“UNIVERSIDAD CENTRAL DEL ECUADOR”

Topic: UI Java - Wallet

Participants: Kevin Segovia, Leidy Vacacela.

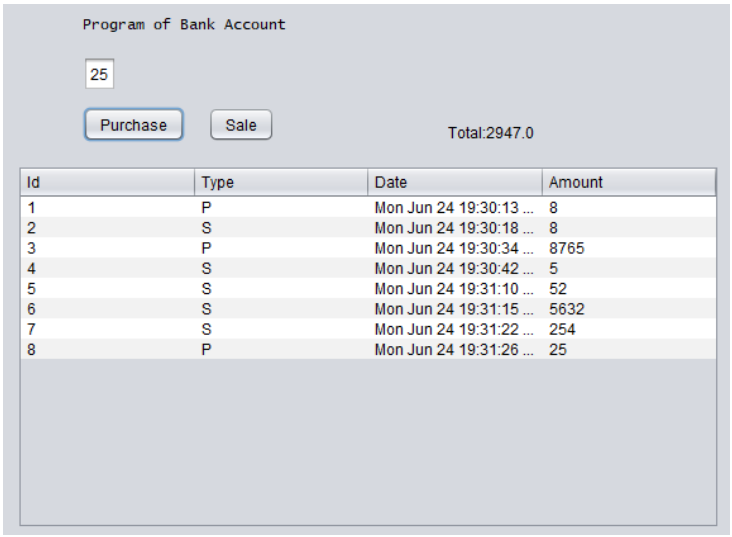
Date: 2024/06

Teacher: Juan Pablo Guevara

Contents.

Program details.

The present program is designed for managing purchases and sales, providing users with a simple and efficient tool to register and monitor their transactions. The application, developed as a desktop application, is compatible with Linux, Windows, and Mac operating systems, as long as the Java JDK is installed. This program facilitates the visualization of the total account value, as well as the addition and subtraction of values through an intuitive graphical interface. Below, the functionalities and features of the program are described in detail.

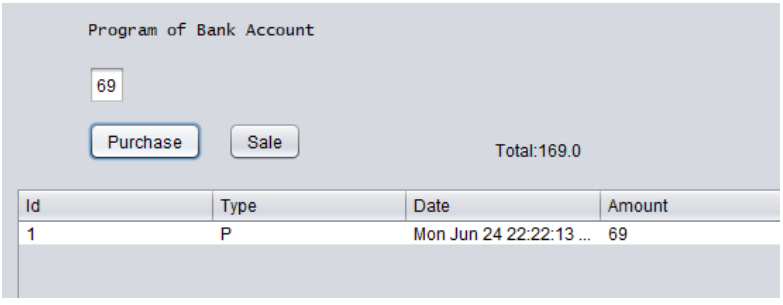


1. **Visualization of the Total Account Value.**

When the program starts, the total account value is set to 100. This value is displayed in a Label object, visible and prominent in the graphical interface, ensuring that the user can easily monitor the current state of their account.

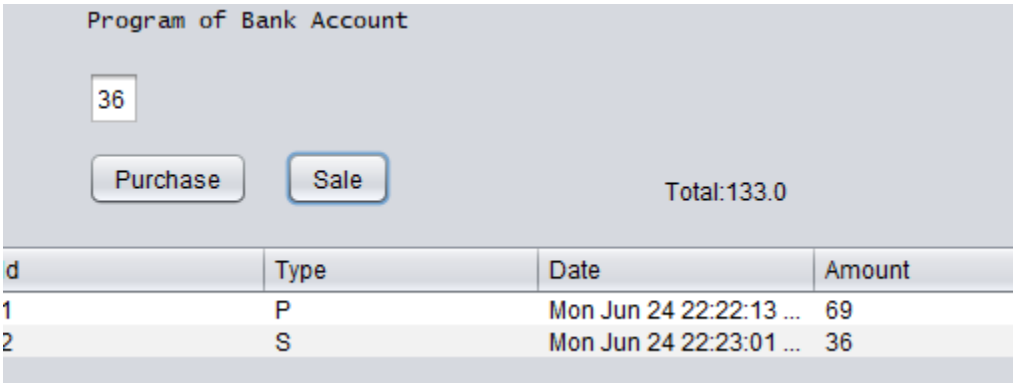
2. Purchasing Functionality.

