**Home Affordability Calculator**

1. **Is the analysis completed to start the coding activity?**

Yes

1. **What is the approach and flow chart to start the coding activity?**

This program calculates the affordability of a home for a family based on their incomes and expenses with defining necessary Variables. The program prompts the user to enter the family name, incomes, and expenses and then calculates the total income, total expense, and affordability. The program also writes the input and output to a file.

**FLOWCHART**

Teams

Description automatically generated with medium confidence

**3.Are all the scenarios documented?**

Yes, all the scenarios are documented.

**4.Are the pseudo code algorithms ready to finalize a skeletal structure for the code to be attempted?**

1)HomeAffordablityImpl class  
    Import necessary Java libraries  
  
2)  homeAffordablityInput method  
        - Create a Scanner object for user input  
        - Initialize a PrintWriter object named "fileWriter" to null  
  
3)       try-catch block  
            - Inside the try block:  
                - Create a new PrintWriter object named "fileWriter" that writes to "history.txt"  
  
                - Prompt the user to enter their family name and store the input in a variable named "familyName"  
                - While the family name is an empty string, keep prompting the user to enter a valid family name  
                - Create a new Family object with the given family name  
                - Write the current date and time and the family name to the file  
  
                - Display the message "Enter incomes (type, amount):" to the user  
                - Display the valid income types (Salary, Business, Freelancing, Bonus, Commission, Other)  
  
                - Set a boolean variable named "moreIncomes" to true  
                - While moreIncomes is true:  
                    - Prompt the user to enter an income type and store the input in a variable named "incomeType"  
                    - If the income type is not a valid income type, display an error message and continue to the next iteration of the loop  
                    - Prompt the user to enter an income amount and store the input in a variable named "incomeAmount"  
                    - If the income amount is not a valid number, display an error message and continue to the next iteration of the loop  
                    - If the income amount is negative, display an error message and continue to the next iteration of the loop  
                    - Add a new Income object with the given income type and amount to the family object  
                    - Prompt the user to enter 'y' or 'n' to indicate whether they want to add another income  
                    - If the user does not enter 'y' or 'n', display an error message and continue to the next iteration of the loop  
                    - If the user enters 'n', set moreIncomes to false  
  
4)   Write the income data to the file  
  
                - Display the message "Enter expenses (name, amount):" to the user  
                - Display the valid expense types (Mortgage, Utilities, Property tax, Insurance, Food, Other)  
  
                - Set a boolean variable named "moreExpenses" to true  
                - While moreExpenses is true:  
                    - Prompt the user to enter an expense type and store the input in a variable named "expenseType"  
                    - If the expense type is not a valid expense type, display an error message and continue to the next iteration of the loop  
                    - Prompt the user to enter an expense amount and store the input in a variable named "expenseAmount"  
                    - If the expense amount is not a valid number, display an error message and continue to the next iteration of the loop  
                    - If the expense amount is negative, skip to the next iteration of the loop  
                    - Add a new Expense object with the given expense type and amount to the family object  
                    - Prompt the user to enter 'y' or 'n' to indicate whether they want to add another expense  
                    - If the user does not enter 'y' or 'n', display an error message and continue to the next iteration of the loop  
                    - If the user enters 'n', set moreExpenses to false  
  
5)              - Calculate the total income and total expense using the IncomeCalculator and ExpenseCalculator classes  
                - If the total expense is greater than the total income, display an error message  
                -else  
                      Calculate the difference between total income and total expenses  
                       difference = total\_income - total\_expenses  
6)               Print the total difference (affordability)

7)  print the affordability price and tell is it suitable or not  
8)   stop

**5. What are the Java Classes being written & what are they composed of?**

HomeAffordability.java - This is the main class that represents that prompts the user input and calculate affordability.

Family.java: This class that represents a family and contains a name and list of income and expenses.

Income.java: This class that represents an income source and contains type and amount.

Expense.java: This class that represents an expense and contains name and amount.

IncomeCalculator.java: This class that calculates the total income from the list of incomes.

ExpenseCalculator.java: This class that calculates the total expense from the list of expense

**6.Any Interfaces are being used? Did you figure out a need of a functional interface?**

He used one functional interface named "Calculable" with a single method "calculateTotal()" that returns a double.

**7. What are the conditional statements, logical loops & ternary operators to be used?**

If-else condition,

While loops,

Switch statements

Advanced for loops.

**8. What are the Collection classes & Stream API classes to be used?**

In this code, the Collection classes used are “ArrayList “for both incomes and expenses list. The Stream API classes used are Stream, “mapToDouble”, and sum. The” stream() “method is used to convert the List of Income and Expense objects to a Stream, mapToDouble() method is used to convert each Income and Expense object to a double value, and sum() method is used to add up all the values obtained from mapToDouble() method.

**9. Exception Handling to be applied - Custom or pre-written?**

When parsing a string input to a double, the code uses a try-catch block to catch the NumberFormatException that may occur if the input is not a valid number and wanted to use the Decimalformat it is showing in decimal format the should be in number format.

**10. Is garbage collection properly applied? If yes, how**

No.