**Canadian Institute of Technology**

Faculty of Engineering

Department of Software Engineering

****

**Bank Management System (BMS)**

A project submitted in partial fulfillment   
of the requirements for the **Fundamentals of Programming II course**   
in the 2nd semester of the 2nd year of bachelor

**by**

Irin Vokopola

Kelvin Berami

**Supervised by**

Evis Plaku

# ABSTRACT

Bank Management System is a simply recreation of an existing system such as the one of BKT (Banka Kombetare Tregetare) is. Our aim was to do some simple CRUD functions with JDBC but also using Java Collections, regular expressions and lambda expressions. We decided that a bank management system was the most efficient and creative way on working with databases for the main reason that it has more options of data to store and types of this data you can store (Varchar, Int, Date, Time).

This system does basic functions for both the employer and for the employee. The employer can see the list of all the employees, check the total balance, and add or delete an employee. On the other hand an employee, can deposit or withdraw a certain amount of money, and can also check his transaction history and his balance.

If we consider adding/deleting, deposit/withdraw such as mini problems, we can proudly say that these problems were solved in the simplest and securest way possible. We realized the connection between MySQL Workbench 8.0 and Java (using IntelliJ) will a mysql-connector library. Then for the way display things, we decided to switch from IntelliJ to NetBeans, for the only reason that it is simpler to use and more practical in the second one than in the first one.

This is a simple but completed program for its requirements. We were asked to create a program where we used Java regular expression, Java Collections, Java GUI, JDBC and Java lambdas and expressions. Looking back at what we learned during 1 year of programming courses at school, and looking back at the requirements this is a fulfilled program for the main reason that it has all of the libraries and technologies we were required to use.

Contents

[ABSTRACT ii](#_Toc105854558)

[1. Introduction 1](#_Toc105854559)

[1.1 Introduction parts 1](#_Toc105854560)

[1.1.1 Topic 1](#_Toc105854561)

[1.1.2 Focus ad Scope 1](#_Toc105854562)

[1.1.3 Relevance and importance 1](#_Toc105854563)

[1.1.4 Questions and Objectives 1](#_Toc105854564)

[1.1.5 Photos 2](#_Toc105854565)

[1.2 Purpose 5](#_Toc105854566)

[1.3 Document Conventions 5](#_Toc105854567)

[1.4 Intended Audience and Reading Suggestions 6](#_Toc105854568)

[2. Overall Description 6](#_Toc105854569)

[2.1 Product Perspective and Scope 6](#_Toc105854570)

[2.2 Product Functionalities 7](#_Toc105854571)

[2.3 Operating Environment 7](#_Toc105854572)

[3. Project Development / Method Explanation 7](#_Toc105854573)

[3.1 First Section - Brainstorm 7](#_Toc105854574)

[3.2 Second Section - Coding 8](#_Toc105854575)

[4. Conclusions / Discussion 8](#_Toc105854576)

[Major Contributions 8](#_Toc105854577)

[5. References 9](#_Toc105854578)

# Introduction

## Introduction parts

### 1.1.1 Topic

This project topic is "Management System" which according to the file FPII-FinalProject.pdf submitted on "Fundamentals of Programming II", contains the following concepts:

* Text manipulation using strings and regular expressions
* Generic collections(in this case lists) and generic methods
* Managing information through a database management system and SQL
* Graphical User Interface

Bank Management system is nothing more than a usual management system. We can find management systems everywhere, at shops, in the pharmaceutical field, at restaurants etc. It is a simple way to organize things and know when something happened, why happened and who did it. In the raw example, we have BMS. In this system you can see an employee (who did it), that deposited (what happened) and a certain time (when happened).

Below we will see all the requirements, technologies used, all the schemas and GUIs but all you need to know until now is that this system is has 2 main users, employers and employees. Employers can do simple CRUD (create read update delete) actions of the employees. The other kind of user, employees (customers), can deposit/withdraw or check their balance.

