JAVA ASSIGNMENT 1

NAME-BHAVYA RATTAN

COURSE-BCA(AI &DS)

SECTION-B

ROLL NO- 2401201004

CODE:-

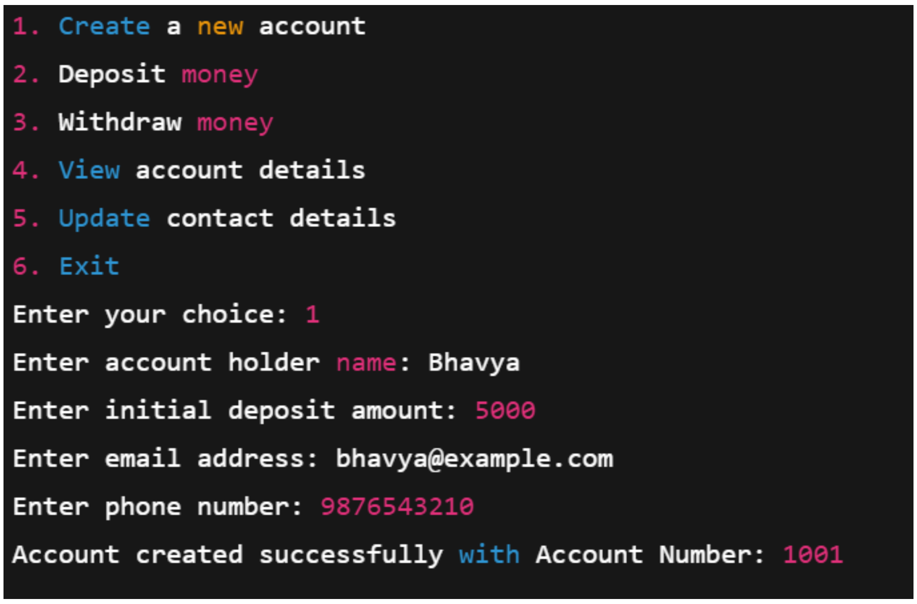
```
class Account {
private int accountNumber;
private String accountHolderName;
private double balance;
private String email;
private String phoneNumber;
public Account(int accountNumber, String accountHolderName, double
balance, String email, String phoneNumber) {
this.accountNumber = accountNumber;
this.accountHolderName = accountHolderName;
this.balance = balance;
this.email = email;
this.phoneNumber = phoneNumber;
}
public void deposit(double amount) {
if (amount > 0) {
balance += amount;
System.out.println("Deposit successful. New balance: " +
balance);
} else {
System.out.println("Invalid deposit amount.");
}
}
public void withdraw(double amount) {
if (amount > 0 && balance >= amount) {
balance -= amount;
System.out.println("Withdrawal successful. Remaining balance:
" + balance);
} else {
System.out.println("Invalid withdrawal amount or insufficient
balance.");
}
}
public void displayAccountDetails() {
System.out.println("Account Number: " + accountNumber);
System.out.println("Account Holder: " + accountHolderName);
System.out.println("Balance: " + balance);
System.out.println("Email: " + email);
System.out.println("Phone Number: " + phoneNumber);
}
public void updateContactDetails(String email, String phoneNumber) {
this.email = email;
this.phoneNumber = phoneNumber;
System.out.println("Contact details updated successfully!");
}
public int getAccountNumber() {
return accountNumber;
```

```
}
}
public class UserInterface {
private static Account[] accounts = new Account[100];
private static int accountCount = 0;
private static int nextAccountNumber = 1001;
private static Scanner sc = new Scanner(System.in);
public static void createAccount() {
System.out.print("Enter account holder name: ");
String name = sc.nextLine();
System.out.print("Enter initial deposit amount: ");
double balance = sc.nextDouble();
sc.nextLine();
System.out.print("Enter email address: ");
String email = sc.nextLine();
System.out.print("Enter phone number: ");
String phone = sc.nextLine();
accounts[accountCount] = new Account(nextAccountNumber, name,
balance, email, phone);
System.out.println("Account created successfully with Account
Number: " + nextAccountNumber);
nextAccountNumber++;
accountCount++;
}
public static Account findAccount(int accountNumber) {
for (int i = 0; i < accountCount; i++) {
if (accounts[i].getAccountNumber() == accountNumber) {
return accounts[i];
}
}
return null;
}
public static void performDeposit() {
System.out.print("Enter account number: ");
int accNum = sc.nextInt();
System.out.print("Enter deposit amount: ");
double amount = sc.nextDouble();
Account acc = findAccount(accNum);
if (acc != null) {
acc.deposit(amount);
} else {
System.out.println("Account not found.");
}
}
public static void performWithdrawal() {
System.out.print("Enter account number: ");
int accNum = sc.nextInt();
System.out.print("Enter withdrawal amount: ");
double amount = sc.nextDouble();
Account acc = findAccount(accNum);
if (acc != null) {
acc.withdraw(amount);
} else {
System.out.println("Account not found.");
}
}
```

```
public static void showAccountDetails() {
System.out.print("Enter account number: ");
int accNum = sc.nextInt();
Account acc = findAccount(accNum);
if (acc != null) {
acc.displayAccountDetails();
} else {
System.out.println("Account not found.");
}
}
public static void updateContact() {
System.out.print("Enter account number: ");
int accNum = sc.nextInt();
sc.nextLine();
System.out.print("Enter new email: ");
String email = sc.nextLine();
System.out.print("Enter new phone number: ");
String phone = sc.nextLine();
Account acc = findAccount(accNum);
if (acc != null) {
acc.updateContactDetails(email, phone);
} else {
System.out.println("Account not found.");
}
}
public static void mainMenu() {
int choice;
do {
```

```
System.out.println("\n--- Banking Application ---");
System.out.println("1. Create a new account");
System.out.println("2. Deposit money");
System.out.println("3. Withdraw money");
System.out.println("4. View account details");
System.out.println("5. Update contact details");
System.out.println("6. Exit");
System.out.print("Enter your choice: ");
choice = sc.nextInt();
sc.nextLine();
switch (choice) {
case 1: createAccount(); break;
case 2: performDeposit(); break;
case 3: performWithdrawal(); break;
case 4: showAccountDetails(); break;
case 5: updateContact(); break;
case 6: System.out.println("Exiting... Thank you!");
break;
default: System.out.println("Invalid choice! Try again.");
}
} while (choice != 6);
}
public static void main(String[] args) {
mainMenu();
}
```

