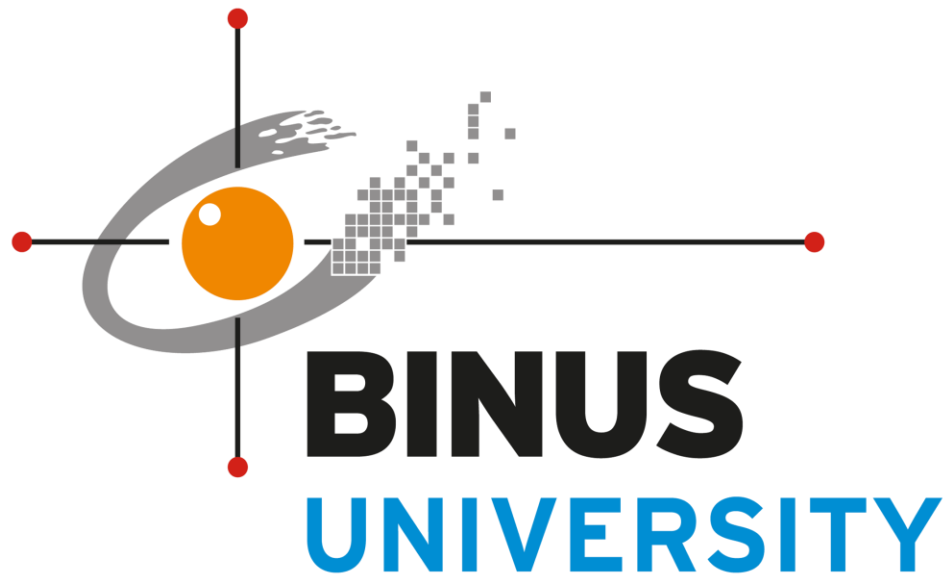


# **OBJECT ORIENTED PROGRAMMING FINAL PROJECT REPORT**



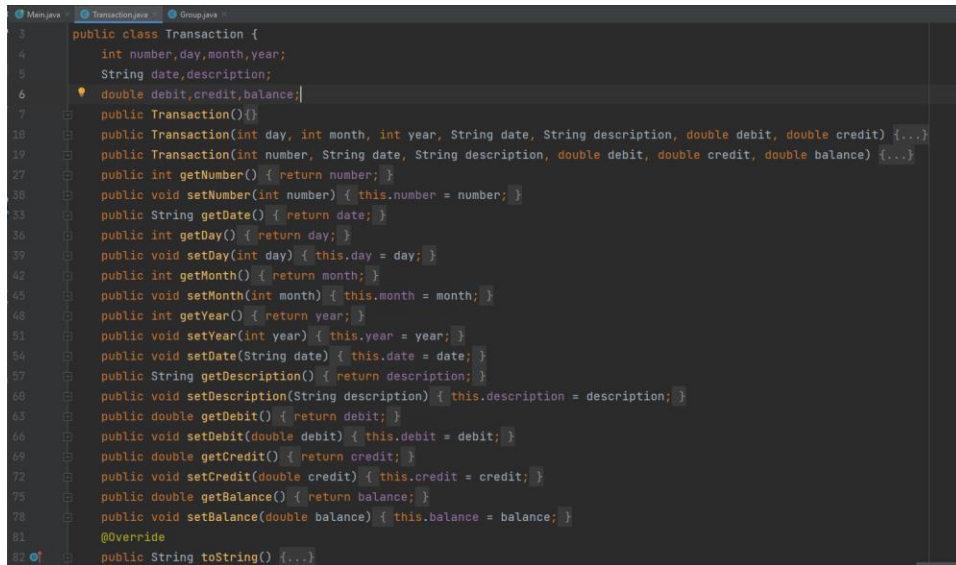
**BY: ARVIN YUWONO (2502009721)**

The application is a digitalized version of *Buku Kas*, a physical book that records the income, expense and balance per each transaction made by a household or company. Its purpose is to monitor the flow of money within the household or company. This application was created to make it easier for everyone to be able to do so without going through the troublesome effort of writing formulas in Excel. The access a User has to the application depends on the level of authorization they have been granted.