The interface includes a button labeled "Purchase." The user can enter the purchase amount in a designated text field. When the purchase button is clicked, the program records the transaction in a table. This table displays the purchase amount and other relevant details. The purchase amount is added to the total account value, and the new value is instantly reflected in the Label object.



3. Sales Functionality.

The interface also includes a button labeled "Sales." The user can enter the sale amount in a corresponding text field. When the sales button is clicked, the program records the transaction in the same table used for purchases, adding a new row that reflects the sale amount. The sale amount is subtracted from the total account value, and the new value is displayed in the Label object.



4. Detailed Transaction Recording.

Each transaction recorded in the table receives a unique auto-incremental identifier. Transaction Date: Additionally, the program records the date and time of each transaction, providing a detailed and accurate history.

5. Non-Numeric Input Handling.

The program validates the values entered in the text fields to ensure they are numeric. If a non-numeric value is entered, a tab or pop-up is displayed, warning the user and providing clear instructions to correct the input.

The purchase and sales management program stands out for its simplicity and effectiveness, providing a robust tool for financial transaction management. With an intuitive graphical interface and key features such as real-time visualization of the total account value, detailed recording of purchases and sales, and input validation, the program ensures a smooth and

seamless user experience. Additionally, its compatibility with multiple operating systems and its ability to provide immediate feedback to the user reinforce its utility and accessibility. In summary, this application is an ideal solution for users looking to manage their purchases and sales efficiently and effectively.

Functional Requirements.

- Display a Total Account Value of Sales/Purchases made using the program. Start this account at 100 and show it in the graphical interface.

- Have a button for making purchases (Purchase) that takes the value from the text field where the purchase value is entered. When this button is clicked, proceed to record the purchase in a Table object, reflecting the purchase value. Additionally, add this purchase value to the Account Value, which is shown in a Label object.

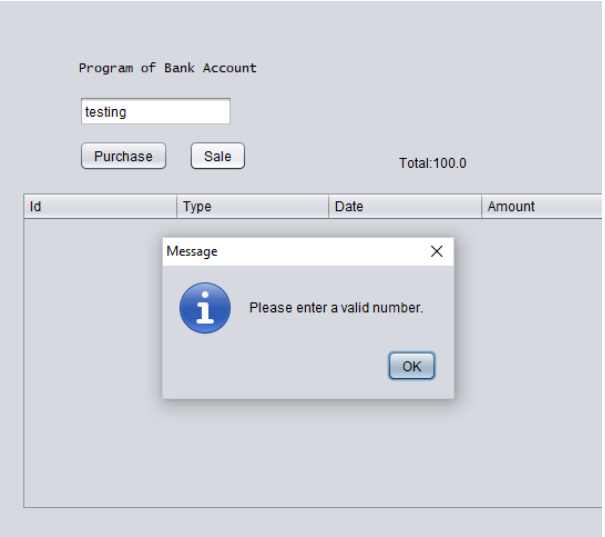
```
private void purchaseButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try{
     String amountText = txtMoney.getText();
     double amount = Double.parseDouble(amountText);
     total+=amount;
     jLabel1.setText("Total:" + total);
     Object[] info = {model.getRowCount() + 1, "P", new java.util.Date(),txtMoney.getText()};
     model.addRow(info);
     model.getRowCount();
    } catch (NumberFormatException ex){
      JOptionPane.showMessageDialog(null,"Please enter a valid number.");
    }
```

