Batch: A1	**Roll No.: 16010123012**
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: Write a program which accepts information about n no of customers from user.

Create an array of objects to store account_id, name, and 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:

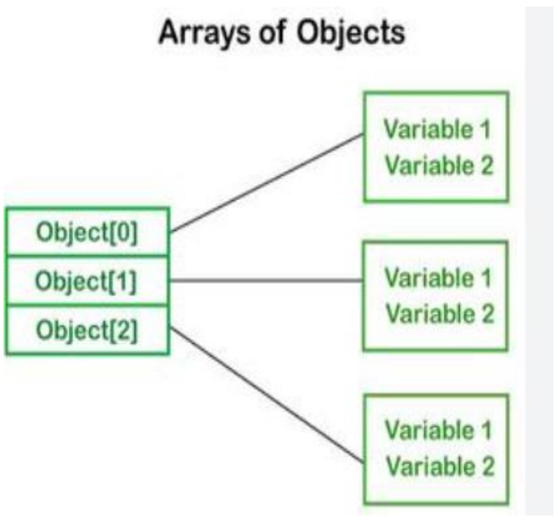
CO1: Apply the features of object oriented programming languages. (C++ and Java)
CO2: Explore arrays, vectors, classes and objects in C++ and Java

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, 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 o bjects

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;
```

Algorithm:

1. Initialize a Bank object with an array of Customer objects, with size n.

2. While Loop:

Repeat the following until the user selects the "Exit" option:

Display the menu with options: Add Account, Delete Account, Display Account, and Exit.

3. Add Account:

Input accountId, name, and balance.

Check if the count is less than the length of the customers array:

If true, create a new Customer object, add it to the array, increment the count, and print a success message.

