**Batch: C3 Roll No.: 16010123217**

**Experiment / assignment / tutorial No. 5**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE : An Array of Objects** |

**AIM:** Create classes Customer, Account and AccountManagement. Your program should provide following menu:

1. To open an account

2. To close an account

3. To display account details.

All the classes should have suitable data members and member functions.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

CO1: Apply the features of object oriented programming languages. (C++ and Java)

CO2:Explore arrays, vectors, classes and objects in C++ and Java **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

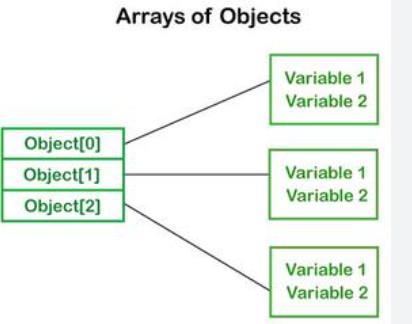
**Books/ Journals/ Websites referred:**

1. E. Balagurusamy, “Programming with Java”, McGraw-Hill.
2. E. Balagurusamy, “Object Oriented Programming with C++”, McGraw-Hill.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Java is an object-oriented programming language. Most of the work done with the help of objects. We know that an array is a collection of the same data type that dynamically creates objects and can have elements of primitive types. Java allows us to store objects in an array. In [Java](https://www.javatpoint.com/java-tutorial), the class is also a user-defined data type. An array that conations class type elements are known as an array of objects. It stores the reference variable of the object.



## Creating an Array of Objects

Before creating an array of objects, we must create an instance of the class by using the new keyword. We can use any of the following statements to create an array of objects.

**Syntax:**

ClassName obj[]=new ClassName[array\_length]; //declare and instantiate an array of objects

**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++)

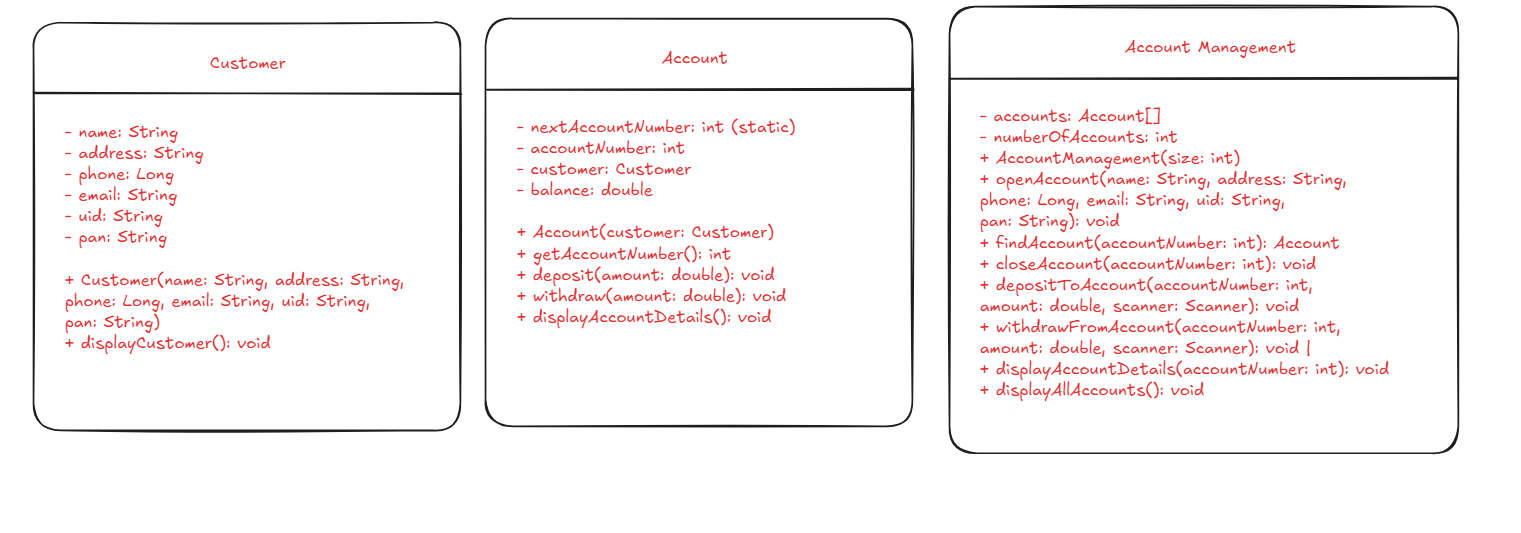
{  
studentArray[i]=new Student(r,name,average);  
}

