**Batch: A2 Roll No.:16010123032**

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

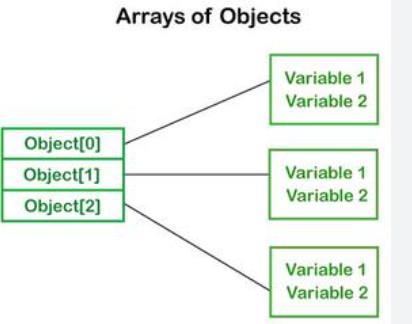
**Books/ Journals/ Websites referred:**

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++)

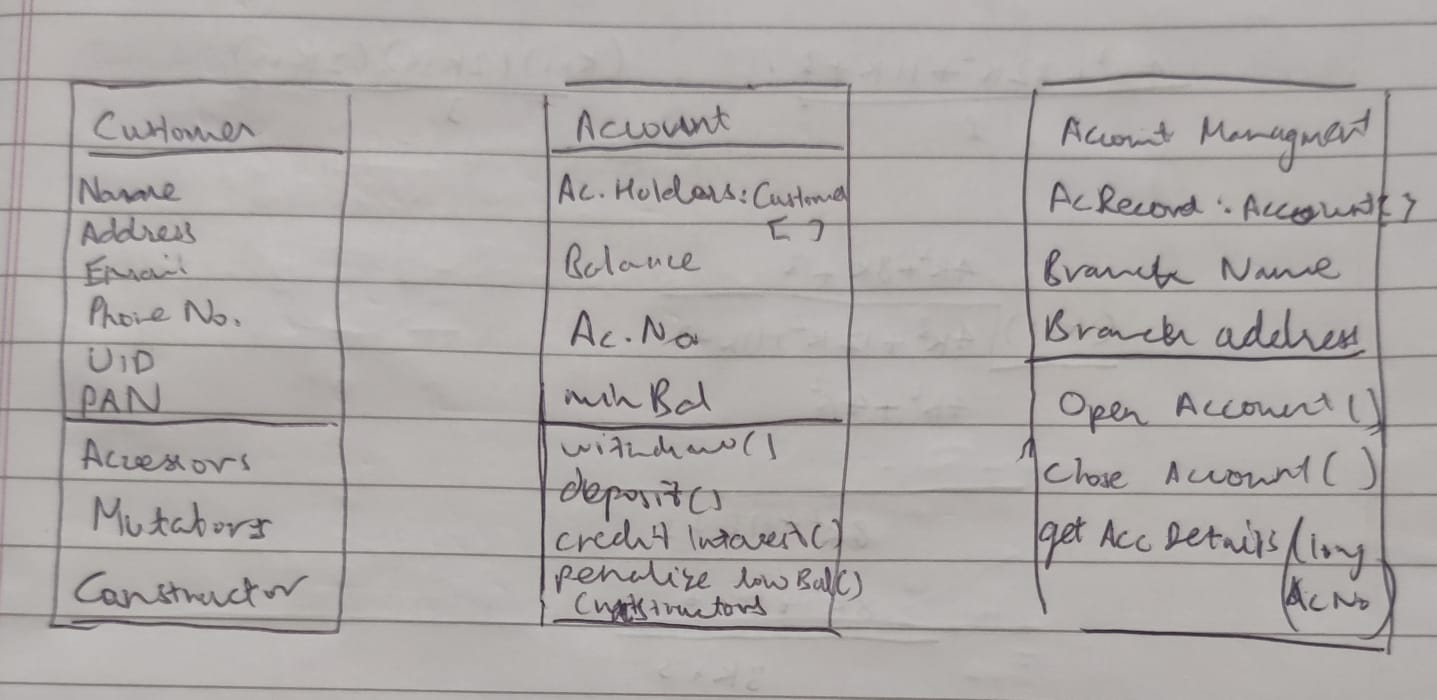
{  
studentArray[i]=new Student(r,name,average);  
}

* 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:**

#### Step 1: Initialize the Program

1.1. Declare an array of Account objects with a maximum size of 100. 1.2. Initialize an integer variable accountCount to keep track of the number of accounts.

#### Step 2: Main Menu Loop

2.1. Start a loop that continues until the user chooses to exit. 2.2. Display the main menu options:

* 1. Open an Account
  2. Close an Account
  3. Display Account Details
  4. Exit

2.3. Prompt the user to enter their choice.

#### Step 3: Handle User Choices

3.1. Case 1: Open an Account

* Call the openAccount function:
  + 3.1.1. Create a new Account object using the accountCount as a parameter to generate the account number.
  + 3.1.2. Add the new Account object to the accounts array.
  + 3.1.3. Increment the accountCount.
  + 3.1.4. Display a message indicating the account was opened successfully.

3.2. Case 2: Close an Account

* Call the closeAccount function:
  + 3.2.1. Prompt the user to enter the account number to close.
  + 3.2.2. Search for the account in the accounts array by comparing the entered account number with the account numbers in the array.
  + 3.2.3. If the account is found:
    - 3.2.3.1. Shift the accounts in the array to overwrite the account being closed.
    - 3.2.3.2. Decrement the accountCount.
    - 3.2.3.3. Set the last account in the array to null.
    - 3.2.3.4. Display a message indicating the account was closed successfully.
  + 3.2.4. If the account is not found, display an error message.