[illegible]

## Solution Design

### Transaction Class



```

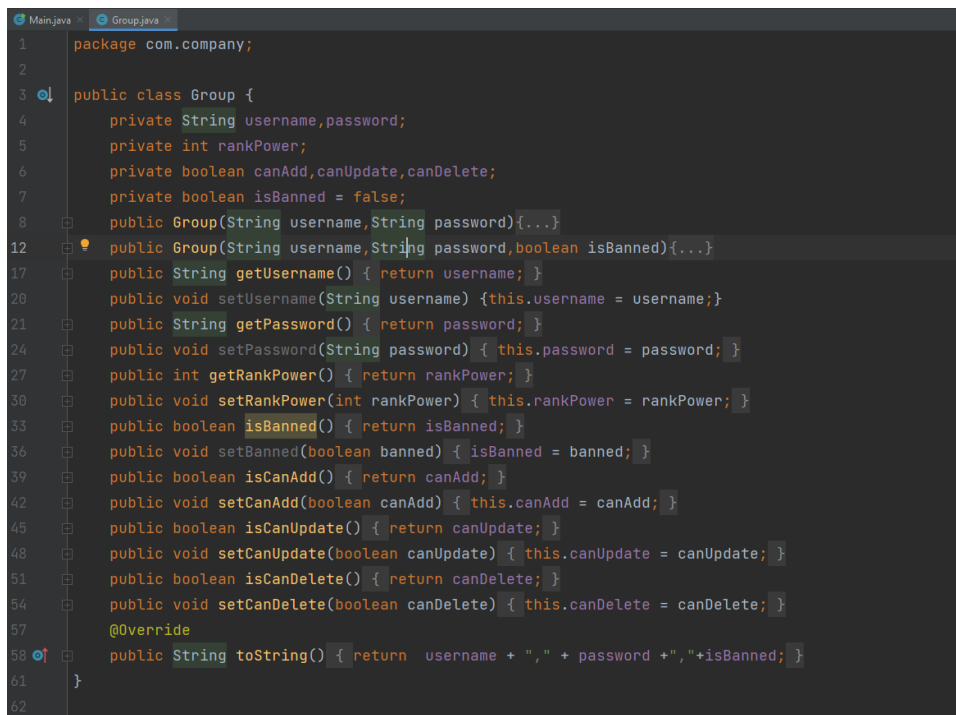
1 public class Transaction {
2     int number, day, month, year;
3     String date, description;
4     double debit, credit, balance;
5     public Transaction() {}
6     public Transaction(int day, int month, int year, String date, String description, double debit, double credit) {...}
7     public Transaction(int number, String date, String description, double debit, double credit, double balance) {...}
8     public int getNumber() { return number; }
9     public void setNumber(int number) { this.number = number; }
10    public String getDate() { return date; }
11    public int getDay() { return day; }
12    public void setDay(int day) { this.day = day; }
13    public int getMonth() { return month; }
14    public void setMonth(int month) { this.month = month; }
15    public int getYear() { return year; }
16    public void setYear(int year) { this.year = year; }
17    public void setDate(String date) { this.date = date; }
18    public String getDescription() { return description; }
19    public void setDescription(String description) { this.description = description; }
20    public double getDebit() { return debit; }
21    public void setDebit(double debit) { this.debit = debit; }
22    public double getCredit() { return credit; }
23    public void setCredit(double credit) { this.credit = credit; }
24    public double getBalance() { return balance; }
25    public void setBalance(double balance) { this.balance = balance; }
26    @Override
27    public String toString() {...}

```

Image 2.1 Transaction Class

This class contains data types which can store details of a transaction such as the number, date, description, debit, credit, and balance. A Constructor with and without parameter is available to be used. Getter and Setter methods are also there to access the private datas. A to String method is given for testing on console.

## Group Class



```

1 package com.company;
2
3 public class Group {
4     private String username, password;
5     private int rankPower;
6     private boolean canAdd, canUpdate, canDelete;
7     private boolean isBanned = false;
8     public Group(String username, String password) {...}
9     public Group(String username, String password, boolean isBanned) {...}
10    public String getUsername() { return username; }
11    public void setUsername(String username) { this.username = username; }
12    public String getPassword() { return password; }
13    public void setPassword(String password) { this.password = password; }
14    public int getRankPower() { return rankPower; }
15    public void setRankPower(int rankPower) { this.rankPower = rankPower; }
16    public boolean isBanned() { return isBanned; }
17    public void setBanned(boolean banned) { isBanned = banned; }
18    public boolean isCanAdd() { return canAdd; }
19    public void setCanAdd(boolean canAdd) { this.canAdd = canAdd; }
20    public boolean isCanUpdate() { return canUpdate; }
21    public void setCanUpdate(boolean canUpdate) { this.canUpdate = canUpdate; }
22    public boolean isCanDelete() { return canDelete; }
23    public void setCanDelete(boolean canDelete) { this.canDelete = canDelete; }
24    @Override
25    public String toString() { return username + ", " + password + ", " + isBanned; }
26 }

```

Image 2.2 Group Class

Act as the parent of actual user groups. It holds data type for the name, password, rank, right of accesses, and ban status.

