OOPS WITH JAVA

CS23333

MINI PROJECT

SIMPLE BANKING APPLICATION

DONE BY:

NAME: A.SHARUKH AKTHAR

ROLL NUMBER: 231401096

CLASS: CSBS-B

AIM:

The purpose of this simple banking application is to allow users to manage accounts, perform deposits, withdrawals, and check balances. It demonstrates the use of Object-Oriented Programming (OOP) principles, including encapsulation and collections, to simplify banking operations.

ALGORITHM:

1. Start.
2. Display the main menu with the following options:

- Create Account
- Deposit Money
- Withdraw Money
- Check Balance
- Exit

3. Accept the user's choice.
4. Perform the corresponding operation:

- Create Account:
    1. Input account holder's name and initial deposit.
    2. Create a new account object.
    3. Add the account to the collection.
- Deposit Money:
    1. Input account number and deposit amount.
    2. Search for the account in the collection.
    3. Add the deposit amount to the balance.
- Withdraw Money:
    1. Input account number and withdrawal amount.
    2. Search for the account.
    3. Deduct the amount from the balance if sufficient funds are available.
- Check Balance:
    1. Input account number.
    2. Display the account balance if found.
- Exit: Terminate the program.

5. Repeat until the user chooses Exit.
6. End.

PROGRAM:

```
import java.util.*;

class Account {
    private static int accountCounter = 1000; // Auto-increment for account numbers
    private int accountNumber;
    private String holderName;
    private double balance;

    public Account(String holderName, double initialDeposit) {
        this.accountNumber = accountCounter++;
        this.holderName = holderName;
        this.balance = initialDeposit;
    }

    public int getAccountNumber() {
        return accountNumber;
    }

    public String getHolderName() {
        return holderName;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        balance += amount;
    }
```

```
    public boolean withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            return true;
        }
        return false;
    }

    @Override
    public String toString() {
        return "Account Number: " + accountNumber +
               ", Holder Name: " + holderName +
               ", Balance: $" + balance;
    }
}

class BankManager {
    private Map<Integer, Account> accounts = new HashMap<>();

    public void createAccount(String holderName, double initialDeposit) {
        Account account = new Account(holderName, initialDeposit);
        accounts.put(account.getAccountNumber(), account);
        System.out.println("Account created successfully!");
        System.out.println("Your Account Number is: " + account.getAccountNumber());
    }

    public Account getAccount(int accountNumber) {
        return accounts.get(accountNumber);
    }

    public Account getAccountByName(String holderName) {
```

```
        for (Account account : accounts.values()) {
            if (account.getHolderName().equalsIgnoreCase(holderName)) {
                return account;
            }
        }
        return null;
    }

    public void viewAccounts() {
        if (accounts.isEmpty()) {
            System.out.println("No accounts available.");
        } else {
            for (Account account : accounts.values()) {
                System.out.println(account);
            }
        }
    }
}

public class BankingApplication {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        BankManager bankManager = new BankManager();

        while (true) {
            System.out.println("\nSimple Banking Application");
            System.out.println("1. Create Account");
            System.out.println("2. Deposit Money");
            System.out.println("3. Withdraw Money");
            System.out.println("4. Check Balance");
            System.out.println("5. View All Accounts");
```

```
System.out.println("6. Search Account by Name");
System.out.println("7. Exit");
System.out.print("Enter your choice: ");
int choice = scanner.nextInt();
scanner.nextLine(); // Consume newline

switch (choice) {
    case 1:
        System.out.print("Enter Holder Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Initial Deposit: ");
        double deposit = scanner.nextDouble();
        bankManager.createAccount(name, deposit);
        break;

    case 2:
        System.out.print("Enter Account Number: ");
        int depositAcc = scanner.nextInt();
        System.out.print("Enter Amount to Deposit: ");
        double amount = scanner.nextDouble();
        Account depositAccount = bankManager.getAccount(depositAcc);
        if (depositAccount != null) {
            depositAccount.deposit(amount);
            System.out.println("Deposit successful!");
        } else {
            System.out.println("Account not found.");
        }
        break;

    case 3:
        System.out.print("Enter Account Number: ");
```

```
            int withdrawAcc = scanner.nextInt();
            System.out.print("Enter Amount to Withdraw: ");
            double withdrawAmount = scanner.nextDouble();
            Account withdrawAccount = bankManager.getAccount(withdrawAcc);
            if (withdrawAccount != null) {
                if (withdrawAccount.withdraw(withdrawAmount)) {
                    System.out.println("Withdrawal successful!");
                } else {
                    System.out.println("Insufficient balance.");
                }
            } else {
                System.out.println("Account not found.");
            }
            break;

        case 4:
            System.out.print("Enter Account Number: ");
            int balanceAcc = scanner.nextInt();
            Account balanceAccount = bankManager.getAccount(balanceAcc);
            if (balanceAccount != null) {
                System.out.println("Account Balance: $" + balanceAccount.getBalance());
            } else {
                System.out.println("Account not found.");
            }
            break;

        case 5:
            bankManager.viewAccounts();
            break;

        case 6:
```

```
                System.out.print("Enter Holder Name: ");
                String searchName = scanner.nextLine();
                Account searchedAccount = bankManager.getAccountByName(searchName);
                if (searchedAccount != null) {
                    System.out.println(searchedAccount);
                } else {
                    System.out.println("Account not found.");
                }
                break;

            case 7:
                System.out.println("Exiting...");
                scanner.close();
                return;

            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
  }
}
```