3.3. Case 3: Display Account Details

* Call the displayAccountDetails function:
  + 3.3.1. Prompt the user to enter the account number to display.
  + 3.3.2. Search for the account in the accounts array.
  + 3.3.3. If the account is found:
    - 3.3.3.1. Display the account number, balance, minimum balance, and details of the account holders.
  + 3.3.4. If the account is not found, display an error message.

3.4. Case 4: Exit

* Display a message indicating that the program is exiting.
* Exit the loop.

3.5. Default Case: Invalid Choice

* If the user enters an invalid choice, display an error message and prompt them to try again.

#### Step 4: End Program

4.1. The program terminates after the user selects the exit option.

### Additional Operations within the Account Class

* Constructor: Initializes an account with a specified balance, auto-generated account number, minimum balance, and account holder details.
* withDraw Method:
  + Prompts the user to enter an amount to withdraw.
  + Checks if the balance is sufficient for the withdrawal.
  + If sufficient, deducts the amount from the balance and displays the updated balance.
  + If not, displays an error message.
* Deposit Method:
  + Prompts the user to enter an amount to deposit.
  + Adds the deposit amount to the balance.
  + Displays the updated balance.
* creditInterest Method:
  + Calculates interest at a fixed rate (e.g., 5%) and credits it to the balance.
  + Displays the updated balance.
* penalizeLowBalance Method:
  + Checks if the balance is below the minimum balance.
  + If below, deducts a penalty from the balance and displays the updated balance.
  + If not, displays a message indicating sufficient balance.
* displayAccountDetails Method:
  + Displays the account number, balance, minimum balance, and details of all account holders associated with the account.

**Implementation details:**

import java.util.Scanner;

class Customer {

private String Name, Address, Email;

private long Phone\_No, UID, PAN;

public String getName() {

return Name;

}

public String getAddress() {

return Address;

}

public String getEmail() {

return Email;

}

public long getPhone\_No() {

return Phone\_No;

}

public long getUID() {

return UID;

}

public long getPAN() {

return PAN;

}

public void setName(String Name) {

this.Name = Name;

}

public void setAddress(String Address) {

this.Address = Address;

}

public void setEmail(String Email) {

this.Email = Email;

}

public void setPhone\_No(long Phone\_No) {

this.Phone\_No = Phone\_No;

}

public void setUID(long UID) {

this.UID = UID;

}

public void setPAN(long PAN) {

this.PAN = PAN;

}

Customer() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Name: ");

this.Name = sc.nextLine();

System.out.print("Enter Address: ");

this.Address = sc.nextLine();

System.out.print("Enter Email: ");

this.Email = sc.nextLine();

System.out.print("Enter Phone No.: ");

this.Phone\_No = sc.nextLong();

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

this.UID = sc.nextLong();

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

this.PAN = sc.nextLong();

}

}

class Account {

private Customer[] accountHolders;

private int c;

private long Balance, Ac\_No, Min\_Bal;

public long getBalance() {

return Balance;

}

public long getAc\_No() {

return Ac\_No;

}

public long getMin\_Bal() {

return Min\_Bal;

}

public void setBalance(long Balance) {

this.Balance = Balance;

}

public void setAc\_No(long Ac\_No) {

this.Ac\_No = Ac\_No;

}

public void setMin\_Bal(long Min\_Bal) {

this.Min\_Bal = Min\_Bal;

}

Account(int p)

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter Balance: ");

this.Balance = sc.nextLong();

Ac\_No = 1000+p;

System.out.print("Enter Minimum Balance: ");

this.Min\_Bal = sc.nextLong();

System.out.print("Enter Number of Account Holders: ");

this.c = sc.nextInt();

accountHolders = new Customer[c];

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

accountHolders[i] = new Customer();

}

}

public void withDraw() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Amount you want to Withdraw: ");

long Amt = sc.nextLong();

if (Balance >= Amt) {

Balance -= Amt;

System.out.println("Withdrawal successful. New Balance: " + Balance);

} else {

System.out.println("Invalid Amount. Insufficient funds.");

}

}

public void Deposit() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Amount you want to Deposit: ");

long Amt = sc.nextLong();

Balance += Amt;

System.out.println("Deposit successful. New Balance: " + Balance);

}

public void creditInterest() {

int r = 5; // Assuming 5% interest rate

Balance += Balance \* (r / 100.0);

System.out.println("Interest credited. New Balance: " + Balance);

}

public void penalizeLowBalance() {

if (Balance < Min\_Bal) {

Balance -= 300;

System.out.println("Penalty applied due to low balance. New Balance: " + Balance);

} else {

System.out.println("Account has sufficient Balance.");

}

}

public void displayAccountDetails() {

System.out.println("Account Number: " + Ac\_No);

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

System.out.println("Minimum Balance: " + Min\_Bal);