### Superadmin Subclass

```
1 package com.company;
2
3
4
5 public class Superadmin extends Group implements Staff{
6     public Superadmin(String username,String password){
7         super(username,password);
8         this.setRankPower(2);
9         this.setCanAdd(true);
10        this.setCanUpdate(true);
11        this.setCanDelete(true);
12    }
13    public Superadmin(String username, String password,boolean isBanned) {
14        super(username,password,isBanned);
15        this.setRankPower(2);
16        this.setCanAdd(true);
17        this.setCanUpdate(true);
18        this.setCanDelete(true);
19    }
20
21    @Override
22    public boolean isStaff() { return Staff.super.isStaff(); }
23
24 }
25
26
```

Image 2.3 Superadmin Subclass

The group's child with the highest authority of access rights, which means it can access every form and button. This class can access the Staff Panel.

### Admin Subclass

```

1  package com.company;
2
3  public class Admin extends Group implements Staff{
4
5      public Admin(String username, String password) {
6          super(username,password);
7          this.setRankPower(1);
8          this.setCanAdd(true);
9          this.setCanUpdate(true);
10         this.setCanDelete(false);
11     }
12     public Admin(String username, String password,boolean isBanned) {
13         super(username,password,isBanned);
14         this.setRankPower(1);
15         this.setCanAdd(true);
16         this.setCanUpdate(true);
17         this.setCanDelete(false);
18     }
19
20     @Override
21     public boolean isStaff() { return Staff.super.isStaff(); }
22
23
24
25
26 }

```

Image 2.4 Admin Subclass

The Group's child with a medium authority of access rights, with the only restricted access of being unable to delete or clear items. This class can access the Staff Panel.

### Operator Subclass

```

1  package com.company;
2
3  public class Operator extends Group implements Member{
4
5      public Operator(String username,String password){
6          super(username,password);
7          this.setRankPower(0);
8          this.setCanAdd(true);
9          this.setCanUpdate(false);
10         this.setCanDelete(false);
11     }
12
13     public Operator(String username,String password,boolean isBanned) {
14         super(username,password,isBanned);
15         this.setRankPower(0);
16         this.setCanAdd(true);
17         this.setCanUpdate(false);
18         this.setCanDelete(false);
19     }
20
21
22 }
23

```

Image 2.5 Operator Subclass

Group's child with the lowest authority of access rights. This subclass is only able to add items.

### User Class

```

5 public class User {
6     private String name,rank,password;
7     private int rankPower;
8     private boolean isBanned;
9
10    public User(String name, String rank,boolean isBanned,String password,int rankPower) {...}
17
18    public String getName() { return name; }
21
22    public void setName(String name) { this.name = name; }
25
26    public String getRank() { return rank; }
29
30    public void setRank(String rank) { this.rank = rank; }
33
34    public boolean isBanned() { return isBanned; }
37    public String getIsBanned() { return String.valueOf(isBanned); }
40
41    public void setBanned(boolean banned) { isBanned = banned; }
44
45    public String getPassword() { return password; }
48
49    public int getRankPower() { return rankPower; }
52
53    public void setRankPower(int rankPower) { this.rankPower = rankPower; }
56 }
57

```

Image 2.6 User Class

Used to help group all separate user groups into a new class to be able to display all available accounts on a single table.