- Have a button for making sales (Sales) that takes the value from the text field where the sale value is entered. When this button is clicked, proceed to record the sale in the same Table object used for purchases. This new row reflects the sale value. Additionally, subtract the Sale Value from the Account Value, which is shown in a Label object.

```
private void saleButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        String amountText = txtMoney.getText();
        double amount = Double.parseDouble(amountText);
        total-=amount;
        jLabell.setText("Total:" + total);
        Object[] info = {model.getRowCount() + 1, "S", new java.util.Date(),txtMoney.getText()};
        model.addRow(info);
    }catch(NumberFormatException ex){
        JOptionPane.showMessageDialog(null,"Please enter a valid number.");
    }
```

- Each row of sales and purchases added in the Table object must have a unique auto-incremental identifier to represent a sale/purchase. Additionally, the date of each sale/purchase must be recorded when it is registered.

- When a non-numeric value is entered in the designated section for entering the quantity to be processed, a tab or pop-up is displayed. In this particular case, it appears as a warning that the value entered does not meet the necessary requirements to carry out the respective addition or subtraction with the previous amount.



Non-Functional Requirements.

- The program must be able to operate smoothly and indicate to the user if the entered data is incorrect, guiding them to input valid data.

- The program is a desktop application, easily accessible.

- The program can be used on Linux, Windows, and Mac operating systems, provided the Java JDK is installed.

- The program can be used by only one client at a time, while running on a PC or virtual machine that meets the specified requirements above.

- The program should be easily manageable, with a simple interface to control and understand its elements in their entirety, avoiding confusion, especially for users with limited technological proficiency.

Conclusions.

1. The program starts with a predefined account value of 100, which is automatically updated and displayed in the graphical interface with each purchase or sale transaction.

2. The program includes specific buttons for making purchases and sales, recording each transaction in a table with unique identifiers and dates, and updating the total account value shown in the interface.

3. The program operates smoothly, provides feedback to the user regarding the validity of entered data, and can be used by only one client at a time on a PC or virtual machine with Java JDK installed.

Recommendations.

- Ensure that the total account value is prominently displayed in the graphical interface to facilitate transaction tracking.

- Implement real-time updates to instantly reflect changes after each transaction.

- Design purchase and sale buttons that are easily identifiable and accessible.

- Incorporate validations in the text fields to ensure that only valid numerical values are entered.

- Consider adding clear descriptions or labels to the input fields and buttons to enhance the user experience.

- Ensure that each transaction has a unique identifier and automatically record the date and time of the transaction.

- Allow for detailed viewing of transactions in the table, including all relevant fields such as identifier, value, and date.

- Implement clear and helpful error messages that guide the user to correct incorrect inputs.

- Optimize program performance to ensure smooth operation without noticeable delays.

- Conduct tests on different operating systems (Linux, Windows, and Mac) to ensure compatibility.

- Document the installation and configuration requirements for each operating system, with particular emphasis on the Java JDK version.

- Ensure that the program properly handles single-client execution, possibly by implementing mechanisms to prevent multiple instances.

- Consider implementing user sessions to maintain data integrity in case of unexpected program closures.

- Design a clean and simple user interface focused on ease of use and efficiency.

- Conduct usability tests with end users to identify areas for improvement in the interface and user interaction with the program.

Bibliography.

Diferencia entre requisitos funcionales y no funcionales en el desarrollo de software. (s/f). Cjavaperu.com. Recuperado el 25 de junio de 2024, de

https://cjavaperu.com/2021/09/diferencia-entre-requisitos-funcionales-y-no-funcionales-en-el-desarrollo-de-software/

Pop Up. (2022, febrero 10). iSocialWeb Agency. https://www.isocialweb.agency/wiki/pop-up/

Requerimientos no funcionales: Ejemplos. (s/f). Pmoinformatica.com. Recuperado el 25 de junio de 2024, de https://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html.

(S/f-a). Visuresolutions.com. Recuperado el 25 de junio de 2024, de https://visuresolutions.com/es/blog/functional-requirements/

(S/f-b). Edu.co. Recuperado el 25 de junio de 2024, de https://repositorio.konradlorenz.edu.co/micrositios/001-1527/requerimientos_no_funcionales.html.