* 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:**



**Algorithm:**

Initialization

Create a Scanner object for input.

Create an instance of AccountManagement with a fixed size (10 accounts).

Main Loop

Print the menu options.

Read the user's choice.

User Choice Handling

Choice 1: Open an Account

Prompt the user for customer details (name, address, phone, email, UID, PAN).

Call openAccount method of AccountManagement with the provided details.

Choice 2: Close an Account

Prompt the user for the account number to close.

Call closeAccount method of AccountManagement with the provided account number.

Choice 3: Display Account Details

Prompt the user for the account number.

Call displayAccountDetails method of AccountManagement with the provided account number.

Choice 4: Display All Accounts

Call displayAllAccounts method of AccountManagement to show all accounts.

Choice 5: Deposit

Prompt the user for the account number and deposit amount.

Call depositToAccount method of AccountManagement with the provided details.

Choice 6: Withdraw

Prompt the user for the account number and withdrawal amount.

Call withdrawFromAccount method of AccountManagement with the provided details.

Choice 7: Exit

Print a message indicating the program is exiting.

Close the Scanner and terminate the program.

Default Case: Invalid Choice

Print a message indicating an invalid choice and prompt the user to try again.

**Implementation details:**

import java.util.Scanner;

class Customer {

    private String name;

    private String address;

    private Long phone;

    private String email;

    private String uid;

    private String pan;

    public Customer(String name, String address, Long phone, String email, String uid, String pan) {

*this*.name = name;

*this*.address = address;

*this*.phone = phone;

*this*.email = email;

*this*.uid = uid;

*this*.pan = pan;

    }

    public void displayCustomer() {

        System.out.println("Name: " + name);

        System.out.println("Address: " + address);

        System.out.println("Phone: " + phone);

        System.out.println("Email: " + email);

        System.out.println("UID: " + uid);

        System.out.println("PAN: " + pan);

    }

}

class Account {

    private static int nextAccountNumber = 1001;

    private int accountNumber;

    private Customer customer;

    private double balance;

    public Account(Customer customer) {

*this*.accountNumber = nextAccountNumber++;

*this*.customer = customer;

*this*.balance = 0.0;

    }

    public int getAccountNumber() {

        return accountNumber;

    }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("₹" + amount + " deposited successfully.");

        } else {

            System.out.println("Invalid deposit amount.");

        }

    }

    public void withdraw(double amount) {

        if (amount > 0 && amount <= balance) {

            balance -= amount;

            System.out.println("₹" + amount + " withdrawn successfully.");

        } else {

            System.out.println("Insufficient balance or invalid amount!");

        }

    }

    public void displayAccountDetails() {

        System.out.println("Account Number: " + accountNumber);

        customer.displayCustomer();

        System.out.println("Balance: ₹" + balance);

    }

}

class AccountManagement {

    private Account[] accounts;

    private int numberOfAccounts;

    public AccountManagement(int size) {

        accounts = new Account[size];

        numberOfAccounts = 0;

    }

    public void openAccount(String name, String address, Long phone, String email, String uid, String pan) {

        if (numberOfAccounts < accounts.length) {

            Customer customer = new Customer(name, address, phone, email, uid, pan);

            accounts[numberOfAccounts] = new Account(customer);

            System.out.println("Account created successfully with Account Number: " + accounts[numberOfAccounts].getAccountNumber());

            numberOfAccounts++;

        } else {

            System.out.println("Cannot create more accounts. Storage is full.");