### Staff Interface

```

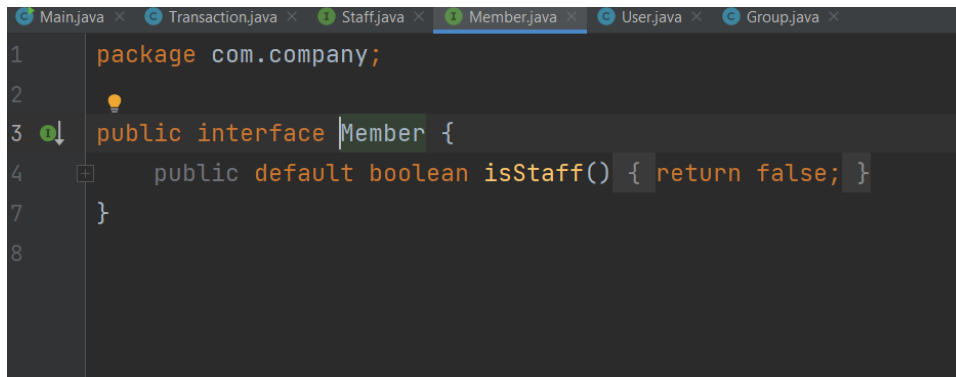
1 package com.company;
2
3 public interface Staff {
4     public default boolean isStaff() { return true; }
7 }
8

```

Image 2.7 Staff Interface

This interface contains a default Boolean method, which returns true. Used by subclasses such as Admin and Superadmin to recognize it is considered a staff and should be able to access the staff panel.

### Member Interface



```


1  package com.company;
2
3  public interface Member {
4      public default boolean isStaff() { return false; }
5
6  }
7
8

```

Image 2.8 Member Interface

This interface contains a default Boolean method, which returns false. Used by subclasses such as Operator to recognize it is a member and not a Staff.

### Main Form



```

1  package com.company;
2
3
4  import java.io.FileNotFoundException;
5
6  public class Main {
7
8      public static void main(String[] args) throws FileNotFoundException {
9          LoginForm loginForm = new LoginForm(isVisible: true); //Open the login screen
10     }
11 }
12

```

Image 2.9 Main Screen

This form contains the main function which opens the Login Form (Screen) upon execution of the application.

### Registration Form



```
Main.java x FileForm.java x BookForm.java x KasForm.java x LoginForm.java x RegistrationForm.java x Transaction.java x Admin
27 private final String operatorKeysTextFile = "AppData/Accounts/operatorKeys.txt";
28 private final String adminKeysTextFile = "AppData/Accounts/adminKeys.txt";
29 private final String superadminKeysTextFile = "AppData/Accounts/superadminKeys.txt";
30
31
32 private final String operatorAccountsTextFile = "AppData/Accounts/operatorAccounts.txt";
33 private final String adminAccountsTextFile = "AppData/Accounts/adminAccounts.txt";
34 private final String superadminAccountsTextFile = "AppData/Accounts/superadminAccounts.txt";
35
36 public String getOperatorAccountsTextFile(){return operatorAccountsTextFile;}
37 public String getAdminAccountsTextFile(){return adminAccountsTextFile;}
38 public String getSuperadminAccountsTextFile(){return superadminAccountsTextFile;}
39
40 private ArrayList<String> operatorKeys = new ArrayList<>();
41 private ArrayList<String> adminKeys = new ArrayList<>();
42 private ArrayList<String> superadminKeys = new ArrayList<>();
43
44 private HashMap<String,ArrayList<String>> operatorMap = new HashMap<>();
45 private HashMap<String,ArrayList<String>> adminMap = new HashMap<>();
46 private HashMap<String,ArrayList<String>> superadminMap = new HashMap<>();
47 private ArrayList<Operator> operatorArrayList = new ArrayList<>();
48 private ArrayList<Admin> adminArrayList = new ArrayList<>();
49 private ArrayList<Superadmin> superadminArrayList = new ArrayList<>();
50
51
52 public RegistrationForm(boolean isVisible) throws FileNotFoundException {...}
96 void LoginPage() throws FileNotFoundException {...}
99 @ private ArrayList<String> getKeys(String textFile) throws FileNotFoundException {...}
109 @ private void setKeys(ArrayList<String> keysArrayList, String textFile) throws IOException {...}
116 @ public HashMap<String,ArrayList<String>> getAccounts(String textFile) throws FileNotFoundException {...}
130 @ public ArrayList<Operator> getOperatorAccounts(HashMap<String,ArrayList<String>> accountsHashMap){...}
139 @ public ArrayList<Admin> getAdminAccounts(HashMap<String,ArrayList<String>> accountsHashMap) {...}
148 @ public ArrayList<Superadmin> getSuperadminAccounts(HashMap<String,ArrayList<String>> accountsHashMap) {...}
157 @ public void saveAccount(int groupIndex, String textFile) throws IOException {...}
181 @ private boolean registerUser() throws IOException {...}
227
228
229
```

Image 2.10 Registration Screen

File and Scanner are used to read text files that contain serial keys for either Operator, Admin or Superadmin, which are then added into their respective ArrayList<String>. File Writers are then used to store username and password into respective text file based on the serial key input. The user can be returned to the Login Form by either pressing the ‘LOGIN’ button or submitting the registration.