If false, print a message that the customer limit has been reached.

4. Delete Account:

Input accountId.

Search for the customer with the matching accountId in the array.

If found, shift all customers after the matched customer one position to the left.

Decrease the count by 1 and print a success message.

If not found, print an error message that the account ID was not found.

5. Display Account:

Input accountId.

Search for the customer with the matching accountId in the array.

If found, display the account details.

If not found, print an error message that the account ID was not found.

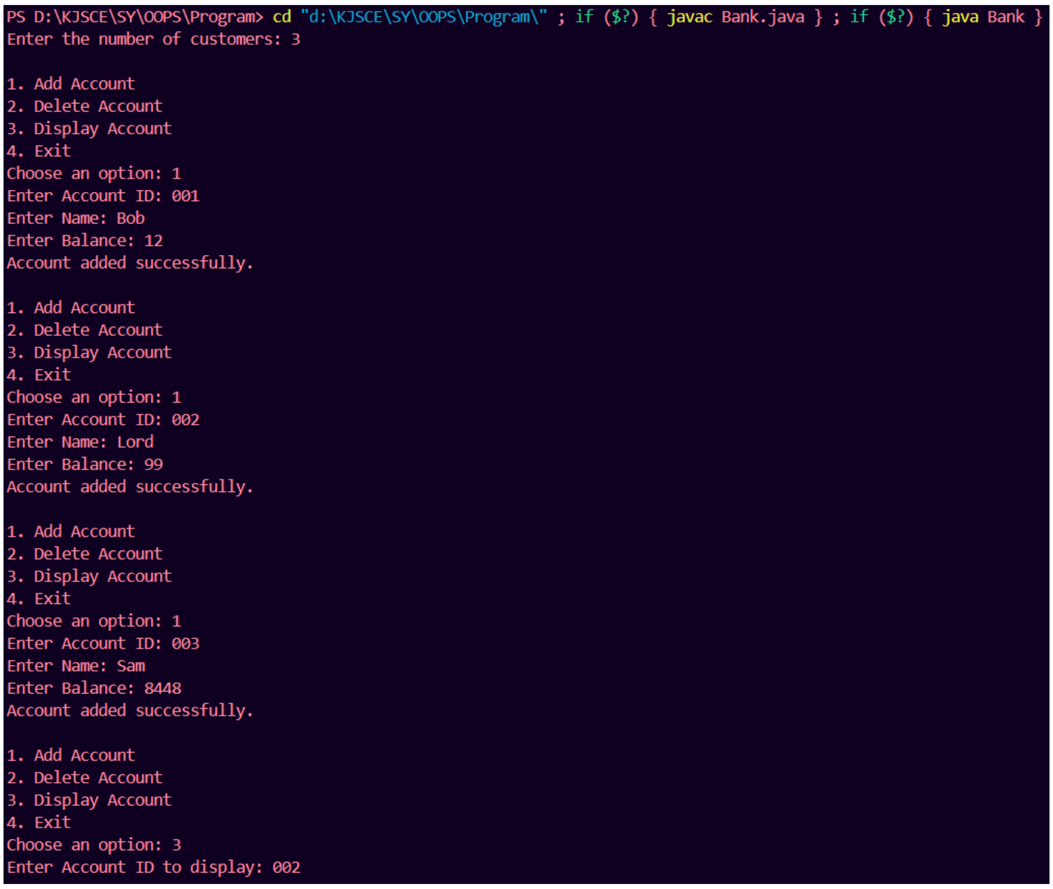
6. Exit:

Close the scanner and exit the program.

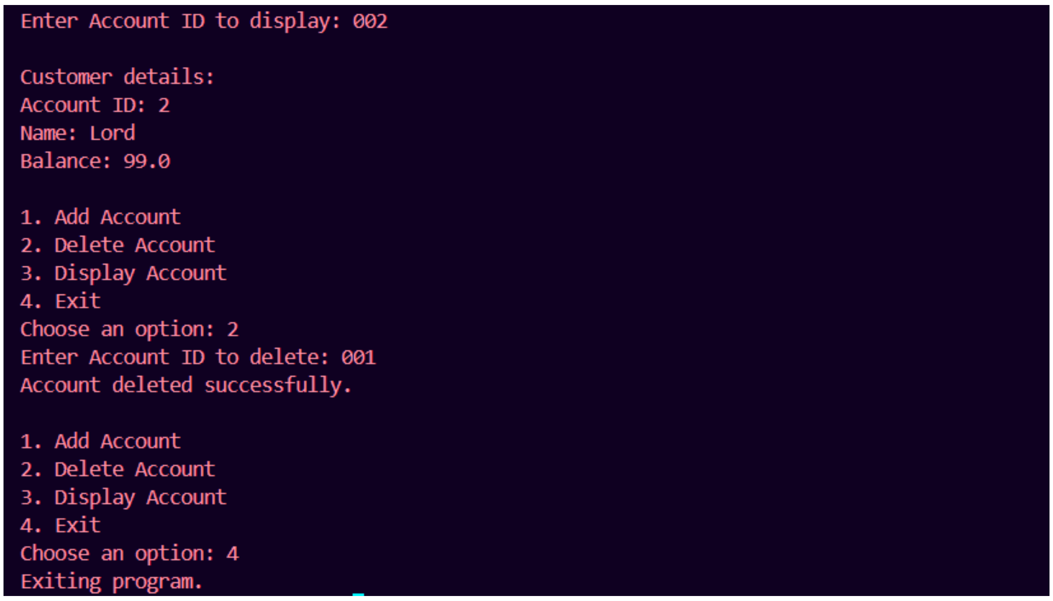
Implementation details:

```
import java.util.Scanner;

class Customer {

    int accountId;

    String name;

    double balance;

    public Customer(int accountId, String name, double balance) {

        this.accountId = accountId;

        this.name = name;

        this.balance = balance;

    }

    public void displayDetails() {

        System.out.println("Account ID: " + accountId);