        }

    }

    public Account findAccount(int accountNumber) {

        for (int i = 0; i < numberOfAccounts; i++) {

            if (accounts[i].getAccountNumber() == accountNumber) {

                return accounts[i];

            }

        }

        return null;

    }

    public void closeAccount(int accountNumber) {

        for (int i = 0; i < numberOfAccounts; i++) {

            if (accounts[i].getAccountNumber() == accountNumber) {

                System.out.println("Closing account number: " + accountNumber);

                accounts[i] = accounts[numberOfAccounts - 1];

                numberOfAccounts--;

                return;

            }

        }

        System.out.println("Account not found.");

    }

    public void depositToAccount(int accountNumber, double amount, Scanner scanner) {

        Account account = findAccount(accountNumber);

        if (account != null) {

            account.deposit(amount);

        } else {

            System.out.println("Account not found. Would you like to create a new account? (yes/no): ");

            String response = scanner.nextLine();

            if (response.equalsIgnoreCase("yes")) {

                System.out.print("Enter customer name: ");

                String name = scanner.nextLine();

                System.out.print("Enter customer address: ");

                String address = scanner.nextLine();

                System.out.print("Enter customer phone: ");

                Long phone = scanner.nextLong();

                scanner.nextLine();

                System.out.print("Enter customer email: ");

                String email = scanner.nextLine();

                System.out.print("Enter customer UID: ");

                String uid = scanner.nextLine();

                System.out.print("Enter customer PAN: ");

                String pan = scanner.nextLine();

                openAccount(name, address, phone, email, uid, pan);

                accounts[numberOfAccounts - 1].deposit(amount);

            }

        }

    }

    public void withdrawFromAccount(int accountNumber, double amount, Scanner scanner) {

        Account account = findAccount(accountNumber);

        if (account != null) {

            account.withdraw(amount);

        } else {

            System.out.println("Account not found. Would you like to create a new account? (yes/no): ");

            String response = scanner.nextLine();

            if (response.equalsIgnoreCase("yes")) {

                System.out.print("Enter customer name: ");

                String name = scanner.nextLine();

                System.out.print("Enter customer address: ");

                String address = scanner.nextLine();

                System.out.print("Enter customer phone: ");

                Long phone = scanner.nextLong();

                scanner.nextLine();

                System.out.print("Enter customer email: ");

                String email = scanner.nextLine();

                System.out.print("Enter customer UID: ");

                String uid = scanner.nextLine();

                System.out.print("Enter customer PAN: ");

                String pan = scanner.nextLine();

                openAccount(name, address, phone, email, uid, pan);

                accounts[numberOfAccounts - 1].withdraw(amount);

            }

        }

    }

    public void displayAccountDetails(int accountNumber) {

        Account account = findAccount(accountNumber);

        if (account != null) {

            account.displayAccountDetails();

        } else {

            System.out.println("Account not found.");

        }

    }

    public void displayAllAccounts() {

        if (numberOfAccounts == 0) {

            System.out.println("No accounts available.");

        } else {

            for (int i = 0; i < numberOfAccounts; i++) {

                accounts[i].displayAccountDetails();

                System.out.println();

            }

        }

    }

}

public class BankApp {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        AccountManagement accountManagement = new AccountManagement(10);

        while (true) {

            System.out.println("\nBank Account Management System");

            System.out.println("1. Open an account");

            System.out.println("2. Close an account");

            System.out.println("3. Display account details");

            System.out.println("4. Display all accounts");

            System.out.println("5. Deposit");

            System.out.println("6. Withdraw");

            System.out.println("7. Exit");

            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            scanner.nextLine();

            switch (choice) {

                case 1:

                    System.out.print("Enter customer name: ");

                    String name = scanner.nextLine();

                    System.out.print("Enter customer address: ");

                    String address = scanner.nextLine();

                    System.out.print("Enter customer phone: ");

                    Long phone = scanner.nextLong();

                    scanner.nextLine();

                    System.out.print("Enter customer email: ");

                    String email = scanner.nextLine();

