

Practical Work 2 Documentation

GRAPHICAL CALCULATOR APPLICATION

FRANCISCO JAVIER DE SANTOS

Contents

Introduction..... 2

Description..... 3

 GUI Flow 3

 Form Classes description. 5

Problems..... 10

Conclusion..... 11

Introduction

Summary:

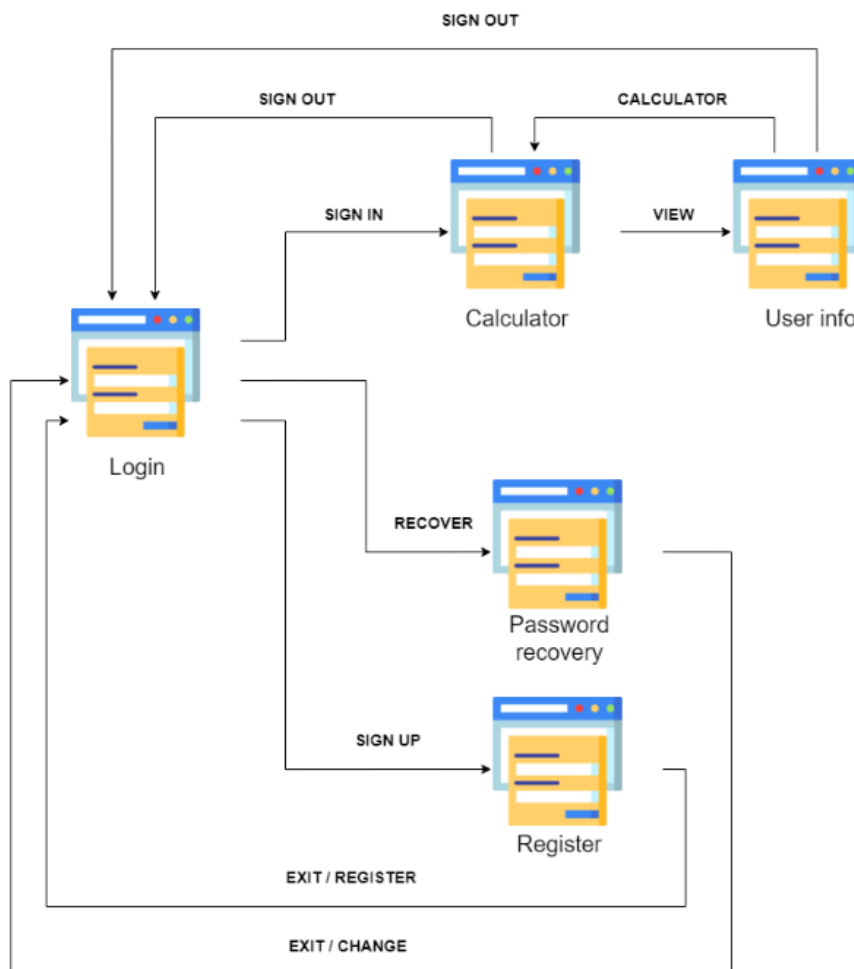
This document describes the second practice for the Object-Oriented Programming subject of Universidad Francisco de Vitoria. This document gives a detailed description of the project and the functionalities it provides. This includes a class diagram explaining a graphical representation of the classes used, encompassing their attributes and methods along with the interrelationships among the classes. I too explain the problems faced during the development of the solution to the exercise proposed and end this document with a brief conclusion about the project solution.