### 1.1.2 Focus ad Scope

As we mentioned we focused on those concepts. As for text manipulation using strings and regular expression, we achieved this part by making such regEx-es so that the Name and Surname should contain each only one capitalized word. It cannot start in a lowercase letter. We used generic collections to store data, such as in the case we used Array Lists to store the information the program gets when it reads the MySQL database. We used to manage the information through a database management system and SQL, so that the information stored is more secure and safe. We used JDBC to connect Java with MySQL and a mysql-connector. We also used WAMP Server to create a localhost connection for the only reason that we could use phpmyadmin to manage better our database tables.

### 1.1.3 Relevance and importance

This project is useful on every aspect possible.

If we discuss about code, libraries and technologies used, this project is an example for all those who want to understand how some things are done e.g.: when someone wants to know how to CRUD Time and Date type of variables.

If we mention where this program can be used, it can be used everywhere. It is a small step to create a banking system but still it is a start.

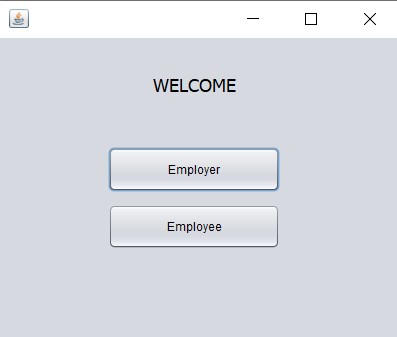
It can be an inspiration for those who want to create simple and reliable programs but do not know where to start.

### 1.1.4 Questions and Objectives

The main purpose of this project was to create a simple and effective system that includes all of the requirements, we the least lines of code possible, without compromising the security and the reliability of the program itself. This purpose was achieved by continuous work, learning and researching on what are the securest ways to achieve the problems we wanted to solve.

### 1.1.5 Photos

#### 1.1.5.1 GUIs

In this section you will see some of the main GUI (Graphical User Interface) the program has.

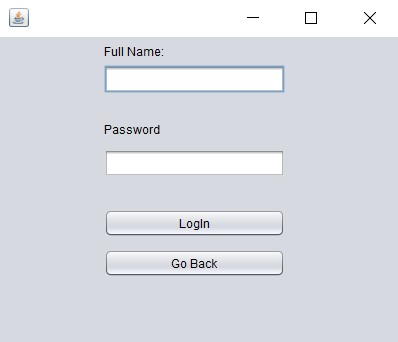
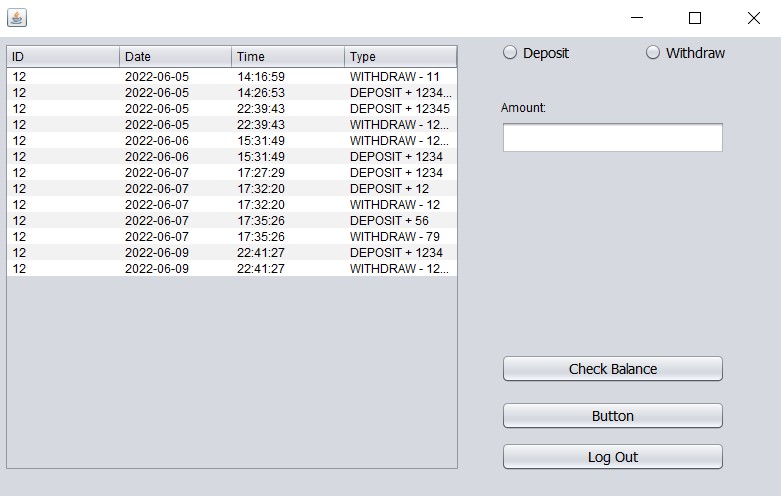
Figure 1: LogIn Interface