OUTPUT:-



```
1. Create a new account
2. Deposit money
3. Withdraw money
4. View account details
5. Update contact details
6. Exit
Enter your choice: 1
Enter account holder name: Bhavya
Enter initial deposit amount: 5000
Enter email address: bhavya@example.com
Enter phone number: 9876543210
Account created successfully with Account Number: 1001
```

Explanation –

1. Account Class

The Account class is used to represent a **bank account**.
It contains **data members** (variables) to store account details and **methods** (functions) to perform operations.

- **Data Members**
  - accountNumber → a unique number for each account
  - accountHolderName → name of the account holder
  - balance → current money in the account
  - email, phoneNumber → contact details
- **Constructor**
  Initializes the account with given details when a new account object is created.
- **Methods**
  - deposit(amount) → increases balance if amount is valid
  - withdraw(amount) → decreases balance if funds are sufficient
  - displayAccountDetails() → shows all account information

  - updateContactDetails(email, phone) → updates email and phone number
  - getAccountNumber() → returns account number (used for searching)

Thus, the Account class represents **one customer’s account** and all the actions that can be done on it.

---

2. UserInterface Class

This class manages the interaction between the **user** and the **Account objects**.
It acts like a **banking system menu**.

- **Static Variables**
  - accounts[] → array to store multiple Account objects (up to 100 accounts)
  - accountCount → keeps track of number of accounts created
  - nextAccountNumber → generates unique account numbers starting from 1001
  - Scanner sc → for taking input from the user
- **Methods**
  - createAccount() → takes user details, creates a new Account, and stores it in the array
  - findAccount(accountNumber) → searches for an account using its account number
  - performDeposit() → deposits money into a given account
  - performWithdrawal() → withdraws money from a given account
  - showAccountDetails() → displays details of a given account
  - updateContact() → updates email and phone of a given account
  - mainMenu() → displays the menu repeatedly, allowing users to choose actions until they exit

---

3. Main Method

The main() method simply calls mainMenu() to start the program.
This displays the banking menu and allows the user to perform actions like:

1. Create new account

2. Deposit money
3. Withdraw money
4. View account details
5. Update contact details
6. Exit

Working Principle

1. When the program runs, the user is shown a menu.
2. Based on the choice entered, the corresponding function is executed.
3. Each account is stored in the accounts array, and users can access their account using their **account number**.
4. The program runs continuously in a loop until the user selects **Exit**.

Concepts Used

- **Encapsulation** → account details are private and accessed via methods
- **Object-Oriented Programming** → use of classes (Account) and objects (individual accounts)
- **Array of Objects** → multiple accounts are stored in an array
- **Control Structures** → switch-case, if-else, do-while loop for menu handling
- **Scanner Class** → for user input