System.out.println("Account Holders:");

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

System.out.println(" Name: " + accountHolders[i].getName());

System.out.println(" Address: " + accountHolders[i].getAddress());

System.out.println(" Email: " + accountHolders[i].getEmail());

System.out.println(" Phone No: " + accountHolders[i].getPhone\_No());

System.out.println(" UID: " + accountHolders[i].getUID());

System.out.println(" PAN: " + accountHolders[i].getPAN());

System.out.println();

}

}

}

public class AccountManagement {

private static Account[] accounts = new Account[100];

private static int accountCount = 0;

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int choice;

do {

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. Exit");

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

choice = sc.nextInt();

switch (choice) {

case 1:

openAccount();

break;

case 2:

closeAccount();

break;

case 3:

displayAccountDetails();

break;

case 4:

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

break;

default:

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

}

} while (choice != 4);

}

private static void openAccount() {

accounts[accountCount++] = new Account(accountCount);

System.out.println("Account opened successfully.");

}

private static void closeAccount() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Account Number to close: ");

long acNo = sc.nextLong();

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

if (accounts[i].getAc\_No() == acNo) {

for (int j = i; j < accountCount - 1; j++) {

accounts[j] = accounts[j + 1];

}

accounts[--accountCount] = null;

System.out.println("Account closed successfully.");

return;

}

}

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

}

private static void displayAccountDetails() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Account Number to display: ");

long acNo = sc.nextLong();

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

if (accounts[i].getAc\_No() == acNo) {

accounts[i].displayAccountDetails();

return;

}

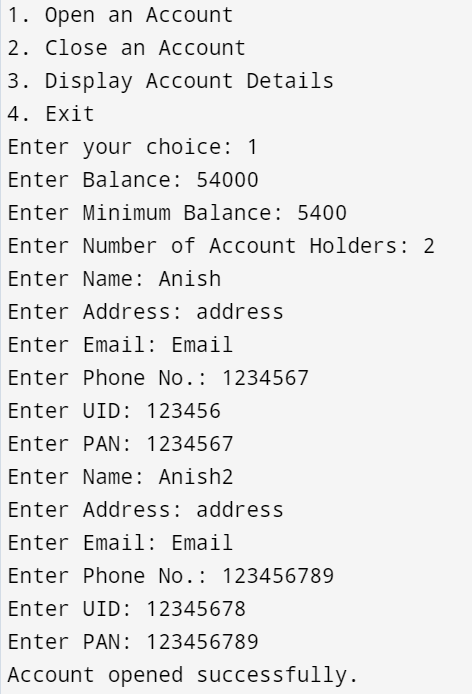
}

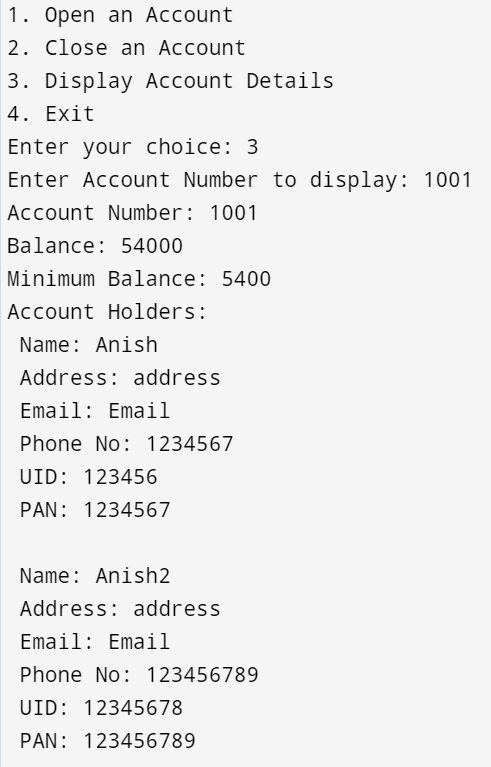
System.out.println("Account not found.");

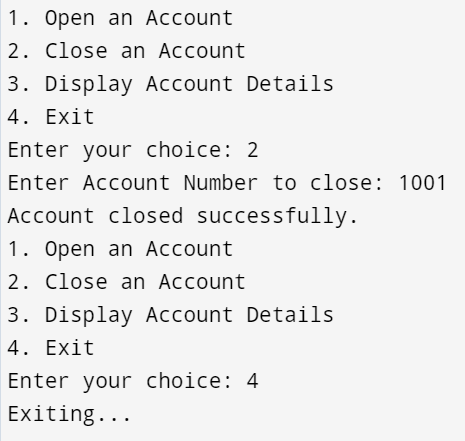
}

}

**Output:**







**Conclusion: Understood implementation of Arrays of objects and classes.**

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ 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:** a)Heap memory

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

**Jagged Array:**

* **A jagged array is an array of arrays where the inner arrays can have different lengths.**
* **It is essentially a multi-dimensional array where each row (or dimension) can vary in size.**

**Array of Objects:**

* **An array of objects is an array where each element is a reference to an object.**
* **All the elements in this array are references to instances of a class, which can be the same or different objects.**