Batch:B4 **Roll No.: 16010122221**
Experiment / assignment / tutorial No. 04
Grade: AA / AB / BB / BC / CC / CD /DD
Signature of the Staff In-charge with date

TITLE :An Array of Objects

AIM: Write a program which accepts information about n no of customers from user .Create an array of objects to store account_id ,name,balance.
Your program should provide following functionalities

1. To add account
2. To delete any account detail
3. To display account details.

Expected OUTCOME of Experiment:

CO1: Understand the features of object oriented programming compared with procedural approach with C++ and Java

CO2: Explore arrays, vectors, classes and objects in C++ and Java.

Books/ Journals/ Websites referred:

1. Ralph Bravaco , Shai Simoson , “Java Programing From the Group Up” Tata McGraw-Hill.
2. Grady Booch, Object Oriented Analysis and Design .

Pre Lab/ Prior Concepts:

Arrays of Objects:

Unlike traditional array which store values like string, integer, boolean, etc. array of objects stores objects. The array elements store the location of reference variables of the object.

For example:

```
class Student {
   int rno;
   String name;
   float avg;
}
Student(int r, String name, float average)
{
     rno=r;
    this.name=name;
     avg=average;
}

Student studentArray[] = new Student[n];
```

● The above statement creates the array which can hold references to n number of Student objects. It doesn't create the Student objects themselves. They have to be created separately using the constructor of the Student class. The studentArray contains n number of memory spaces in which the address of n Student objects may be stored.

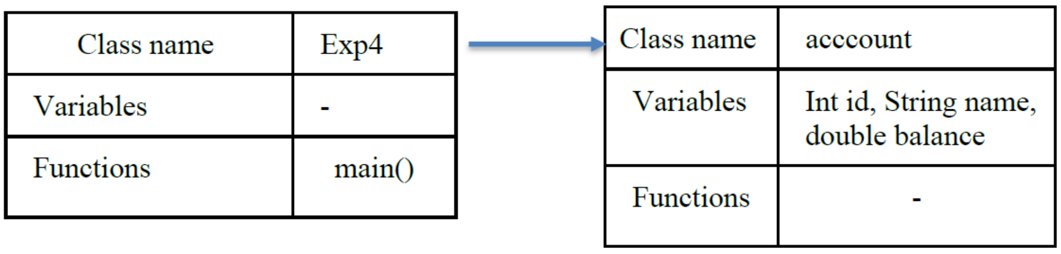
```
for ( int i=0; i<studentArray.length; i++) {
studentArray[i]=new Student(r,name,average);
}
```

● The above for loop creates n Student objects and assigns their reference to the array elements. Now, a statement like the following would be valid.

```
studentArray[i].r=1001;
```

.

Class Diagram:



Class name	Exp4
Variables	-
Functions	main()

Class name	acccount
Variables	Int id, String name, double balance
Functions	-

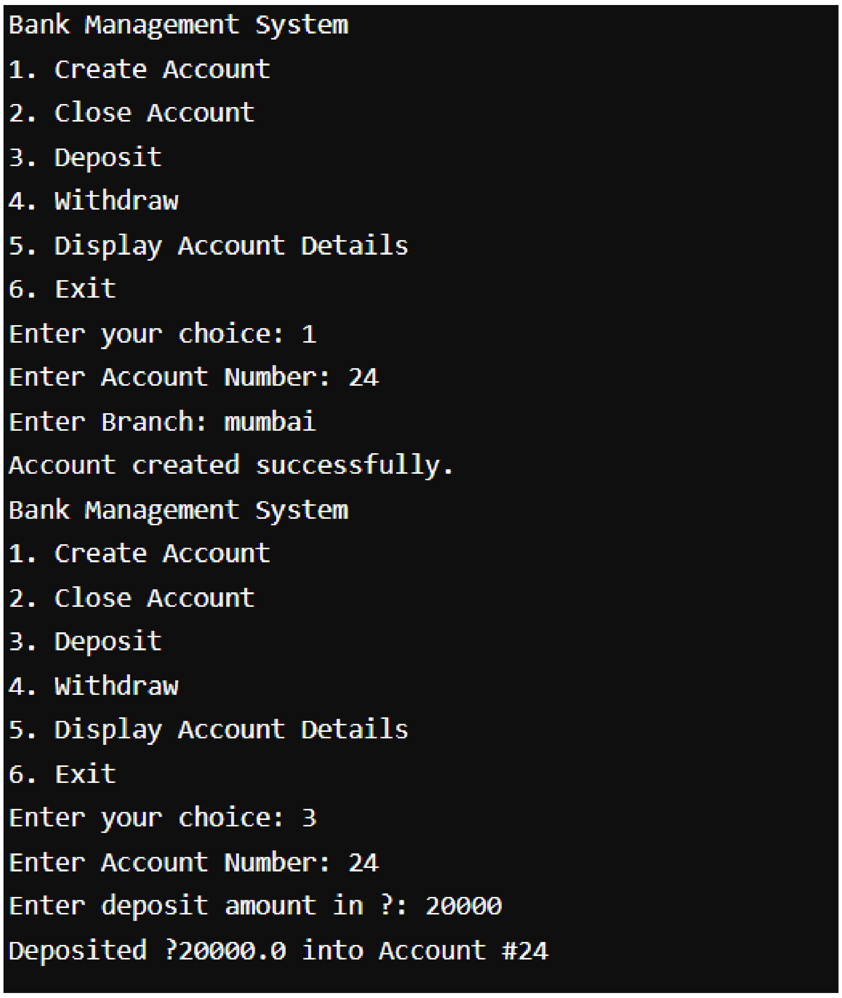
Algorithm:

1. Create a class Account with attributes int is, String name and float Balance.
2. Create a constructor for this class.
3. Create a public class Expt_4.
4. Define the main method in this class.
5. Create an object of Scanner class.
6. Get the total number of customers from the user and declare the array of same length.
7. Define a while loop.
8. In this while loop get choice from the user to Add, Delete, Display the account or to Exit.
9. If user selects option 1.
10. Get the number of accounts to be added initially.
11. Get account number, account holder name and balance.
12. Store this in the array of objects defines earlier.
13. If user selects option 2.
14. Get the account number from the user.
15. Find that account number in the array and shift the next element of the array at that position.
16. If user selects option 3.
17. Print the contents of array using for loop.
18. If user selects option 4.
19. Exit

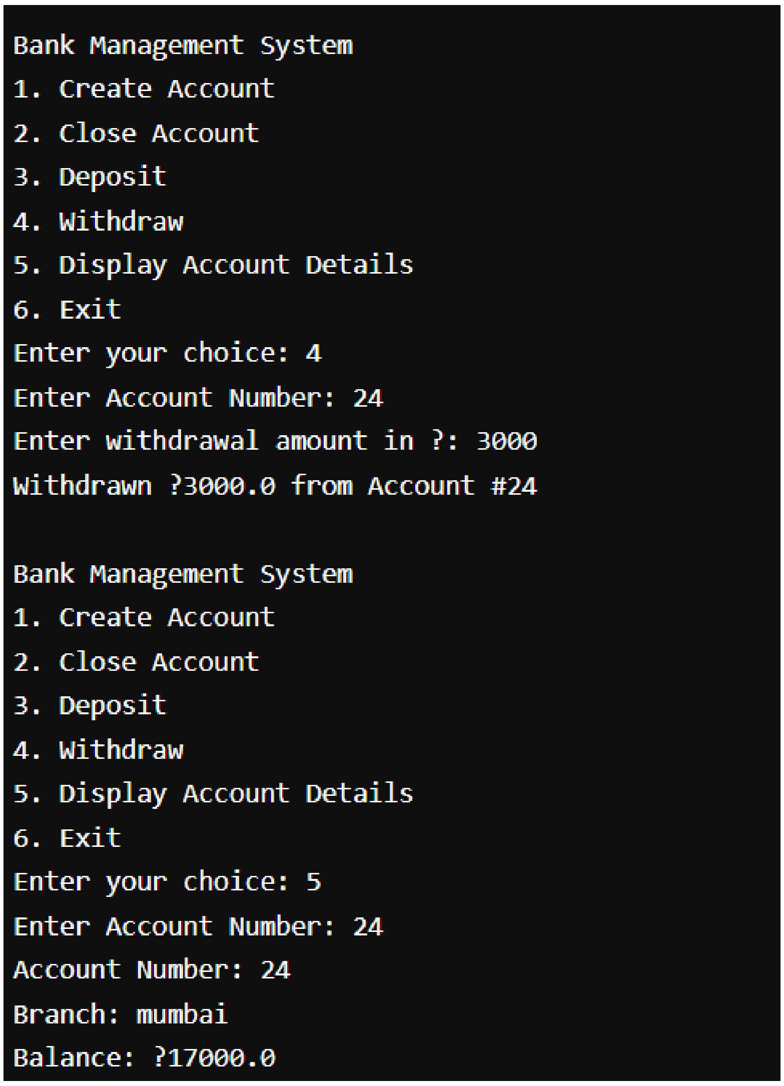
Implementation details:

```java
import java.util.ArrayList;
import java.util.Scanner;

class Customer {
    private int id;
    private String name;
    private String address;

    public Customer(int id, String name, String address) {
        this.id = id;
        this.name = name;
        this.address = address;
    }

    public int getId() {
        return id;
    }

    @Override
    public String toString() {
        return "Customer ID: " + id + ", Name: " + name + ", Address: " +
address;
    }
}

class Account {
    private int accountNumber;
    private double balance;
    private String branch;
    private ArrayList<Customer> accountHolders;

    public Account(int accountNumber, String branch) {
        this.accountNumber = accountNumber;
        this.branch = branch;
        this.balance = 0.0;
        this.accountHolders = new ArrayList<>();
    }

    public int getAccountNumber() {
        return accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public String getBranch() {
        return branch;
    }

    public ArrayList<Customer> getAccountHolders() {
        return accountHolders;
    }

    public void addAccountHolder(Customer customer) {
        accountHolders.add(customer);
    }

    public void removeAccountHolder(Customer customer) {
        accountHolders.remove(customer);
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited ₹" + amount + " into Account #"
+ accountNumber);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn ₹" + amount + " from Account #"
+ accountNumber);
        } else {
            System.out.println("Insufficient funds or invalid withdrawal
amount.");
        }
    }

    public void displayAccountDetails() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Branch: " + branch);
        System.out.println("Balance: ₹" + balance);
        System.out.println("Account Holders:");
        for (Customer customer : accountHolders) {
            System.out.println(customer);
        }
    }
}

public class BankManagementSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ArrayList<Account> accounts = new ArrayList<>();

        while (true) {
            System.out.println("\nBank Management System");
            System.out.println("1. Create Account");
            System.out.println("2. Close Account");
            System.out.println("3. Deposit");
            System.out.println("4. Withdraw");
            System.out.println("5. Display Account Details");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    createAccount(scanner, accounts);
                    break;
                case 2:
                    closeAccount(scanner, accounts);
                    break;
                case 3:
                    deposit(scanner, accounts);
                    break;
                case 4:
                    withdraw(scanner, accounts);
                    break;
                case 5:
                    displayAccountDetails(scanner, accounts);
                    break;
                case 6:
                    scanner.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Please enter a
valid option.");
            }
        }
    }

    private static void createAccount(Scanner scanner, ArrayList<Account>
accounts) {
        System.out.print("Enter Account Number: ");
        int accountNumber = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        System.out.print("Enter Branch: ");
        String branch = scanner.nextLine();

        Account account = new Account(accountNumber, branch);
        accounts.add(account);

        System.out.println("Account created successfully.");
    }

    private static void closeAccount(Scanner scanner, ArrayList<Account>
accounts) {
        System.out.print("Enter Account Number to close: ");
        int accountNumber = scanner.nextInt();

        Account accountToRemove = null;
        for (Account account : accounts) {
            if (account.getAccountNumber() == accountNumber) {
                accountToRemove = account;
                break;
            }
        }

        if (accountToRemove != null) {
            accounts.remove(accountToRemove);
            System.out.println("Account #" + accountNumber + " closed
successfully.");
        } else {
            System.out.println("Account not found.");
        }
    }

    private static void deposit(Scanner scanner, ArrayList<Account>
accounts) {
        System.out.print("Enter Account Number: ");
        int accountNumber = scanner.nextInt();

        Account accountToDeposit = findAccountByNumber(accounts,
accountNumber);
        if (accountToDeposit != null) {
            System.out.print("Enter deposit amount in ₹: ");
            double amount = scanner.nextDouble();
            accountToDeposit.deposit(amount);
        } else {
            System.out.println("Account not found.");
        }
    }

    private static void withdraw(Scanner scanner, ArrayList<Account>
accounts) {
        System.out.print("Enter Account Number: ");
        int accountNumber = scanner.nextInt();

        Account accountToWithdraw = findAccountByNumber(accounts,
accountNumber);
        if (accountToWithdraw != null) {
            System.out.print("Enter withdrawal amount in ₹: ");
            double amount = scanner.nextDouble();
            accountToWithdraw.withdraw(amount);
        } else {
            System.out.println("Account not found.");
        }
    }

    private static void displayAccountDetails(Scanner scanner,
ArrayList<Account> accounts) {
        System.out.print("Enter Account Number: ");
        int accountNumber = scanner.nextInt();

        Account accountToDisplay = findAccountByNumber(accounts,
accountNumber);
        if (accountToDisplay != null) {
            accountToDisplay.displayAccountDetails();
        } else {
            System.out.println("Account not found.");
        }
    }

        private static Account findAccountByNumber(ArrayList<Account>
accounts, int accountNumber) {
        for (Account account : accounts) {
            if (account.getAccountNumber() == accountNumber) {
                return account;
            }
        }
        return null;
    }
}
```

