Object-Oriented Programming Lab#4

Today's Topics

- Class and Object
- Initialization of fields.
- Constructor
- String concatenation
- Array of reference type

Problems/Assignments

Problem#1

Create a Banking System, where a user can **create** a new account, **deposit** money to a specific account, **withdraw** money from a specific account and **check** the balance of a specific account. Each Account is identified by its **account number, balance, and the name** of the account holder. The system should be able to handle multiple accounts.

What you need to do:

1) Create a **BankAccount** class which has **3 instance variables**; ***name, accNum*** and ***balance***.

   a. Create a ***constructor*** with 3 arguments and initialize the attributes.

   The class also has the **following 4 methods**;

   b. void deposit(double depAmount)

   - Inside the method the ***balance*** variable needs to be increased by the "***depAmount***" amount.

   c. void withdraw(double withAmount)

   - The ***balance*** is decreased by "***withAmount***" amount. We have to make sure the ***balance*** does not become negative.

   d. double getBalance()

   - The method returns the ***balance***.

   e. void display()

   - This method displays the attributes in the format "Name:[name]; AccNum:[accNum]; Balance:[balance]". Use ***toString***() method to get the formatted string.

2) Now create an **application class** (that has the main method) named "**Bank**" which will have the **main** method.

   - In the main method, create an **array/ArrayList** of **BankAccount** type. If you choose array, set the array size to 10. Name the **array/ArrayList** variable as **accounts**. After creating the array provide the following **menu** on the console and take appropriate action.

     - '1' to create new account.

- So, for this option, take input for the 3 fields (name, accNum, balance) from the user. After taking the input, create a ***BankAccount*** object add the object to **accounts** array.

- '2' to **deposit** money.
    - For this option, you have to ask user for the account number of the account s/he wants to deposit and amount of money s/he wants to deposit. After taking the input do the following.
        - **Find** the account from the Array
        - If the account is available in the list**, call** the **deposit** method for that object with appropriate parameter.

- '3' to **withdraw** money.
    - For this option, you have to ask user for the account number of the account s/he wants to deposit and amount of money s/he wants to deposit. After taking the input do the following.
        - **Find** the account from the Array
        - If the account is available in the list**, call** the **withdraw** method for that object with appropriate parameter.

- '4' to **display** the **balance** of a specific account.
    - For this option, you have to ask user for the account number of the account s/he wants to check the balance. After taking the input do the following.
        - **Find** the account from the Array
        - If the account is available in the list**, call** the **getBalalnce()** method for that object and print the output.

- '5' to **display** the **details** of a specific account.
    - For this option, you have to ask user for the account number of the account s/he wants to view the details. After taking the input do the following.
        - **Find** the account from the Array
        - If the account is available in the list**, call** the **display()** method for that object.

- '6' to **display** the **details** of a all accounts.
    - For this option, do the following.
        - **Access** each of the account object from the Array
        - **Call** the **display()** method for each object.
- '0' to **exit** the system.
    - Come out of the loop if user chooses this option.

Problem#2:

Update the **Online Store of Lab#3/4** to handle **multiple products**, where user can do the following

- Add new product to the system
- Update the price of a product
- Get the discounted price of a product
- View the list of products with their details.

Problem#3:

Update the **Employee Management System of Lab#3/4** to handle **multiple employees**, where user can do the following

- Add new employee to the system
- Update the salary of an Employee
- Get the salary of an Employee
- View the list of employees with their details.