Figure 2: Employee Interface

Figure 3: Employee Menu Interface

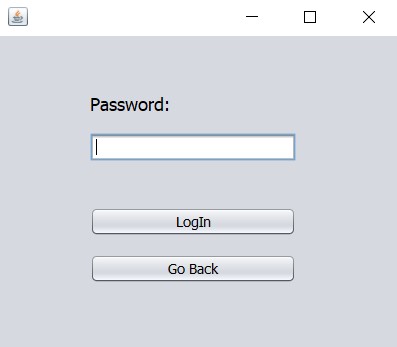


Figure 4: Employer Interface

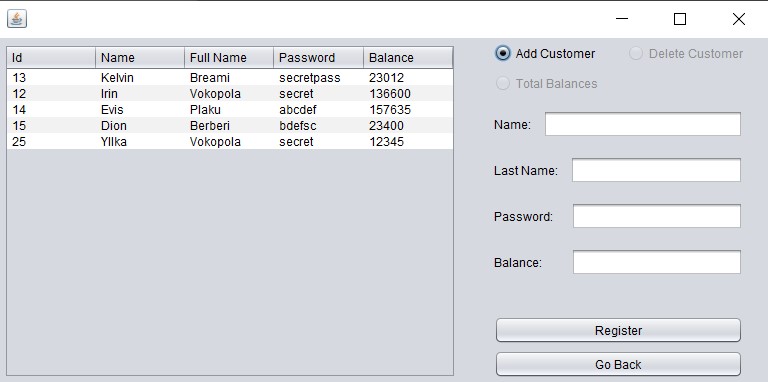


Figure 5: Employer Menu Interface

#### 1.1.5.2 Code

Figure 6: Code

Bms directory - a code where the display is terminal-based

Pages directory - bms code where the display is gui-based

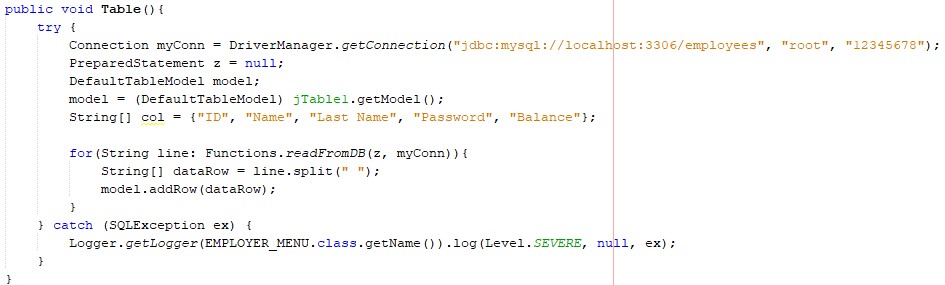


Figure 7: Method to populate a jTable



Figure 8: Code to check whether the employer password exists

## Purpose

The purpose of BMS is to help different categories on how some things work.

Firstly, the main goal is to help other developers, students with lines of simple code (because the program is deployed in GitHub), who can get the code, test it, get and understand the logic behind some operations and understand (from this document and from the ReadMe file) how a proper code/program/system/software documentation is done.

Secondly, we want to show to others also that the system works no matter how simple the graphical user interface is. We know this program is not an inspiration for UI/UX designers, but big things require time.

Thirdly, the other category we wanted to reach was to show our professors that we are capable of doing effective and kind of complex programs, no matter on how much pressure we are or how much time we have.

## Document Conventions

The document is worked on Microsoft Word Office 2013.

All the document is written in the font "Times New Roman".

Heading 1 (Introduction, Overall Description ...) - is bolded, size 18, number pointed.

Heading 2 (Introduction parts, Purpose ...) - is bolded, size 14, number pointed.

Heading 3 (Topic, Focus and Scope ...) - is bolded, size 12, number pointed.