                    System.out.print("Enter customer UID: ");

                    String uid = scanner.nextLine();

                    System.out.print("Enter customer PAN: ");

                    String pan = scanner.nextLine();

                    accountManagement.openAccount(name, address, phone, email, uid, pan);

                    break;

                case 2:

                    System.out.print("Enter account number to close: ");

                    int accountNumberToClose = scanner.nextInt();

                    accountManagement.closeAccount(accountNumberToClose);

                    break;

                case 3:

                    System.out.print("Enter account number to display: ");

                    int accountNumberToDisplay = scanner.nextInt();

                    accountManagement.displayAccountDetails(accountNumberToDisplay);

                    break;

                case 4:

                    accountManagement.displayAllAccounts();

                    break;

                case 5:

                    System.out.print("Enter account number for deposit: ");

                    int accountNumberToDeposit = scanner.nextInt();

                    System.out.print("Enter amount to deposit: ");

                    double depositAmount = scanner.nextDouble();

                    scanner.nextLine();

                    accountManagement.depositToAccount(accountNumberToDeposit, depositAmount, scanner);

                    break;

                case 6:

                    System.out.print("Enter account number for withdrawal: ");

                    int accountNumberToWithdraw = scanner.nextInt();

                    System.out.print("Enter amount to withdraw: ");

                    double withdrawalAmount = scanner.nextDouble();

                    scanner.nextLine();

                    accountManagement.withdrawFromAccount(accountNumberToWithdraw, withdrawalAmount, scanner);

                    break;

                case 7:

                    System.out.println("Exiting program.");

                    scanner.close();

                    System.exit(0);

                default:

                    System.out.println("Invalid choice. Please try again.");

