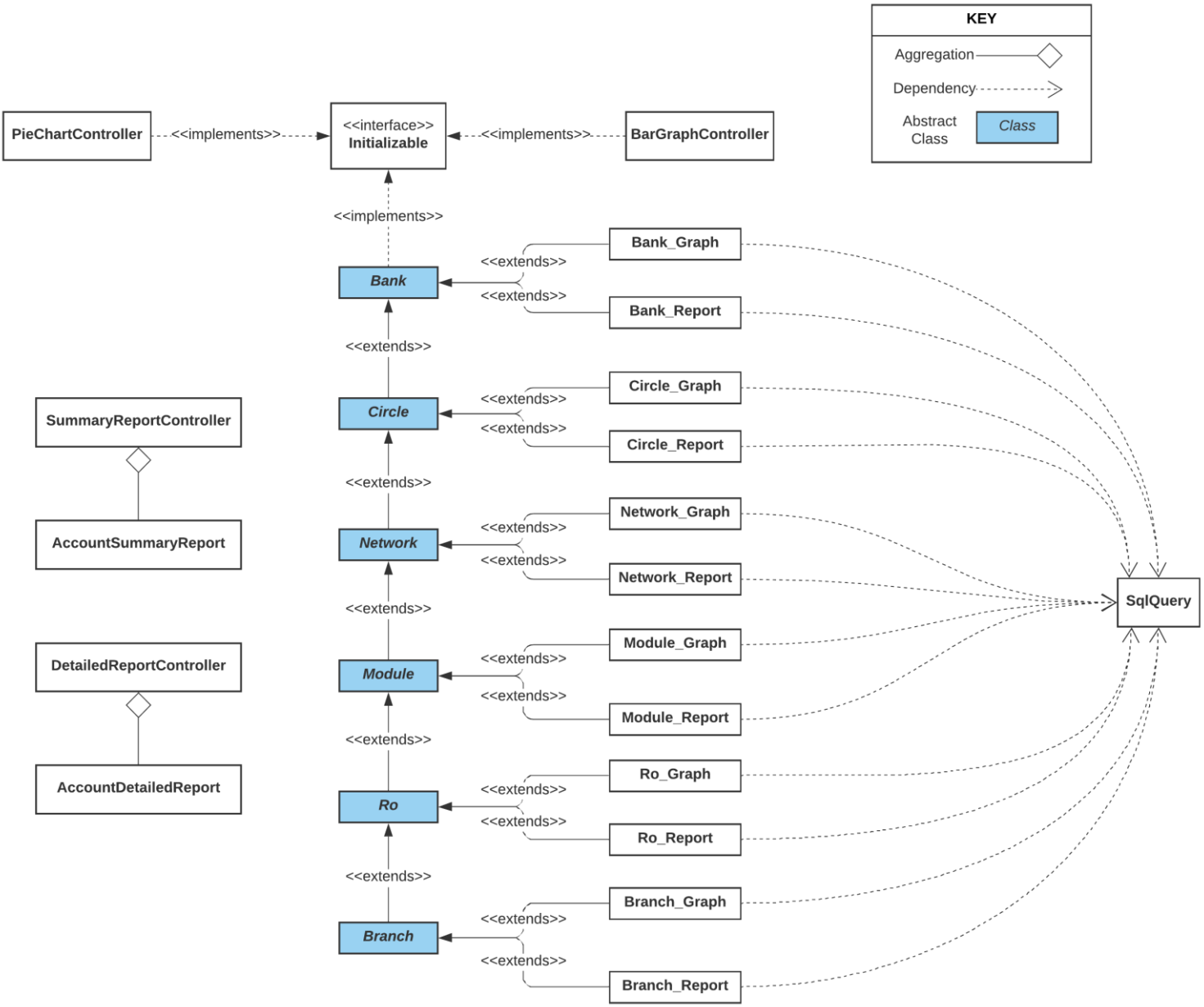
Report:

1. Implement two types of report generation for each level in the hierarchy for a given input date period which a user can ask for: summary report and detailed report.
2. Handle errors in queries, if any.
3. Provide filtering option (for each attribute) to search through generated reports.
4. Make the columns of the report mutable, i.e., user can select to add/remove column(s) as per the need.

Graph:

1. Implement two types of chart generation for each level in the hierarchy for a given input date period which a user can ask for: bar chart and pie chart.
2. Handle errors in queries, if any.
3. Provide parameter option to choose for the plot.
    a. Parameters for bar chart: date-wise graph, account type + date-wise graph.
    b. Parameters for pie chart: date-wise graph, account type wise graph.
4. Provide options for period selection for the graph. The options include: monthly, weekly and daily.