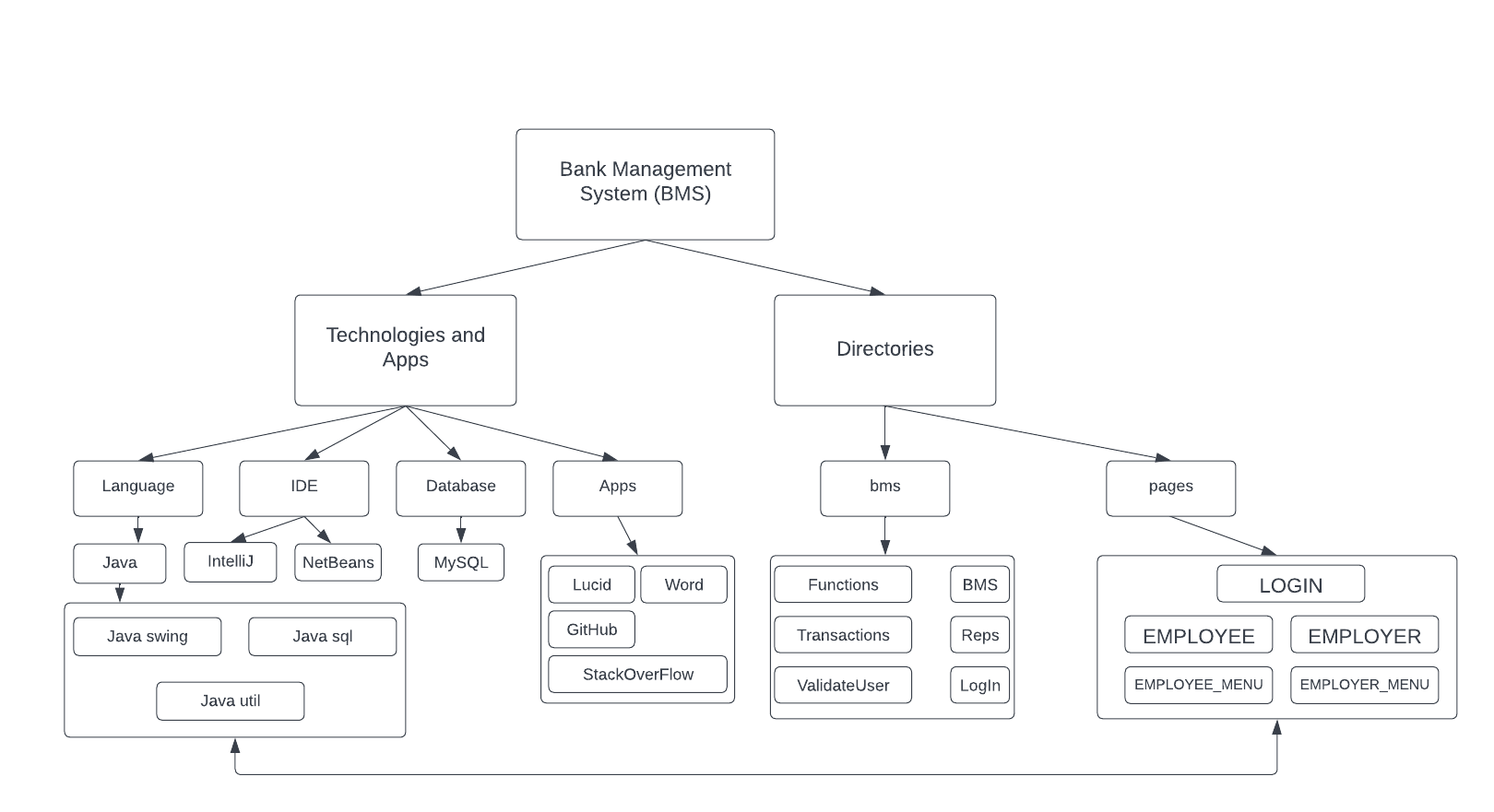
Heading 4 (GUIs, Code ...) - is bolded and italic, size 11, number pointed.

## Intended Audience and Reading Suggestions

This program with this kind of documentation is intended for all the ages and for all the different type of knowledge people have, because the documentation is explained very clearly and the code is commented in a way that everyone can understand it.

# Overall Description

## Product Perspective and Scope

The program is divided in two folders, called bms and pages respectively. The first folder contains java classes, 6 of them exactly. The second folder contains 5 jForm classes (GUIs). Why is there a need of having 2 folders in the same project? The folders are different from one another. They have the same functionalities and are connected to the same database, but the first one, bms directory, has classes that display all the information and actions using the terminal and take input from Scanner. The second directory takes input from jTextFields, jPasswordFields, jButton and jRadioButtonand displays the data in GUI (tables).

## Product Functionalities

The most familiar functionalities for a developer, in relation with databases are: CRUD.

CRUD (Create Read Update Delete) can be defined in SQL syntax as INSERT, SELECT, UPDATE, And DELETE.

Users, both employer and employees, can make such actions. Employers can add/delete/check the employees. Employees can deposit/withdraw/check balance.