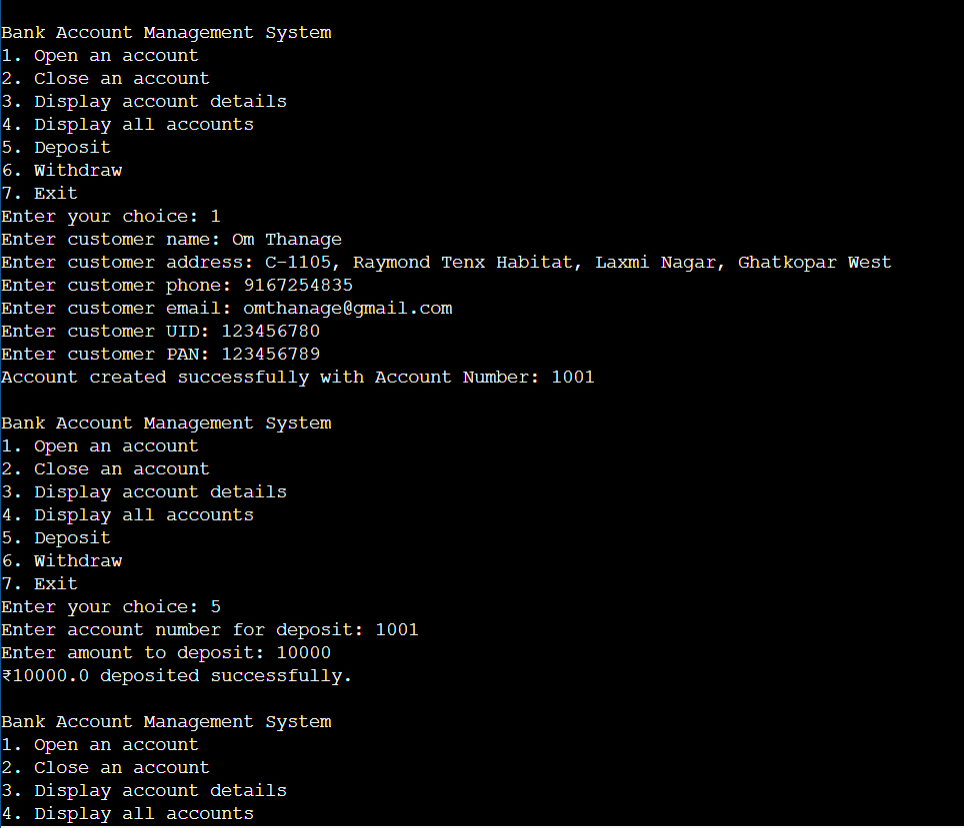
            }

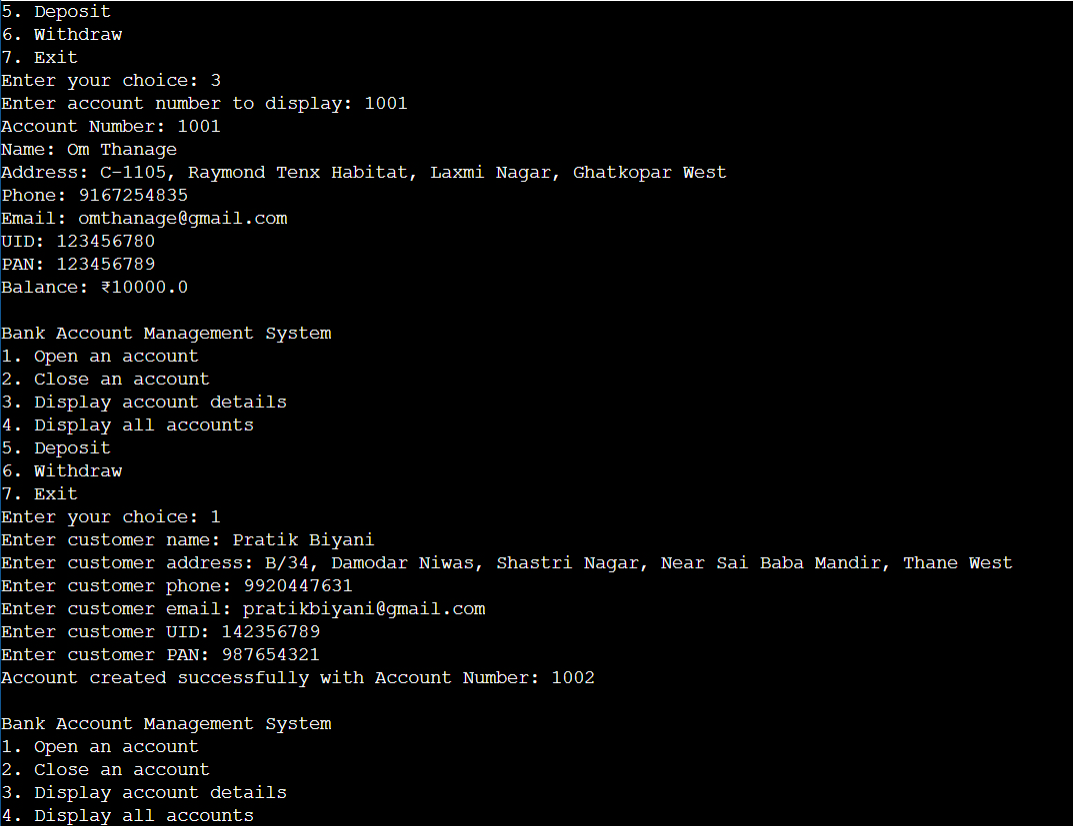