OUTPUT:

```
Simple Banking Application
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. View All Accounts
6. Search Account by Name
7. Exit
Enter your choice: 1
Enter Holder Name: Sharukh
Enter Initial Deposit: 1500
Account created successfully!
Your Account Number is: 1000
```

```
Simple Banking Application
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. View All Accounts
6. Search Account by Name
7. Exit
Enter your choice: 2
Enter Account Number: 1000
Enter Amount to Deposit: 500
Deposit successful!
```

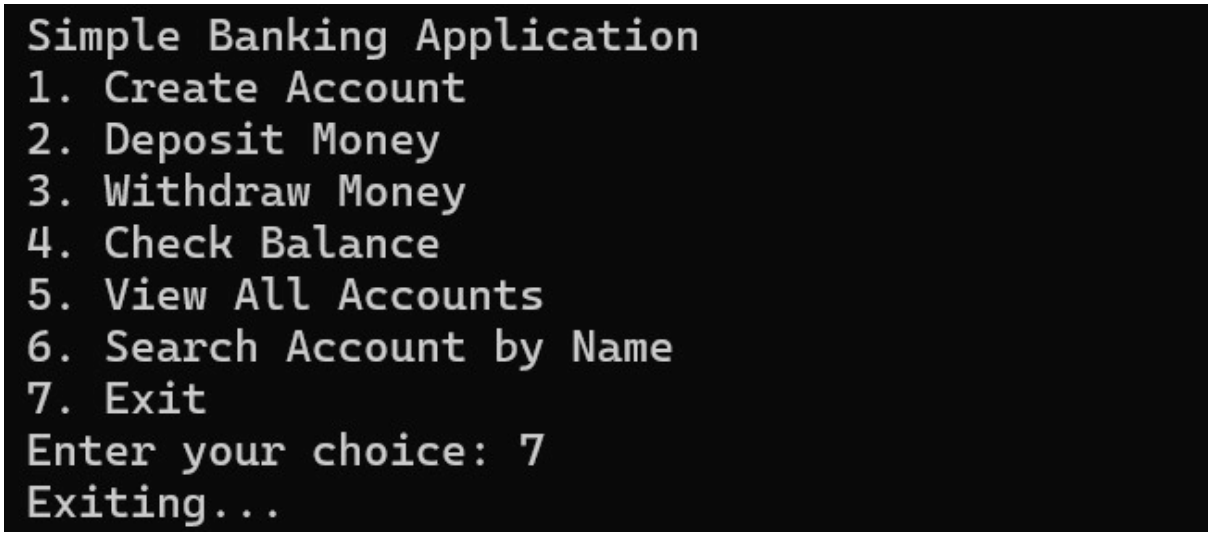
```
Simple Banking Application
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. View All Accounts
6. Search Account by Name
7. Exit
Enter your choice: 3
Enter Account Number: 1000
Enter Amount to Withdraw: 1500
Withdrawal successful!
```

```
Simple Banking Application
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. View All Accounts
6. Search Account by Name
7. Exit
Enter your choice: 4
Enter Account Number: 1000
Account Balance: $500.0
```

```
Simple Banking Application
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. View All Accounts
6. Search Account by Name
7. Exit
Enter your choice: 1
Enter Holder Name: Naveen
Enter Initial Deposit: 5000
Account created successfully!
Your Account Number is: 1001
```

```
Simple Banking Application
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. View All Accounts
6. Search Account by Name
7. Exit
Enter your choice: 5
Account Number: 1000, Holder Name: Sharukh, Balance: $500.0
Account Number: 1001, Holder Name: Naveen, Balance: $5000.0
```

```
Simple Banking Application
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. View All Accounts
6. Search Account by Name
7. Exit
Enter your choice: 6
Enter Holder Name: Naveen
Account Number: 1001, Holder Name: Naveen, Balance: $5000.0
```



CONCLUSION:

The **Simple Banking Application** successfully demonstrates the practical use of Object-Oriented Programming (OOP) concepts, such as encapsulation, inheritance, and collections, to solve real-world problems in an efficient and user-friendly way. This project emphasizes the importance of using structured algorithms, clear code organization, and intuitive user interactions to achieve a reliable and maintainable software solution.