BANKING SYSTEM OOPS CONCEPT

1. BankOperations.java

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

2. Account.java

```
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 abstract void deposit(double amount);
    public abstract void withdraw(double amount);

    public void transfer(Account target, double amount) {
        this.withdraw(amount);
        target.deposit(amount);
        addTransaction("Transferred " + amount + " to Account " + target.accountNumber);
        target.addTransaction("Received " + amount + " from Account " + this.accountNumber);
    }

    public double checkBalance() {
        return balance;
    }

    public void addTransaction(String info) {
        transactionHistory.add(info);  }
    public void showTransactionHistory() {
        System.out.println("Transaction History for " + accountNumber + ":");
        for (String t : transactionHistory) {
            System.out.println("- " + t);
        }
    }
}
```

3. SavingsAccount.java

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

    public SavingsAccount(String accNo, double initialBalance) {
        this.accountNumber = accNo;
        this.balance = initialBalance;
        addTransaction("Account opened with " + initialBalance);
    }

    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            addTransaction("Withdrawn " + amount);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 " + amount + ". Minimum balance " +
MIN_BALANCE + " required.");
        }
    }
}
```

4. CurrentAccount.java

```
public class CurrentAccount extends Account implements BankOperations {
    private final double OVERDRAFT_LIMIT = 2000.0;

    public CurrentAccount(String accNo, double initialBalance) {
        this.accountNumber = accNo;
        this.balance = initialBalance;
        addTransaction("Account opened with " + initialBalance);
    }

    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 " + amount + ". Overdraft limit " +
OVERDRAFT_LIMIT + " exceeded.");
        }
    }
}
```

5. Customer.java

```
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 customerId, String name) {
        this.customerId = customerId;
        this.name = name;
        System.out.println("Customer Created: " + name + " [Customer ID: " + customerId + "]");
    }

    public void addAccount(Account acc) {
        accounts.add(acc);
    }

    public List<Account> getAccounts() {
        return accounts;
    }

    public String getCustomerId() {
        return customerId;
    }

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

6. BankBranch.java

```
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 branchId, String branchName) {
        this.branchId = branchId;
        this.branchName = branchName;
        System.out.println("Branch Created: " + branchName + " [Branch ID: " + branchId + "]");
    }

    public void addCustomer(Customer c) {
        customers.add(c);
        System.out.println("Customer added to branch: " + c.getName());
    }

    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() + "]");
        }
    }
}
```

7. Main.java

```
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 savings = new SavingsAccount("S001", 5000.0);
        CurrentAccount current = new CurrentAccount("C002", 2000.0);

        c1.addAccount(savings);
        c1.addAccount(current);

        savings.deposit(2000.0);
        current.withdraw(2500.0);
        savings.transfer(current, 1000.0);

        System.out.println("Savings Account Balance: " + savings.checkBalance());
        System.out.println("Current Account Balance: " + current.checkBalance());

        savings.showTransactionHistory();
        current.showTransactionHistory();
    }

}
```