        }

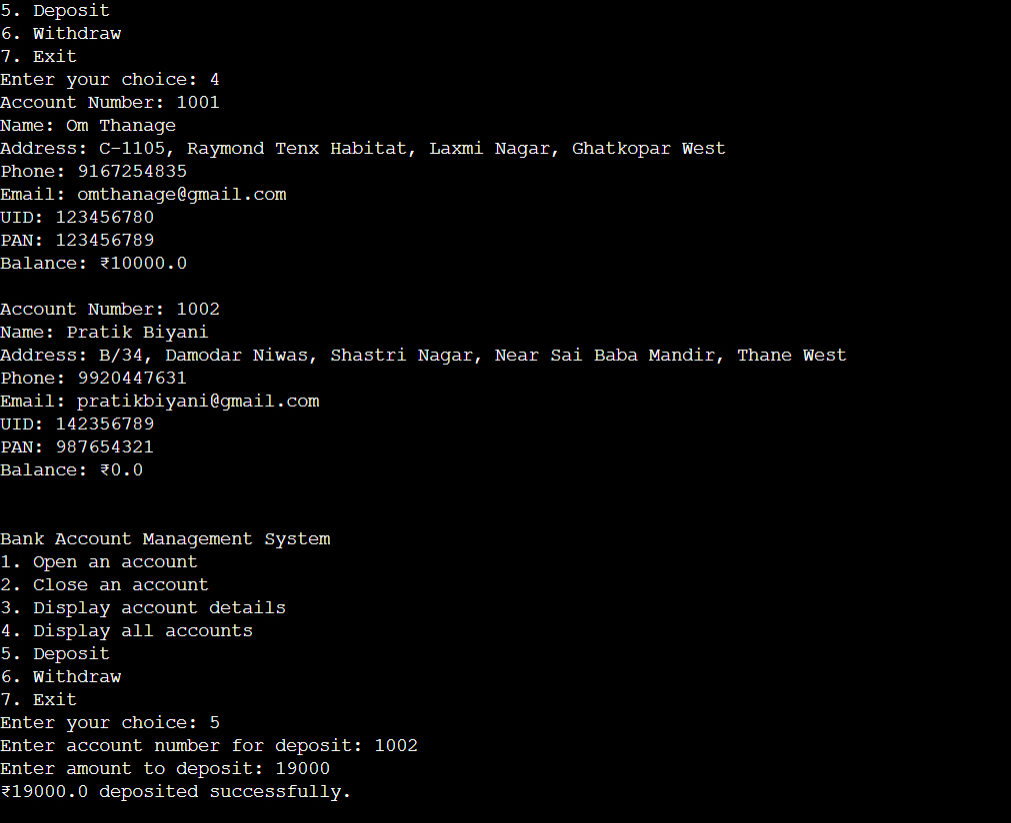
    }

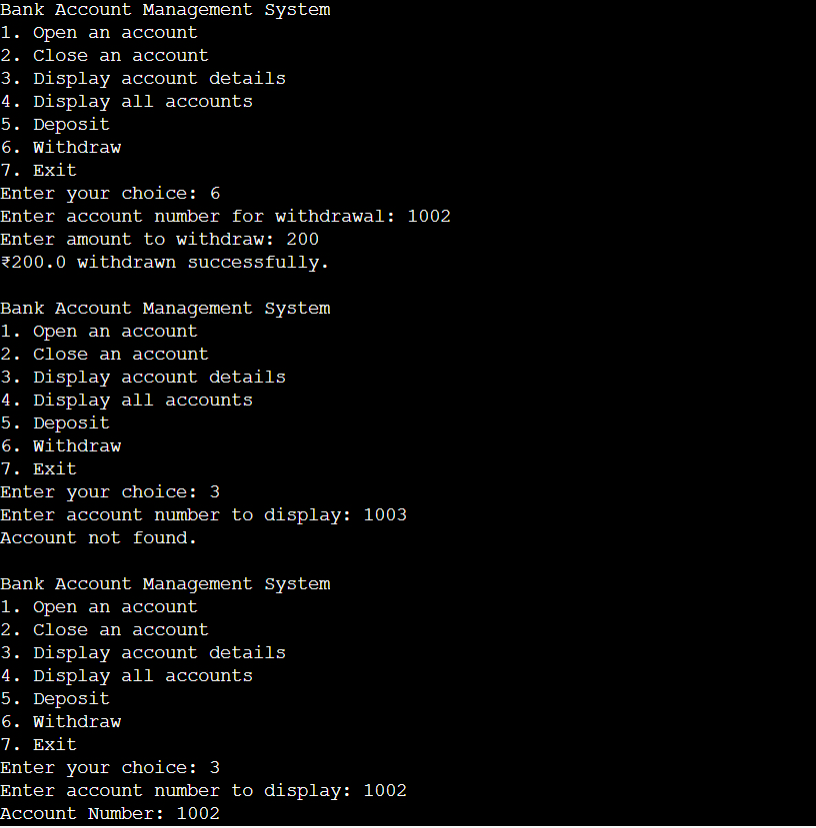