## Login Form

```

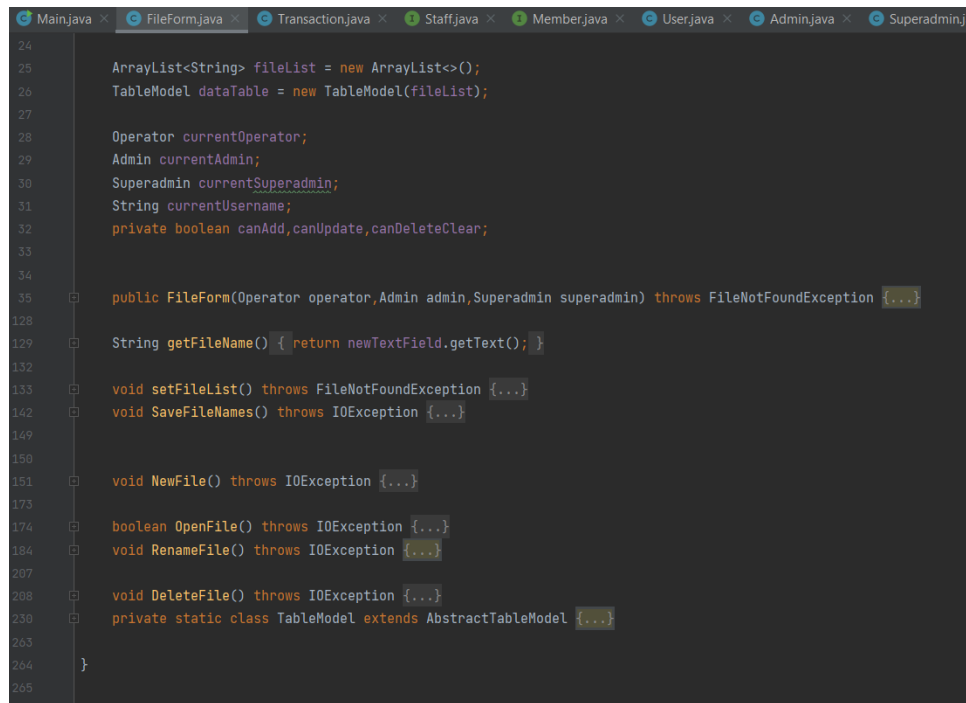
12 public class LoginForm extends Component {
13     private RegistrationForm registrationForm = new RegistrationForm(false); //to access get and set account functions
14     //store the user groups
15     private ArrayList<Operator> operatorArrayList = new ArrayList<>();
16     private ArrayList<Admin> adminArrayList = new ArrayList<>();
17     private ArrayList<Superadmin> superadminArrayList = new ArrayList<>();
18     // file's path directory
19     private final String operatorAccountsTextFile = registrationForm.getOperatorAccountsTextFile();
20     private final String adminAccountsTextFile = registrationForm.getAdminAccountsTextFile();
21     private final String superadminAccountsTextFile = registrationForm.getSuperadminAccountsTextFile();
22
23
24     private JPanel loginPanel;
25     private JTextField usernameField;
26     private JPasswordField passwordField;
27     private JButton loginButton;
28     private JButton registryButton;
29
30     //initialize values to allow parameter input to other forms
31     private Operator currentOperator = null;
32     private Admin currentAdmin = null;
33     private Superadmin currentSuperadmin = null;
34
35     //temporary store for conversion
36     private HashMap<String,ArrayList<String>> operatorHashMap = new HashMap<>();
37     private HashMap<String,ArrayList<String>> adminHashMap = new HashMap<>();
38     private HashMap<String,ArrayList<String>> superadminHashMap = new HashMap<>();
39
40     public LoginForm(boolean isVisible) throws FileNotFoundException {...}
78     private boolean loginAccount() throws FileNotFoundException {...}
101 void filePage() throws FileNotFoundException {...}

```

Image 2.11 Login Screen

File and Scanner are used to read text files that contain account's username and password, which are then put into a `HashMap<String, String>`. An object of type Operator, Admin or Superadmin will then be created and added into their respective ArrayList. A valid login detail will allow the user to be brought to the File Form. If no account exists, the user can press the 'REGISTRY' button to be brought to the Registration Form.

