**Lab Task 4.1**

**[using Arrays]**

Refer to the BankAccount class created in Lab 2. The class actually contained the following.

* + Data
    - Title name
    - Balance
    - A static field **transactionCountLimit**
  + Constructors
    - One parameterized constructor for name, assign default value of 500 to balance.
    - One parameterized constructor for both name and balance
  + Methods
    - A method to print balance
    - To deposit given amount into balance
    - To withdraw given amount from balance

1. Now enhance the BankAccount class, so that it contains a list (**transactionAmount**) of 100 previous transactions amount (negative for credit, positive for debit)
2. The class **constructors** now also create an empty Array.
3. Update the **doposit/withdraw methods** so that they add to the transactionAmount list. If the transactions increase beyond **transactionCountLimit**, then no deposit/withdraw should be allowed. (Hint: you have to keep track of number of transactions performed so far.)
4. Add a method **computeTransactionBalance** (which uses enhanced for loop) to compute and return the balance transactions amount.
5. Add methods **printCreditTransactions,** which prints only credit transactions**.**
6. Implement the **toString** method, which returns the following string:

Mr. Ali has 5000 Rs. (Transcations count: 3)

1. Write test class for Account class
   1. Create an **account object** **using** **parameterized constructor** by passing appropriate values.
   2. Use a **sentinel controlled loop** to ask whether user wants to perform debit or credit (-1 or +1). The loop ends when the user enters zero ‘0’.
      1. Input the amount from user and perform appropriate transaction according to user choice previously entered.
   3. When the loop ends, then
      1. Call **computeTransactionBalance** andprint the transaction balance**.**

**Lab Task 4.2**

**[usage of ArrayList class with wrapper classes]**

Refer to the BankAccount class created in Lab 2. The class actually contained the following.

* + Data
    - Title name
    - balance
  + Constructors
    - One parameterized constructor for name, assign default value of 500 to balance.
    - One parameterized constructor for both name and balance
  + Methods
    - A method to print balance
    - To deposit given amount into balance
    - To withdraw given amount from balance

1. Now enhance the BankAccount class, so that it contains a list (**transactionAmounts**) of 5 previous transactions amount (negative for credit, positive for debit). **Hint:** use ArrayList of Double Wrapper class.
2. The class **constructors** now also create an empty ArrayList.
3. Update the **doposit/withdraw methods** so that they add to the transactionsAmount list, however, only latest 5 transactions can be saved. (**Hint:** if transaction limit exceeds 5 transactions, use the remove method to delete first transaction amount, and add method to add amount at the end. Otherwise, simply add the amount at the end.)
4. Add a method **computeTransactionBalance** (which uses enhanced for loop) to compute and return the balance transaction amount for five previous transactions.
5. Add methods **printCreditTransactions,** which prints only credit transactions among the last 5 transactions**.**
6. The test class should be the same as Task 4.1.

**Lab Task 4.3:**

Text

Description automatically generated

[optional]: Create the following additional functions in the polynomial class:

* + - * addPolynomial: adds 2 polynomials. Create 2 overloaded methods:
        + void addPolynomial(Polynomial p2);

adds polynomial p2 to the current polynomial object.

* + - * + static Polynomial addPolynomials(Polynomial p1, Polynomial p2);

a static class method which adds 2 polynomials p1 and p2 and returns the resultant polynomial. Note that you should create a deep copy of the result polynomial.

* + - * multiplyPolynomials: multiplies 2 polynomials
        + add 2 overloaded versions of multiplyPolynomials in the same way as for addPolynomials.

**Home Tasks**

**Task 1.**

Lab Task 4.3 used an *array* of coefficients to implement a polynomial class. Complete the same task using an *ArrayList* of coefficients.

**Task 2.**

Graphical user interface, text

Description automatically generated

Text, letter

Description automatically generated

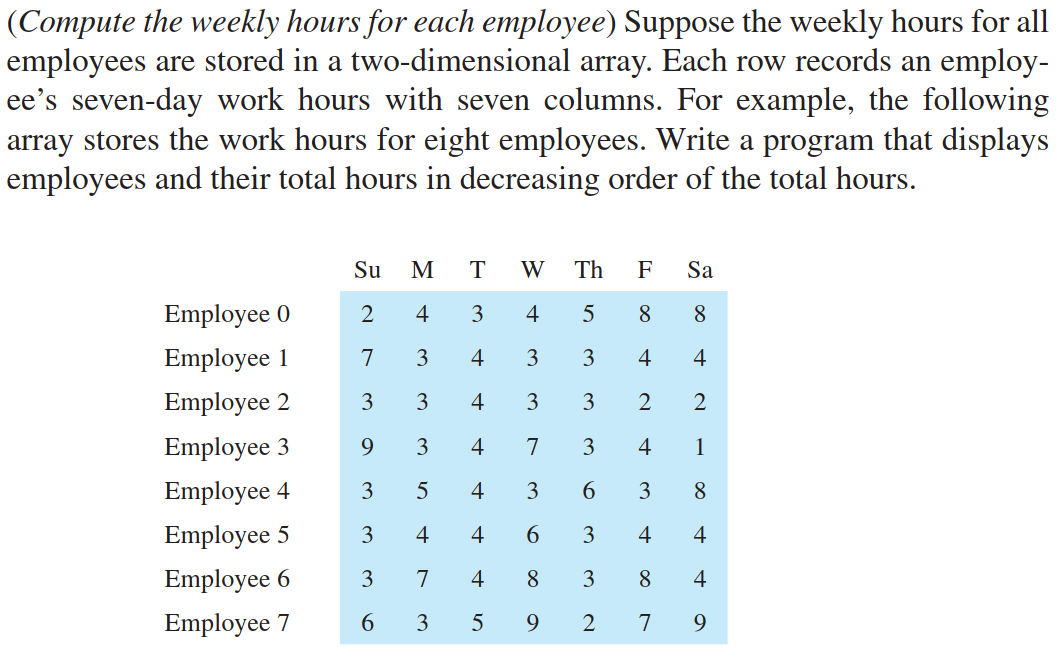
Text

Description automatically generated with low confidence

**Note:** Printing matrices horizontally as in above sample output can be a bit challenging. Therefore, first try printing it vertically, and then try horizontal printing. A template for vertical printing is as follows:

**Matrix 1**+   
**Matrix 2**=  
**Matrix 3**

**Task 3.**



[**Hint**: Assume “Employee 0” … “Employee 7” are employee names. use a sorting algorithm (like bubble sort) to sort the total hours. But, the original position (index) of each sorted element should also be saved. ]