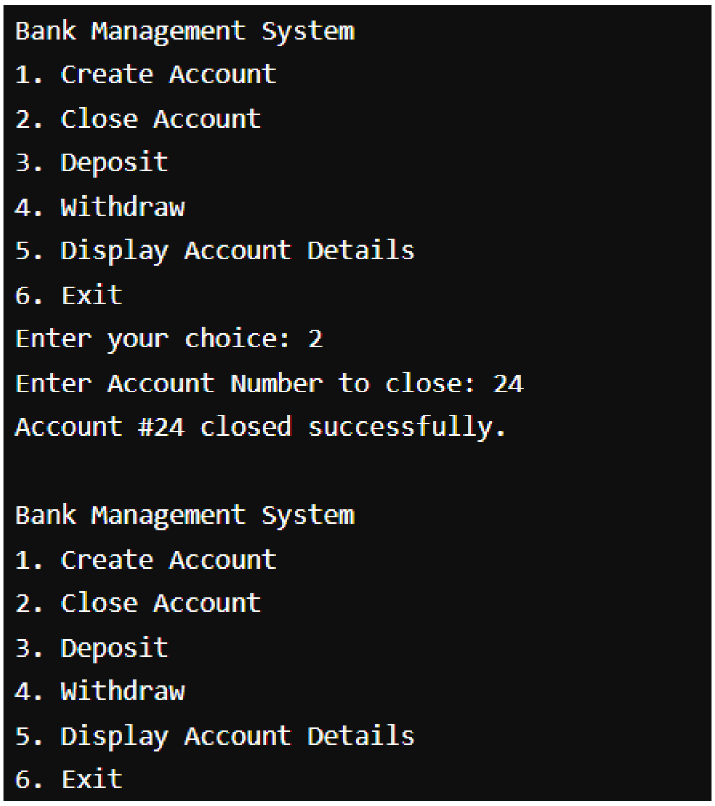
Output:



```
Bank Management System
1. Create Account
2. Close Account
3. Deposit
4. Withdraw
5. Display Account Details
6. Exit
Enter your choice: 1
Enter Account Number: 24
Enter Branch: mumbai
Account created successfully.
Bank Management System
1. Create Account
2. Close Account
3. Deposit
4. Withdraw
5. Display Account Details
6. Exit
Enter your choice: 3
Enter Account Number: 24
Enter deposit amount in ?: 20000
Deposited ?20000.0 into Account #24
```



```
Bank Management System
1. Create Account
2. Close Account
3. Deposit
4. Withdraw
5. Display Account Details
6. Exit
Enter your choice: 4
Enter Account Number: 24
Enter withdrawal amount in ?: 3000
Withdrawn ?3000.0 from Account #24

Bank Management System
1. Create Account
2. Close Account
3. Deposit
4. Withdraw
5. Display Account Details
6. Exit
Enter your choice: 5
Enter Account Number: 24
Account Number: 24
Branch: mumbai
Balance: ?17000.0
```


```
Bank Management System
1. Create Account
2. Close Account
3. Deposit
4. Withdraw
5. Display Account Details
6. Exit
Enter your choice: 2
Enter Account Number to close: 24
Account #24 closed successfully.

Bank Management System
1. Create Account
2. Close Account
3. Deposit
4. Withdraw
5. Display Account Details
6. Exit
```

Conclusion:

The provided Java code for a Bank Management System allows users to create and manage bank accounts. It includes features like creating accounts, closing accounts, depositing, and withdrawing funds, and displaying account details.

Date: _____________ **Signature of faculty in-charge**

<u>Post Lab Descriptive Questions</u>

Q.1 If an array of objects is of size 10 and a data value have to be retrieved from 5th object then ________________ syntax should be used.

a)Array_Name[4].data_variable_name;
b)Data_Type Array_Name[4].data_variable_name;
c)Array_Name[4].data_variable_name.value;
d) Array_Name[4].data_variable_name(value);
Ans: a

Q.2)The Object array is created in _____________________
a)Heap memory
b) Stack memory
c) HDD
d) ROM
Ans: a