## File Form



```
24
25     ArrayList<String> fileList = new ArrayList<>();
26     TableModel dataTable = new TableModel(fileList);
27
28     Operator currentOperator;
29     Admin currentAdmin;
30     Superadmin currentSuperadmin;
31     String currentUsername;
32     private boolean canAdd, canUpdate, canDeleteClear;
33
34
35     public FileForm(Operator operator, Admin admin, Superadmin superadmin) throws FileNotFoundException {...}
128
129     String getFileName() { return newTextField.getText(); }
132
133     void setFileList() throws FileNotFoundException {...}
142     void SaveFileNames() throws IOException {...}
149
150
151     void NewFile() throws IOException {...}
173
174     boolean OpenFile() throws IOException {...}
184     void RenameFile() throws IOException {...}
207
208     void DeleteFile() throws IOException {...}
230     private static class TableModel extends AbstractTableModel {...}
263
264 }
265
```

Image 2.12 File Selection Screen

Inside a Scroll Panel and Table, based off an ArrayList, all the books available to be opened will be displayed. The user can create a new book by inputting the desired name in a Text Field and pressing the ‘New’ button. To open an existing file, select a row in the table and press the ‘Open’ button, which will bring the user to the Kas Form. The first course of action is the same as opening for file renaming and deletion, except the button to be pressed differs.

## Book Form

```

28  private JButton clearButton;
29  Ab private JLabel userLabel;
30  Ab private JLabel bookLabel;
31  private JButton EXITButton;
32
33  private ArrayList<Transaction> transactionArrayList = new ArrayList<>();
34  private String currentFile, displayName;
35  private boolean canAdd, canUpdate, canDeleteClear;
36
37  BookForm(String textFile, Operator operator, Admin admin, Superadmin superadmin) throws FileNotFoundException {...}
154
155
156  void OpenBook(String textFile) throws FileNotFoundException {...}
181
182  boolean isTextFieldEmpty(){...}
189
190  private int ComboBoxToInteger(Object comboBox) { return Integer.parseInt(String.valueOf(comboBox)); }
193
194  private int CalculateNumber(int currentDay, int currentMonth, int currentYear){...}
232
233  private void RecalculateNumberAndBalance(){...}
252  void AddTransaction(){...}
279  void UpdateTransaction(){...}
308  void DeleteTransaction(){...}
315  void ClearTransactions(){...}
320  void SaveTransactions() throws IOException {...}
328
329
330  private static class TableModel extends AbstractTableModel{...}
364  }

```

Image 2.13 *Buku Kas* Screen

The list of buttons (Add, Update, Delete, and Clear) available will depend on the current user's rank. The date can be adjusted by using the available Combo Box, and the same goes for the type of transaction: Debit or Credit. As for money and description, they can be written in the Text Fields. When a button is pressed, it will convert the Combo Boxes' selected item and Text Fields' Text into the appropriate data types.

When a new transaction is to be added, it will check whether the `ArrayList<Transaction>` is empty first. If it is not empty, it will first calculate the appropriate number and index for it to be inserted at. Else If it is empty, it will immediately add the transaction to the `ArrayList`. After either course of flow, it will recalculate every transaction's appropriate number and balance before updating the Table.

The difference between updating and deleting a transaction is that updating removes the selected existing transaction and adds a new one, while deleting a transaction only removes the selected transaction.

## Staff Form

```

36 public StaffForm(Operator operator, Admin admin, Superadmin superadmin) throws FileNotFoundException {...}
104
105 void setUserList() throws FileNotFoundException {...}
123 @ void SaveHashmap(String rank, User user){...}
134 void SaveToFile() throws IOException {...}
161
162
163 String getNextRank(String currentRank){...}
170 String getPreviousRank(String currentRank){...}
178
179 boolean isTargetValid(int index){...}
183
184
185
186
187
188
189
190 void PromoteRank() throws IOException {...}
203 void DemoteRank() throws IOException {...}
215
216 void BanAndUnban() throws IOException {...}
227
228
229
230
231
232
233 private static class TableModel extends AbstractTableModel {...}
264 void UpdateTable() { dataTable.fireTableDataChanged(); }
267
268

```

Image 2.14 Staff Panel Screen

This Form is exclusively visible to users whose rank is part of Staff (Admin and Superadmin). All users regardless of ranks will be displayed in a Table, based off `ArrayList<User>`, which data is obtained from the same Hashmap method Login Form used and iterated using for each loop to be added into the `ArrayList`.