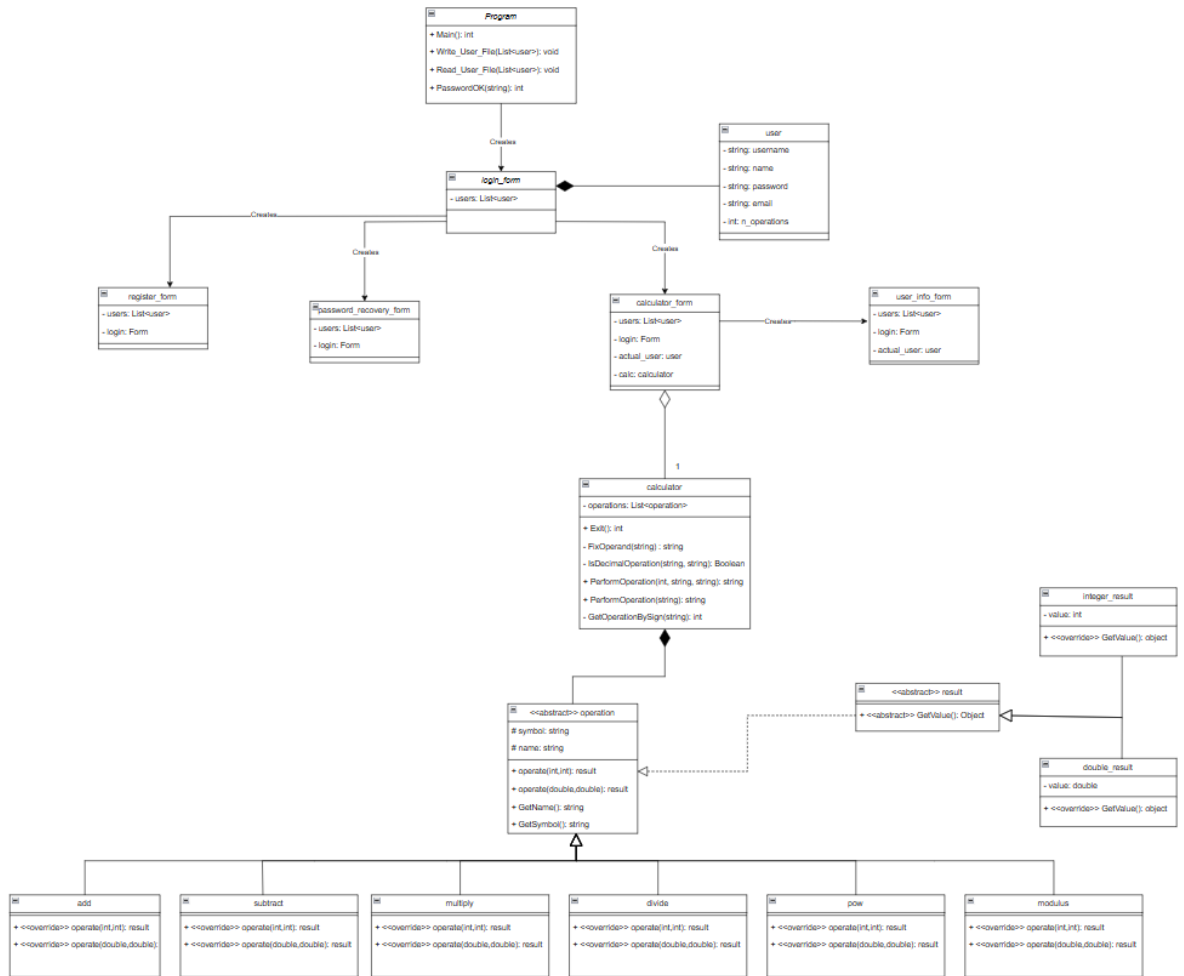
Description

GUI Flow

The GUI for this application follows the flow specified in the practical work statement.

Firstly, the user will start from a login. From the login the user will be able to register, recover the password if it was forgotten, or login to the calculator. From the registration form and password recovery from the user will be able to go back to the login. If the user logs in successfully, will be able to access the calculator form. From this form the user will be able to see the it's user information or go back to the login. If the user decides to see it's user information, will be able to go back to the calculator or go back to the login. In this process, the only form hidden is the first one, login form. This form will be passed through the classes and shown when necessary. The rest of the forms will be closed when the user opens other form.





Form Classes description.

Program class:

Our application starts in the Program class. Our program will host a list of users for them to have access to the calculator. The first thing the program does is add the users from the user_file.txt to the list for the application to handle the login. From here now the program is ready to run the application creating a login form.