## Operating Environment

Firstly to have any project running you need a laptop or a desktop with at least 4 GB of RAM and a stable connection to internet.

BMS is a Java program, and to exist and to function it needs a Java IDE. We decided to develop the main methods and functionalities using IntelliJ IDE, and NetBeans IDE to develop the GUI side of the program.

This program in particular cannot exist without haven a database, and a connection to this database. We used MySQL workbench 8.0, to create the database and its tables, we then used mysql-connector to connect MySQL to IntelliJ and NetBeans.

# Project Development / Method Explanation

In your report, you will have to discuss the methods you used to develop your software. The methodology chapter explains what you did and how you did it, allowing readers to evaluate the [reliability and validity](https://www.scribbr.com/methodology/reliability-vs-validity/) of the work. It includes:

* The[type of work](https://www.scribbr.com/methodology/types-of-research/) you did
* How you [collected your data](https://www.scribbr.com/category/methodology/#examples-of-data-collection-methods), if any
* How you [analyzed your data](https://www.scribbr.com/category/methodology/#examples-of-data-analysis-methods), if any
* Any tools or materials you used in the work
* Your rationale for choosing these methods
* A thorough explanation of the key steps of your work

You may break this section into several sub-sections as the one used below. Feel free to add or delete subsections accordingly.

## First Section - Brainstorm

The first Section of this system is to brainstorm all the things. What we need to do and how ne need to do it. As we talked about requirements , we knew where to focus and as we talked about how time we had in disposition, we know where to work and how to distribute work among the two of us team members.

We decided to create a banking system, that will have the basic CRUD operations, connect this application to MySQL, provide comments with the code and a proper documentation.

We decided that we needed to have 2 sections, such as employers and employees. Then we thought about the functionalities. As we all have used banks, we know that a customer can deposit, withdraw or check his balance (basic operations). On the other hand, we thought of the bank as we were part of the management of a bank. We thought that the actions we need to make is to add, delete and check the employers.

## Second Section - Coding

Firstly, as we saw our working plan, we tried to be organized. All the functionalities in one class (Transactions, Functionalities), the validation of user name and last name in one class (ValidateUser), the display of the information (Reps), the log in in one class (LogIn) and a main class where to run the project(BMS).

Secondly when we started coding we thought that displaying the information on the terminal would be enough but then we thought of other ways displaying this kind of information, and GUI was the easiest one. This part of the project, coding lasted 10 days, on each day programming for 2 hours, so in total 20 hours, plus an addition 3 to make the GUI and the error detection.

Finally, making GUI classes was not that difficult because we kept it simple. We got the logic and the code from the classes we discussed in the first part of this section, and manipulated this code accordingly to GUI components logic.

# Conclusions / Discussion

Looking back at this project, from the day we were informed about this project, until the date we submitted it, we are happy with our work. We tried and maintained the work plan we designed since the start. WE knew when to work and what to do. We used GitHub service to work on the same project from different places, remotely. We think that this project is done correctly, within the limits and the scope of this course.

We know that there are things that we can improve, such as adding an option to update employees, adding some testing classes to our project, removing GUIs and displaying the information in a Website.

# Major Contributions

**Irin Vokopola**

* Brainstorm
* Code

Kelvin Berami

* Brainstorm
* Documentation

# References

BKT. Retrieved from Banka Kombetare Tregtare: https://www.bkt.com.al/

Java. Retrieved from Java: https://www.java.com/en/

IntelliJ. Retrieved from IntelliJ: https://www.jetbrains.com/idea/

NetBeans. Retrieved from NetBeans: https://netbeans.apache.org/

WampServer. Retrieved from Wamp: https://www.wampserver.com/en/

*Java Swing*. (2022, February 9). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Swing\_(Java)

*Java Text*. (2020). Retrieved from Oracle: https://docs.oracle.com/javase/7/docs/api/java/text/package-summary.html

*Java Util*. (2020). Retrieved from Oracle: https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html

*Microsoft Windows*. (n.d.). Retrieved from Microsoft: https://www.microsoft.com/en-us/windows/?r=1

MySQL. (n.d.). Retrieved from MySQL: https://www.mysql.com/