After selecting a user's row, there are several choices to be made: Promote, Demote, Ban and Unban. Promotion and Demotion can only occur if the target's next rank or previous rank is lower than the user's rank. Ban and unban can be done when the target's rank is below the user's rank. It does not remove the account, but simply toggle a Boolean value to make login impossible.

## Evidence of Working Program

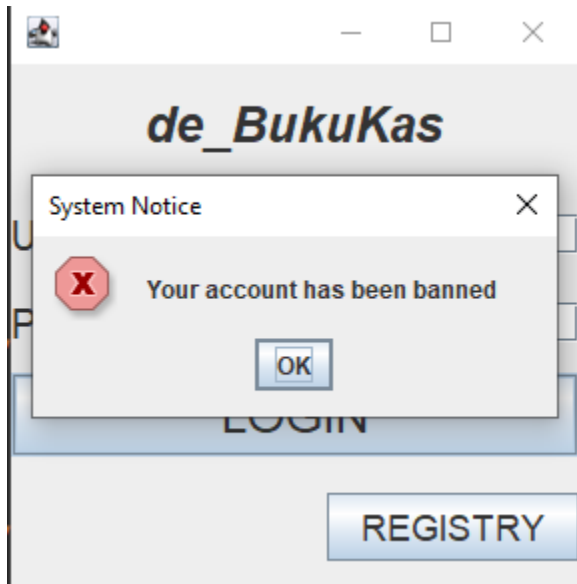


Image 3.1 Unable to login due to account being banned

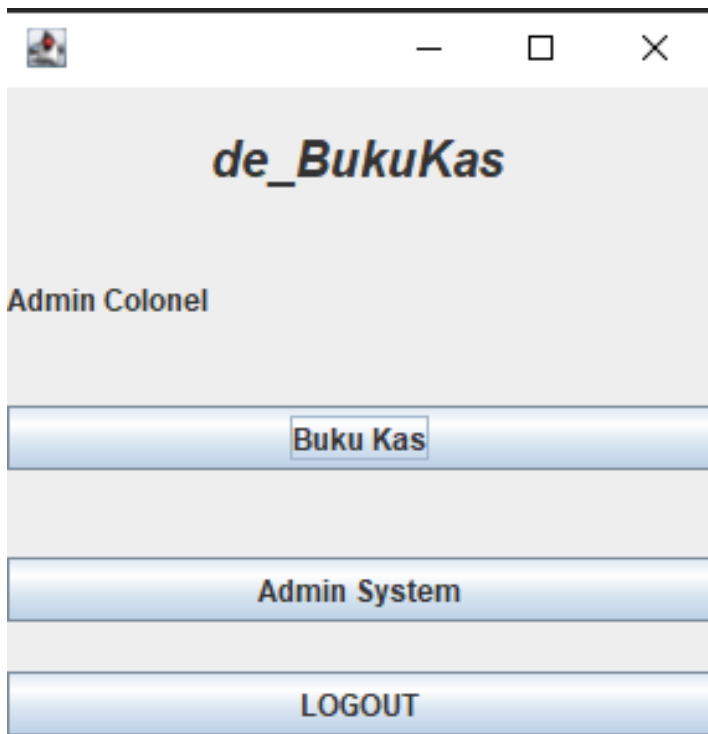


Image 3.2 Menu Screen after successful login

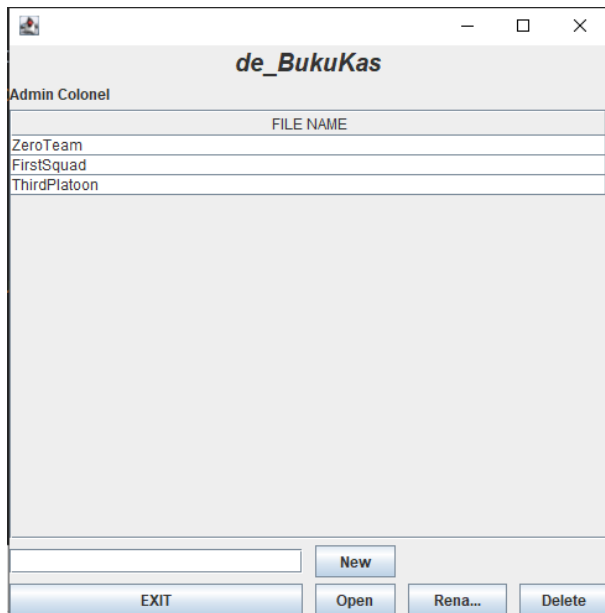


Image 3.3 File Selection Screen after choosing to go to *Buku Kas*

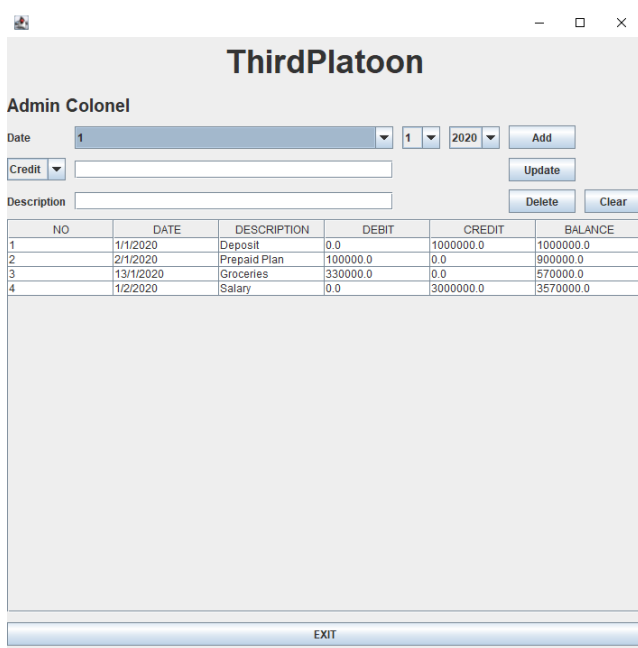


Image 3.4 A *Buku Kas* Screen