This class will have too 3 functions that will help for the correct operation of the application. These functions are:

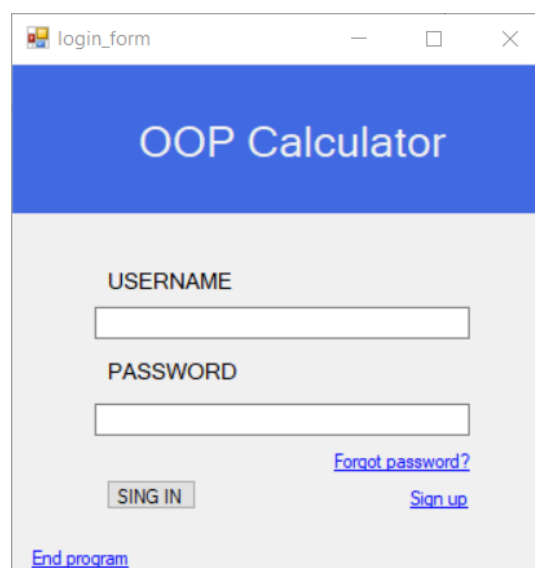
- Write_User_File: Writes users information into user_file.txt file.
- Read_User_File: Reads users information from user_file.txt and adds it to the list of users in the application.
- PasswordOK: Returns 1 if a given string contains all password requirements.

Login form class:

For the proper operation of the application, the login will have a list of users as an attribute. This list will be used for login purposes and will be sent through all the classes.

This login form contains two text boxes (one for username and other for password) for the user to be logged in into the calculator account. The login form contains, too, three text links, one for redirection to the password recovery form, other to a register form and the last one will close the application.

When the user wants to log in, just has to press the button SING IN to enter to its calculator account. If the username or password are incorrect the program will raise a Message box explaining the error.



The screenshot shows a window titled "login_form" with standard Windows window controls (minimize, maximize, close). The window has a blue header bar with the text "OOP Calculator" in white. Below the header, the background is light gray. There are two text input fields: the first is labeled "USERNAME" and the second is labeled "PASSWORD". To the right of the password field, there are two blue text links: "Forgot password?" and "Sign up". Below the input fields, there is a button labeled "SING IN" (note the typo in the image). At the bottom left of the window, there is a blue text link labeled "End program".

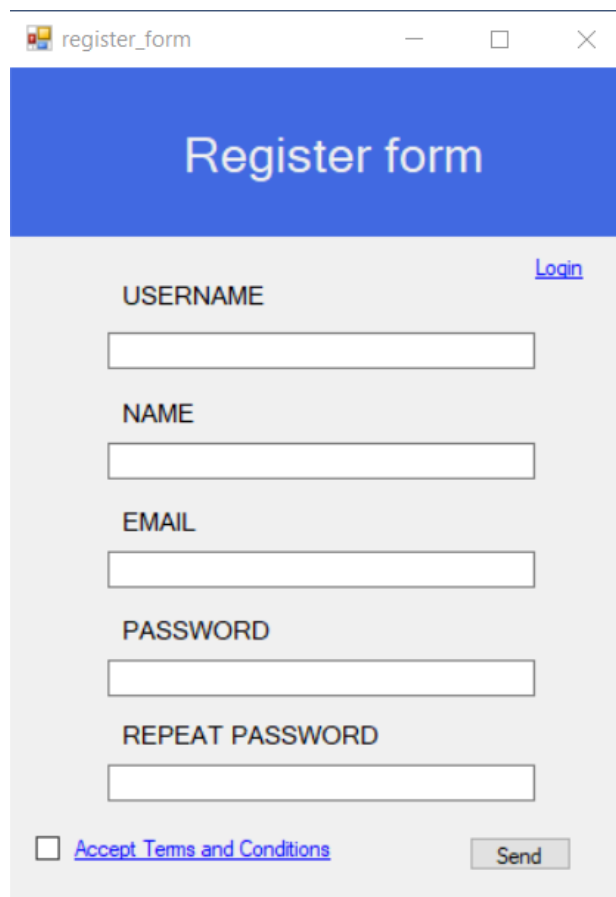
Register form class:

For the proper operation of the application, the register form class will have a list of users and a form as attributes. The first one will be used to confirm and add the new user, and the form will be the instance of the login of the program.

