BankOperations (Interface)

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

Account (Abstract Class)

```
package Oops;
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(extendsAccount,implements,BankOperations)

```
package Oops;
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            addTransaction("Withdrawn: ₹" + amount);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,BankOperations)

```
package Oops;
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 limit ₹2000
exceeded.");
      }
   }
}
```

Customer Account

```
package Oops;
import java.util.ArrayList;
import java.util.List;
public class Customer {
   private String customerId;
   private String name;
   private List<Account> accounts = new ArrayList<>();

   public Customer(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;
    }
}
```

BankBranch

```
package Oops;
import java.util.ArrayList;
import java.util.List;
public class BankBranch {
    private String branchId;
    private String branchName;
    private List<Customer> 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(Customer c) {
        customers.add(c);
        System.out.println("✅ Customer added to branch.");
    }
    public Customer findCustomerById(String id) {
        for (Customer c : customers) {
            if (c.getCustomerId().equals(id)) return c;
        }
        return null;
    }
    public void listAllCustomers() {
        System.out.println("👥 Customers in Branch:");
        for (Customer c : customers) {
            System.out.println("- " + c.getName() + " [ID: " + c.getCustomerId() +
"]");
        }
    }
}
```

Main

```
package Oops;

public class Main {
    public static void main(String[] args) {
        BankBranch branch = new BankBranch("B001", "Main Branch");
        Customer c1 = new Customer("C001", "Alice");

        branch.addCustomer(c1);

        SavingsAccount sa = new SavingsAccount("S001", 5000);
        CurrentAccount ca = new CurrentAccount("C001", 2000);

        c1.addAccount(sa);
        c1.addAccount(ca);

        sa.deposit(2000);
        ca.withdraw(2500);
        sa.transfer(ca, 1000);

        sa.showTransactionHistory();
        ca.showTransactionHistory();
    }
}
```

Output:

```
✅ Branch Created: Main Branch [Branch ID: B001]
✅ Customer added to branch.
📋 Transaction History for Account: S001
 - Deposited: ₹2000.0
 - Withdrawn: ₹1000.0
 - Transferred to Account C001: ₹1000.0
📋 Transaction History for Account: C001
 - Withdrawn: ₹2500.0
 - Deposited: ₹1000.0
 - Received from Account S001: ₹1000.0
```