Day3 Assignment:23-07-25

```
package Day3assign;
public interface BankOperations {
		void deposit(double amount);
		void withdraw(double amount);
		void transfer(Account target, double amount);
		double checkBalance();
		void showTransactionHistiory();
		}
}
```

SavingsAccount(extendsAccount,implements,BankOperati ons):

```
package Day3assign;
import java.util.ArrayList;
import java.util.List;



public  abstract class Account   implements BankOperations {
		protected String accountNumber;
		protected double balance;
		protected List<String> transactionHistory = new ArrayList<>();
		public Account(String accountNumber, double initialBalance) {
		this.accountNumber = accountNumber;
		this.balance = initialBalance;
		}
		public void transfer(Account target, double amount) {
			if (balance >= amount) {
			this.withdraw(amount);
			target.deposit(amount);
			addTransaction("Transferred to Account " + target.accountNumber + ": ₹"+ amount);
			target.addTransaction("Received from Account " + this.accountNumber+ ": ₹" +
amount);
```

```
                }
                else {
                System.out.println(" Insufficient balance to transfer ₹" + amount);
                }
        }
        public double checkBalance() {
                return balance;
        }
        public void addTransaction(String info) {
                transactionHistory.add(info);
                }
                public void showTransactionHistory() {
                System.out.println(" Transaction History for Account: " +
                accountNumber);
                for (String t : transactionHistory) {
                System.out.println(" - "+t);
                }
                }
                public abstract void deposit(double amount);
                public abstract void withdraw(double amount);
}
```

SavingsAccount

```
package Day3assign;

public class SavingsAccount  extends Account{
        private final double MIN_BALANCE = 1000.0;

        public SavingsAccount(String accNum, double balance) {
                super(accNum,balance);
        }
        public void deposit(double amount) {
```

```
                balance += amount;
                addTransaction("Deposited: ₹" + amount);
        }
        public void withdraw(double amount) {
                if (balance - amount >= MIN_BALANCE) {
                balance -= amount;
                addTransaction("Withdrawn: ₹" + amount);
                }
                else {
                System.out.println(" Cannot withdraw. Minimum ₹1000 must be kept");
                }
        }

}
```

CurrentAccount(extendsAccount,implements,BankOperat ions):

```
package Day3assign;

public class CurrentAccount  extends Account{
        private final double OVERDRAFT_LIMIT = 2000.0;
         public CurrentAccount(String accNum, double balance) {
                super(accNum, balance);
        }
        public void deposit(double amount) {
                balance += amount;
                addTransaction("Deposited: ₹" + amount);
        }
        public void withdraw(double amount) {
                if (balance - amount >= -OVERDRAFT_LIMIT) {
                                balance -= amount;
```

```
                                        addTransaction("Withdrawn: ₹" + amount);
                        }
                        else {
                                        System.out.println(" Cannot withdraw. Overdraft ₹2000
exceded");
                        }
        }

}
```

Customer Account:

```
package Day3assign;
import java.util.ArrayList;
import java.util.List;

public class CustomerAccount {
        private String customerId;
         private String name;
         private List<Account> accounts = new ArrayList<>();
         public CustomerAccount(String id, String name) {
                this.customerId = id;
                this.name=name;
         }
         public void addAccount(Account acc) {
                accounts.add(acc);
                }
                public List<Account> getAccounts() {
                return accounts;
                }
                public String getCustomerId() {
```

```
                return customerId;
                }
                public String getName() {
                return name;
                }


        }
```

Bank Branch:

```
package Day3assign;
import java.util.ArrayList;
import java.util.List;
public class BankBranch {
        private String branchId;
        private String branchName;
        private List<CustomerAccount> customers = new ArrayList<>();
        public BankBranch(String id, String name) {
                this.branchId = id;
                this.branchName = name;
                System.out.println("Branch Created:"+name+"[Branch
ID:"+id+"]");
        }
        public void addCustomer(CustomerAccount c) {
                customers.add(c);
                System.out.println(" Customer added to branch.");
        }
        public CustomerAccount findCustomerById(String id) {
                for (CustomerAccount c : customers) {
```

```
if (c.getCustomerId().equals(id)) return c;
                                        }
                                        return null;
                        }
                        public void listAllCustomers() {
                                System.out.println(" Customers in Branch:");
                                for (CustomerAccount c : customers) {
                                        System.out.println("- " + c.getName() + " [ID:c.getCustomerId()
+ "]");
                 }


                }
 }
```

Main:

```
package Day3assign;
import java.util.ArrayList;
import java.util.List;
public class BankBranch {
                        private String branchId;
                        private String branchName;
                        private List<CustomerAccount> customers = new ArrayList<>();
                        public BankBranch(String id, String name) {
                                        this.branchId = id;
                                        this.branchName = name;
                                        System.out.println("Branch Created:"+name+"[Branch
ID:"+id+"]");
                        }
                        public void addCustomer(CustomerAccount c) {
                                customers.add(c);
                                System.out.println(" Customer added to branch.");
```

```
        }
        public CustomerAccount findCustomerById(String id) {
                for (CustomerAccount c : customers) {
                        if (c.getCustomerId().equals(id)) return c;
                }
                return null;
        }
        public void listAllCustomers() {
                System.out.println(" Customers in Branch:");
                for (CustomerAccount c : customers) {
                        System.out.println("- " + c.getName() + " [ID: " +
c.getCustomerId() + "]");
        }

    }
}
```