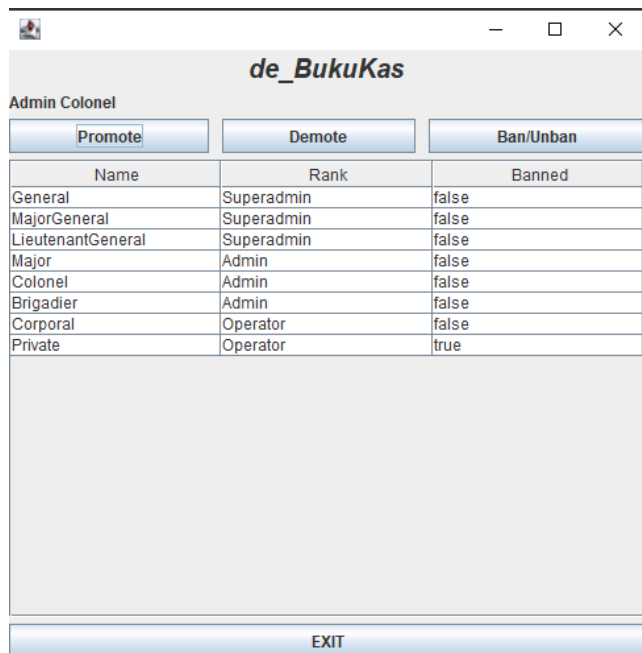


Image 3.5 The Staff Panel Screen

## Resources

[Create Login and Registration Forms Using Java, MySQL and IntelliJ IDEA \(With Source Code\)](#)

<https://www.baeldung.com/java-write-to-file>

[https://www.w3schools.com/java/java\\_hashmap.asp](https://www.w3schools.com/java/java_hashmap.asp)

[https://stackoverflow.com/questions/10960213/how-can-i-read-comma-separated-values-from-a-text-file-in-java#:~:text=You%20may%20use%20the%20String,split\(%22%2C%22\)%3B](https://stackoverflow.com/questions/10960213/how-can-i-read-comma-separated-values-from-a-text-file-in-java#:~:text=You%20may%20use%20the%20String,split(%22%2C%22)%3B)

[Java SWING #19 - How to Exit Program on Button Click in Java Netbeans](#)

<https://www.geeksforgeeks.org/java-program-to-rename-a-file/>