SOFE 4640U
Mobile Application Development

Assignment 1

Date: Oct 3, 2023

Saaruca Kugarajh (100751441)

Github Link:
https://github.com/SaarucaK/SOFE4640_Assignment1

Assignment 1 Premise

In this assignment, a mortgage calculator application was implemented, where the principal amount, interest rate and tenure is used to calculate the monthly payment installments. The user is expected to provide input for the principal amount, interest rate and tenure, and then press calculate to retrieve the resultant monthly payment installment. When they click the “Calculate” button, a new page will open which outlines the previously provided input, as well as displays the resultant payment output. A back button is also provided on this page to allow the user to navigate back to the calculator.

Main Activity - Mortgage Calculator

The Main Activity page consists primarily of 3 EditText features for user input and one button to submit and process the user input. 2 edit text features allow input of numbers with decimal values, which is used for the principal amount and the interest rate, as their units. The EditText feature used for the term only takes integer input, as partial years are not valid. Each of these EditText fields have a hint that says “Enter principal/interest rate/term…” for each respective input to prompt the user, which disappears per user compliance.

A TextView feature is used as a label for each EditText feature. The labels are used to provide the user information about what input is required in each field. It also provides general organization for the layout of the main page of the application by acting as a divider between the EditText fields. A second TextView feature is shown at the bottom of the page, which provides additional information on what input is required by the user. For example, it states that the Principal input is the mortgage amount. Finally a TextView feature at the top of the page acts as a title for the mortgage calculator.

The page title was first constrained to the top of the page, left and right side of the page. The width is adjusted to meet both ends of the page by setting the layout_width to “fill_parent”. The greater text size and darker background color were set to make the title stand out. The additional information at the bottom of the screen is similarly constrained, as it is constrained to the top, bottom , left and right of the screen. The width of this text box is also set to the “fill_parent” parameter to ensure that it spans the whole width of the screen.

The 3 EditText fields were constrained to the right sides of the screen. The left side constraint goes to the input EditText below. For example, the left side of the Principal is constrained to the left side of the Interest Rate EditText. The top constraint for these fields goes to the top of the screen. The bottom constraint goes to the TextField below it for the next user input label. For example, the Principal input field is bottom-constrained to the top of the Interest label, the Interest input is bottom-constrained to the top of the Term label, and the Term input is bottom-constrained to the top of the Calculate button. The constraint bottom is used to constrain the bottom of each TextView (ie label) to the top of their corresponding EditText (ie input) to align the label and the input box.

The calculator function is first launched by creating the using the onCreate method. The setContentView() method is provided with the activity id (activity_main). This will open the Main Activity (the calculator) on the screen. After the activity is created, views are used to retrieve the user input from their respective fields. The findViewById method retrieves the input using each of their corresponding ids and stores them in EditText variables. A button object is also created, where the calculation will only commence when it is clicked. The findViewById method is used to retrieve the button of the Calculate button previously created. This is done through using the View.OnClickListener, which listens to the button to check when it is submitted. When the button is first submitted, the program verifies if there is valid user input provided for each field through the confirmInput boolean method. The program only proceeds to the calculation if each input is at least one character long; otherwise an error message is displayed, informing the user that invalid input was provided.

To proceed to the calculation, the user input is first stored as an integer or decimal as accordingly through the getText().toString() methods, which retrieves the input at strings, and using parseInt or parseDouble to turn the string into a numerical value. After retrieving the numerical values, the mortgage payment is calculated using the EMI payment formula, and stored in the mortgage variable. This resultant value is also rounded to 2 decimal places, since it is a currency value. Then all the input values and the calculated values are converted into strings using getText().toString(). A Bundle object stores these 4 values as a local object. This is used to send variables/information between activities.

Then, an Intent object is created, where the view.getContext() is used to get the current activity, and MortgageResults.class is defined to navigate to that new activity. The method intent.putExtras(bundle) sends the string with the 4 stored values to the new activity. Finally, startActivity(intent) will open the new activity.

Mortgage Calculator Result

When the new activity Mortgage Results is open, it will display 2 sections; in input display and a result display. 2 TextViews act as headers for each section (titled Mortgage Details and My Payment Results), where the layout_width is set to fill_parent to ensure it spans the screen. The input is displayed in 3 rows using 3 TextViews, with an additional 3 TextViews acting as labels. The first row (ie 2 TextViews) are constrained to the top of the screen, while the tops of the 2nd and 3rd are constrained to the bottom of the TextViews above them to ensure equal spacing. The bottom of all the label text views are constrained to the neighboring column (ie the TextView holding the value) and vice versa, to ensure horizontal alignment. The left and right constraints are for all the variables are set the TextViews for the input directly above or below them to ensure vertical alignment. 2 space components are used to ensure even white space above and below the input, and are constrained to the input and result headers. The result TextView uses a constraint on the left and right to the input fields above it. Finally, a space component is utilized in order to provide even white space, and is constrained to the bottom, left and right of the screen.

This activity page is created using the onCreate method and setContentView() method which is provided the activity id (activity_mortgage_results). . Then a Bundle object is created using the getIntent().getExtras() method which refers to the main activity and the information transformed from it (the input and the resultant values). If the bundle contains the information (the calculator successfully obtained input and calculated and output), the information will proceed to be displayed on this screen. This is done through retrieving each individual string in the bundle using getString(), using findViewById to get the corresponding TextView on the screen, and using setText to display each input and the output on the screen. The top constraint goes to the result TextView that acts as a label (EMI - Equal Monthly Installments) to ensure vertical alignment. This label is aligned to the top, left and right of the screen.

The back button was enabled through enabling the setDisplayHomeAsUpEnabled() method. After setting this to true, the parent activity is set to the .MainActivity in the Android Manifest xml file in order to provide functionality to the back button. This allows users to navigate back to the calculated screen.

Reference

https://www.businessinsider.com/personal-finance/how-to-calculate-mortgage-payment