}

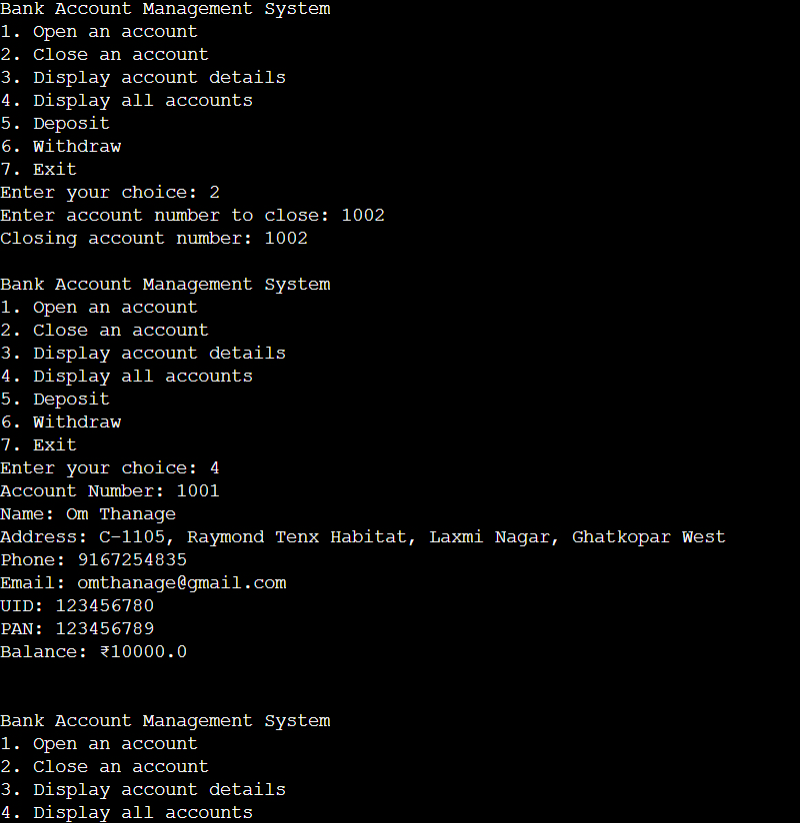
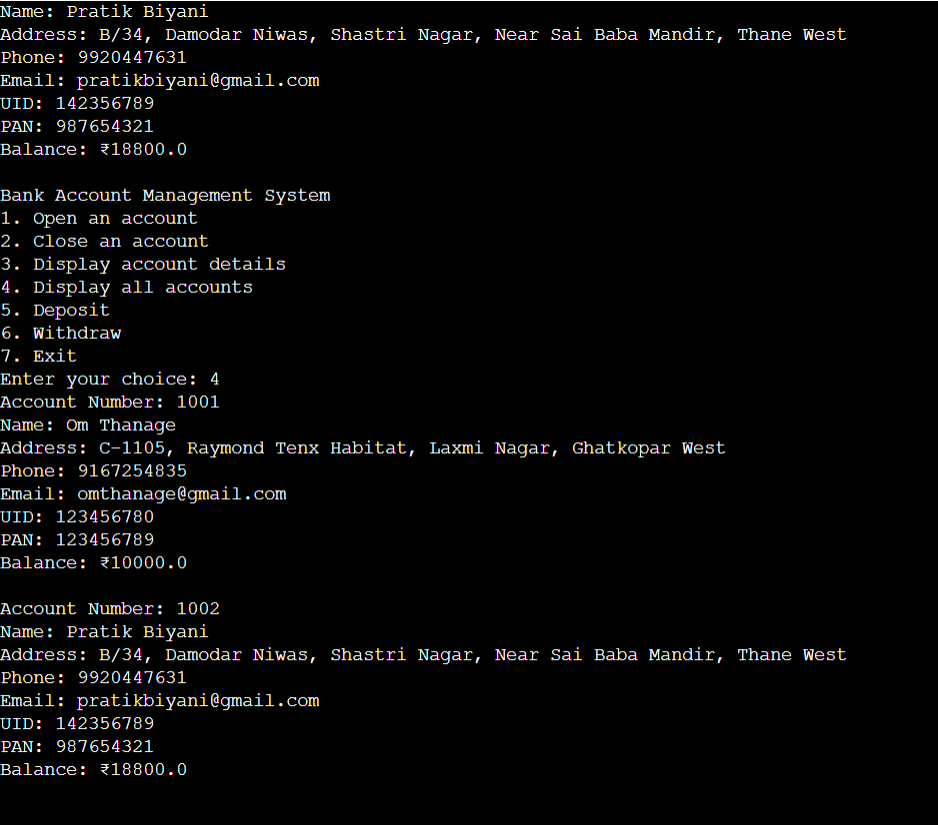
**Output:**