UML Class Diagram



About the Program

Following the hierarchy, nested abstract classes are used (for each level in hierarchy) so as to achieve partial abstraction in order to make efficient use of repetitive code blocks. Each level of abstract class has two concrete classes extending it, i.e., one specifically for the graph and the other for the report; both of which are using instance of the class *SqlQuery* to execute queries provided by the user.

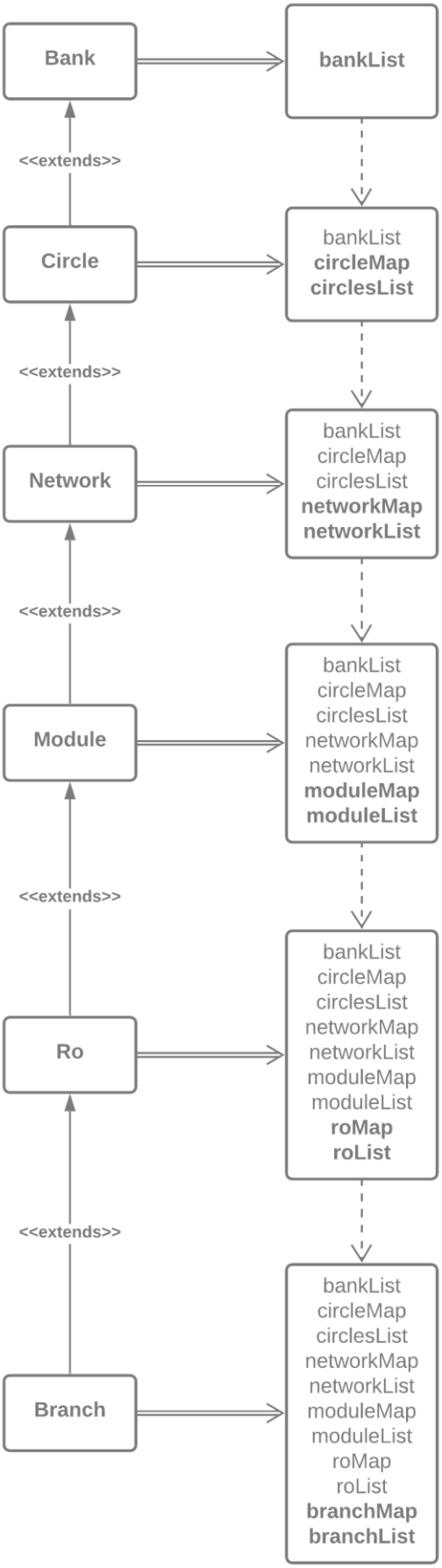


Fig (i)

bankList contains only one bank.

circlesList contains all the circles under the bank.
circleMap maps the ID of circles with their names.

networkList contains all the networks under the selected circle.
networkMap maps the ID of networks with their names.

moduleList contains all the modules under the selected network.
moduleMap maps the ID of modules with their names.

roList contains all the regional offices under the selected module.
roMap maps the ID of regional offices with their names.

branchList contains all the branches under the selected regional office.
branchMap maps the ID of branches with their names.

Abstract Classes:

Each class has a ***getDetailedReport()*** method (which is responsible for generating detailed report) and a ***getSummaryReport()*** method (which is responsible for generating summary report). Implementation of these method differs in each class.

Each class has a *HashMap* (except the Bank class as there is only one Bank) and a *List* as their data members for their specific attributes. The values of the list depend on what the user choose from the drop-down menu. The class extending a class has its own *HashMap* and *List* as well as the *HashMap* and *List* of the parent class. The flowchart [**Fig (i)**] depicts a clearer idea of the above.

SqlQuery:

The *SqlQuery* class has a data member of type String which is supposed to contain the SQL query. The ***sql()*** method of the class establishes a connection with the database and send the query for execution. The result of the executed query is saved as a ***ResultSet,*** which is then processed as per the user's demand.

Controllers:

DetailedReportController: -

Manages the detailed report generation. Binds the data to each column of the generated report. Also, the action listeners in the initialize method of the class manages the add and/or remove column functionality and the filters.

SummaryReportController: -

Manages the summary report generation. Binds the data to each column of the generated report. Also, the action listeners in the initialize method of the class are responsible for the filters.