This form will contain five text boxes for the user to add username, name, email, password, and a password confirmation. The user will have to accept terms and conditions clicking on a checkbox and will be ready to send the information by clicking on the Send button. The main rules for users to be added are:

- Users must enter their name and username. Both must be different.
- Users must enter their password twice to confirm accuracy. Passwords must be at least 8 characters long, including one upper and one lower case letter, one number, and one special symbol.

Additionally, the user will be able to go back to the login form clicking on a login text link.



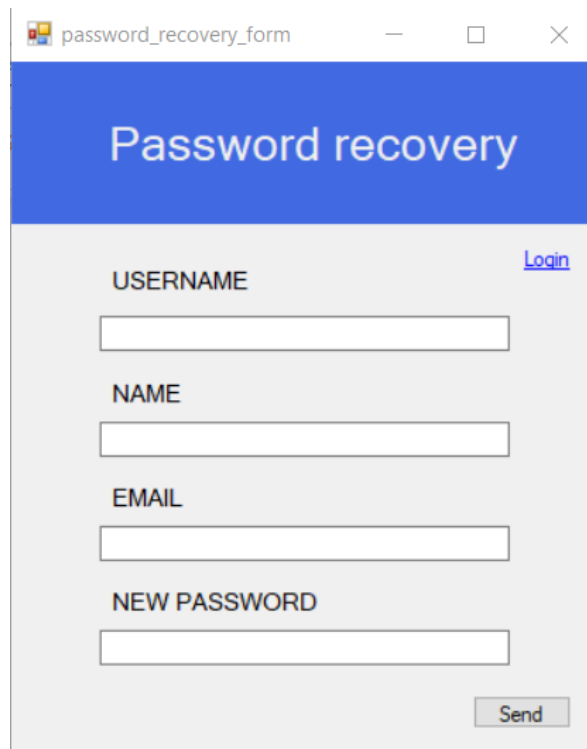
The image shows a web browser window with the title "register_form". The browser's address bar is empty. The page has a blue header with the text "Register form" in white. Below the header, there is a "Login" link in blue text. The main content area is light gray and contains five text input fields, each with a label above it: "USERNAME", "NAME", "EMAIL", "PASSWORD", and "REPEAT PASSWORD". At the bottom of the form, there is a checkbox labeled "Accept Terms and Conditions" and a "Send" button.

Password recovery form class:

For the proper operation of the application, the password recovery form class will have a list of users and a form as attributes. The first one will be used to confirm and add the new password for the user, and the form will be the instance of the login of the program.

This form contains four text boxes for username, name, email, and the new password. For the password to change the user will have to add the username, name, and email information correctly.

Additionally, the user will be able to go back to the login form clicking on a login text link.



The screenshot shows a web browser window with the title bar 'password_recovery_form'. The main content area has a blue header with the text 'Password recovery' in white. Below the header, the form is displayed on a light gray background. It contains four text input fields, each preceded by a label: 'USERNAME', 'NAME', 'EMAIL', and 'NEW PASSWORD'. To the right of the 'USERNAME' field, there is a blue text link labeled 'Login'. At the bottom right of the form, there is a gray button with the text 'Send'.

Calculator form class:

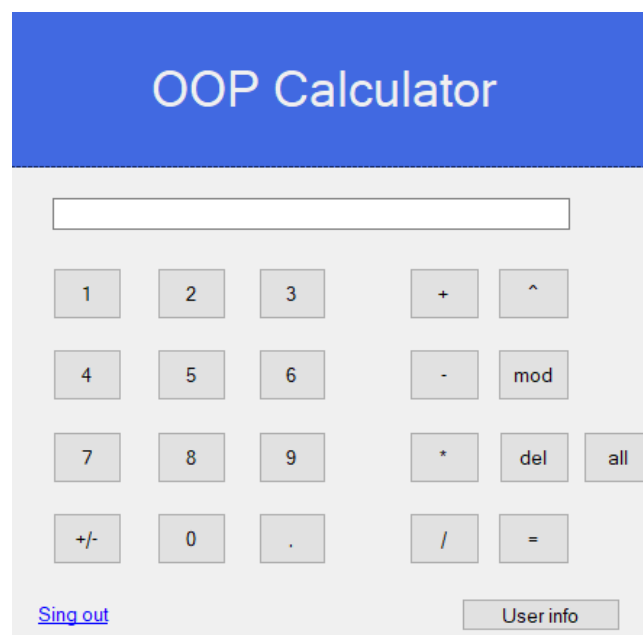
For the proper operation of the application, the calculator form class will have a list of users, a form, a user, and an instance of the calculator as attributes. The first one will be used to keep the user information in the program, the form will be the instance of the login of the program and the user will be the actual user logged in.