****

****







**Conclusion:**

**Develop a Bank Management App with the help of classes, constructor, mutators and accessors**

**Date: 5/09/24 Signature of faculty in-charge**

**Post Lab Descriptive Questions:**

**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: Option a**

**Q.2** The Object array is created in \_\_Heap Memory\_\_\_  
a)Heap memory  
b) Stack memory  
c) HDD  
d) ROM

**Q.3** Explain the difference between Jagged Array and Array of Object .

**Jagged Array**

**Definition**: A jagged array is an array of arrays where each inner array can have different lengths. It’s also known as a "ragged array" or "array of arrays."

**Characteristics**:

* **Structure**: Each element of a jagged array is an array itself. These inner arrays can have different lengths, making the overall structure "jagged."
* **Memory**: Since inner arrays can be of varying lengths, memory allocation is more flexible compared to a multidimensional array where each row must have the same length.
* **Initialization**: You initialize a jagged array with an array of arrays, where each inner array can be independently sized.

**Example :**

int[][] jaggedArray = new int[3][]; // Create an array of 3 int arrays

jaggedArray[0] = new int[2]; // First array with 2 elements

jaggedArray[1] = new int[3]; // Second array with 3 elements

jaggedArray[2] = new int[1]; // Third array with 1 element

**Array of Objects**

**Definition**: An array of objects is a single-dimensional array where each element is an instance of a class (object).

**Characteristics**:

* **Structure**: Each element in the array is an object, and you can access the properties and methods of the class through these objects.
* **Memory**: Memory is allocated for the objects themselves and their data, but the structure of the array is fixed (i.e., all objects are in a single array with a uniform length).
* **Initialization**: You initialize an array of objects with a specified number of objects, which can be created and initialized as needed.

**Example:**

class Person {

String name;

int age;

Person(String name, int age) {

this.name = name;

this.age = age;

}

}

Person[] peopleArray = new Person[3]; // Create an array for 3 Person objects

peopleArray[0] = new Person("Alice", 30);

peopleArray[1] = new Person("Bob", 25);

peopleArray[2] = new Person("Charlie", 35);

**Key Differences**

1. **Data Structure**:
   * **Jagged Array**: An array of arrays, where inner arrays can have different lengths.
   * **Array of Objects**: A single-dimensional array where each element is an object.
2. **Flexibility**:
   * **Jagged Array**: More flexible in terms of memory usage, as inner arrays can vary in size.
   * **Array of Objects**: Fixed length for the array, but each object can have varied data.
3. **Usage**:
   * **Jagged Array**: Useful when dealing with data structures where each row (or sub-array) might have different lengths (e.g., matrix with uneven rows).
   * **Array of Objects**: Useful when you need to store and manage a collection of objects of the same type.
4. **Accessing Data**:
   * **Jagged Array**: Access elements using two indices (e.g., jaggedArray[i][j]).
   * **Array of Objects**: Access elements using a single index (e.g., peopleArray[i]) and then access object properties or methods (e.g., peopleArray[i].name).