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

        System.out.println("Balance: " + balance);
    }
}
public class Bank {
    private Customer[] customers;
    private int count;
    public Bank(int n) {
        customers = new Customer[n];
        count = 0;
    }
    public void addAccount(int accountId, String name, double balance) {
        if (count < customers.length) {
            customers[count++] = new Customer(accountId, name, balance);
            System.out.println("Account added successfully.");
        } else {
            System.out.println("Customer limit reached, cannot add more accounts.");
        }
    }
    public void deleteAccount(int accountId) {
        for (int i = 0; i < count; i++) {
            if (customers[i].accountId == accountId) {
                for (int j = i; j < count - 1; j++) {
                    customers[j] = customers[j + 1];
                }
                customers[--count] = null;
                System.out.println("Account deleted successfully.");
                return;
            }
        }
        System.out.println("Account ID not found.");
    }
    public void displayAccount(int accountId) {
        for (int i = 0; i < count; i++) {
            if (customers[i].accountId == accountId) {
                System.out.println("\nCustomer details:");
                customers[i].displayDetails();
                return;
            }
        }
        System.out.println("Account ID not found.");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of customers: ");
        int n = scanner.nextInt();
        Bank bank = new Bank(n);
        while (true) {
            System.out.println("\n1. Add Account\n2. Delete Account\n3. Display
Account\n4. Exit");
            System.out.print("Choose an option: ");
            int option = scanner.nextInt();
            switch (option) {
                case 1:
                    System.out.print("Enter Account ID: ");
                    int accountId = scanner.nextInt();
                    scanner.nextLine();
                    System.out.print("Enter Name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter Balance: ");
                    double balance = scanner.nextDouble();
                    bank.addAccount(accountId, name, balance);
                    break;
                case 2:
                    System.out.print("Enter Account ID to delete: ");
                    int deleteId = scanner.nextInt();
                    bank.deleteAccount(deleteId);
                    break;
                case 3:
                    System.out.print("Enter Account ID to display: ");
                    int displayId = scanner.nextInt();
                    bank.displayAccount(displayId);
                    break;
                case 4:
                    System.out.println("Exiting program.");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid option! Please try again.");
            }
        }
    }
}
```

Output:



```
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac Bank.java } ; if ($?) { java Bank }
Enter the number of customers: 3

1. Add Account
2. Delete Account
3. Display Account
4. Exit
Choose an option: 1
Enter Account ID: 001
Enter Name: Bob
Enter Balance: 12
Account added successfully.

1. Add Account
2. Delete Account
3. Display Account
4. Exit
Choose an option: 1
Enter Account ID: 002
Enter Name: Lord
Enter Balance: 99
Account added successfully.

1. Add Account
2. Delete Account
3. Display Account
4. Exit
Choose an option: 1
Enter Account ID: 003
Enter Name: Sam
Enter Balance: 8448
Account added successfully.

1. Add Account
2. Delete Account
3. Display Account
4. Exit
Choose an option: 3
Enter Account ID to display: 002
```



```
Enter Account ID to display: 002

Customer details:
Account ID: 2
Name: Lord
Balance: 99.0

1. Add Account
2. Delete Account
3. Display Account
4. Exit
Choose an option: 2
Enter Account ID to delete: 001
Account deleted successfully.

1. Add Account
2. Delete Account
3. Display Account
4. Exit
Choose an option: 4
Exiting program.
```

Conclusion:
This experiment utilizes an array of 'Customer' objects to handle fundamental banking operations, such as adding, deleting, and displaying accounts. It accommodates a fixed number of customers and employs linear search for account management.

Date: 04 / 09 / 2024 **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: a) Array_Name[4].data_variable_name;

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

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

Aspect	**Jagged Array**	**Array of Objects**
Definition	A multi-dimensional array where each row (sub-array) can have different lengths.	A single-dimensional array where each element is an instance of a class.
Structure	An array of arrays; each sub-array can have different sizes.	An array where all elements are objects of the same class.

Data Type	Contains arrays as its elements; each inner array can vary in size.	Contains objects of a specific class type.
Dimension	Multi-dimensional (typically a 2D array with variable row sizes).	Single-dimensional.
Flexibility	Allows for variable-sized rows, suitable for uneven data sets.	Uniform in type; all elements are instances of the same class.
Usage	Useful for representing data with varying row sizes, such as tables with different column counts.	Ideal for managing collections of objects with similar attributes and behaviours.

.