This form contains twenty-one buttons for operations, a text link to sing out, a text box to display the calculations and a user information button to display the user information.

In the left side of the form, we can see twelve buttons which are mostly numbers to perform operations. If the user clicks one of these buttons the number pressed will be displayed in the top textbox. There are two of the twelve left side buttons that are not numbers. One of those is the “,” button which displays a comma in the top text box for the numbers to have decimals. The other button “+/-” permits the user to change the sign of the number is formulating. If the user is writing the first operand, if “+/-” is pressed, that number will change it’s sign; if the user is already writing the second operand, that second operand will change sings.

The right side of the buttons contains the operators for calculations (add, subtract, multiply, divide, pow and modulus) and three extras for deletion and an equal sign to perform the operation. The deletion buttons “del” will delete the last character written, while the button “all” remove all the text in the text box.

When the user presses the equal sign, the string in the text box will be split in three parts (first operand, operator, second operand). If the string doesn’t have this format, the program will raise a message box showing an error to the user. If it does have the required format. we will make use of our Culinary Calculator developed in the course to take use each component in the string to calculate and display the result in the principal text box.



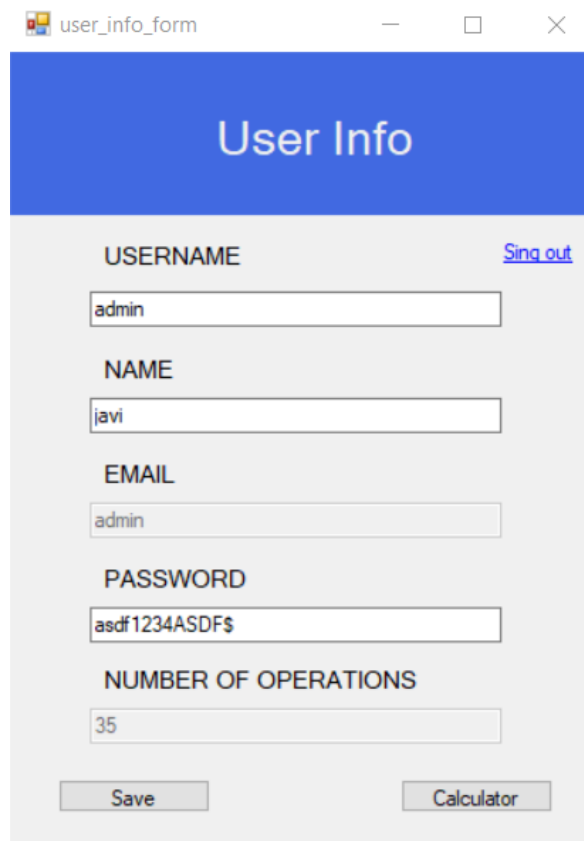
The image shows a screenshot of a web application titled "OOP Calculator". The interface features a blue header with the title. Below the header is a white text input field. The main area contains a grid of buttons. On the left, there are buttons for digits 1-9, 0, and a decimal point. On the right, there are buttons for arithmetic operators (+, -, *, /), a power button (^), a modulus button (mod), a sign change button (+/-), a delete button (del), and an equals button (=). At the bottom left, there is a blue link labeled "Sing out". At the bottom right, there is a button labeled "User info".

User information form class:

For the proper operation of the application, the user information form class will have a list of users, a form, and a user as attributes. The first one will be used to keep the user information in the program, the form will be the instance of the login of the program and the user will be the actual user logged in.

This form contains the 5 text boxes to display the user information, two buttons to go back to the calculator or save the information changed in the text boxes, and a text link to go back to sign out.

The user will be able to modify the username, name and password fields and , if they pass the rules mentioned above in the registration form, they could be changed and saved into the user_file.txt file.



The image shows a screenshot of a Java Swing window titled "user_info_form". The window has a blue header bar with the text "User Info" in white. Below the header, the form contains five text input fields, each with a label above it: "USERNAME" (containing "admin"), "NAME" (containing "javi"), "EMAIL" (containing "admin"), "PASSWORD" (containing "asdf1234ASDF\$"), and "NUMBER OF OPERATIONS" (containing "35"). To the right of the "USERNAME" field is a blue hyperlink labeled "Sing out". At the bottom of the form, there are two buttons: "Save" and "Calculator".

Problems

Initially, I encountered the challenge of the flow the forms should have **to close each form** before opening other one instead of hiding them. For this I had to investigate how to make it happen. The first approach I had was to close each form every time the user wanted to open other one. The problem is that the first form open, when is closed, it ends the program. To solve this problem, I decided to use the first form in all the program and hide it. To clarify, I decided to hide the first login form and pass it through argument through the program classes to be able to close the rest of forms and show the previous form instead of creating new instances.

Additionally, I first struggled to **merge the calculator solution** from made in the course. I didn't know how to create the input in the calculator form and use it as parameters to the calculator. Using the instructions from the practical work statement and researching in the last calculator solution to see how it was working I determined how to do it. I use the text box in the calculator form as the input string for the calculator. I just need to use the string from the text box and split it using spaces to know which part of the string is the first operand, which the operator and which is the second operand. This is possible because when the user adds an operator to the string a space is putted before and after the operator.

Finally, I encountered some problems with the **change of the sign** of the calculator. In the practical work statement, it wasn't really specified how it should work. I didn't really know if I had to change the sign of just one operator, or the sign of the result. For this, I had to gather a functional solution that should help users when calculating. I decided that the best way should be to give the users the opportunity to change the sign of the first operand or the second operand. With this idea in mind, I made possible when users are writing the first operand, to turn this number into negative, and when the user already pushed the operator button, to change the sign of the second operand.

With all this, I think that even with the difficulties explained in this section the work done was good and was able to pass through the difficulties creating a good solution for the practical work.

Conclusion

The project involved overcoming various challenges and learning valuable lessons along the way. The solution proved to be effective in addressing the requirements of the project, despite encountering several challenges along the way.

Here are some aspects where the effectiveness of the solution was demonstrated:

Clarity and understanding: I learned the importance of clarifying requirements and understanding the problem domain thoroughly before starting implementation. Ambiguities led to initial confusions but were eventually resolved through working on the functionality of the code.

Functionality: The solution successfully implemented the core functionalities required for a calculator, including adding, subtracting, multiplying, dividing, pow and modulus.

Error Handling: I incorporated error handling mechanisms to address various scenarios, such as incorrect user inputs, insufficient product availability, and authentication failures. By providing informative error messages, users were guided on how to proceed or correct their actions, enhancing the overall user experience.

Flexibility: Despite initially planning to follow a specific program execution diagram, I adapted the approach based on the evolving needs of the project. I made decisions that allowed for more flexible and practical implementations, such as creating deletion buttons.

Testing and Iterative Development: Through testing and iterative development, I was able to identify and address issues early in the development process. The challenges encountered, such as the issue with sign change, were promptly resolved through iterative improvements and adjustments to the code. I discovered the significance of iterative development and testing.

In conclusion, the effectiveness of the solution was evident in its ability to meet the project requirements, overcome challenges, and deliver a functional calculator. By prioritizing functionality, error handling, flexibility, simplicity, and testing, I successfully developed a solution that addressed the needs of the project while providing valuable insights for future projects. I will certainly apply the lessons learned from this project to future endeavours, understanding of what is asked, simplicity in design, and rigorous testing to ensure the success of other projects.