Core Java Case Study:

Objective: Design and implement the system using Java that demonstrates Object-Oriented Programming (OOP) concepts, including inheritance, polymorphism, and collections.

1. **Banking System**

The Banking System must efficiently manage Account Details, Customer Details, and Transactions. Users should be able to log in to the system and perform banking operations like balance check, deposit, withdraw and money transfer. Banking System performs the functions of Account Management and Transaction Processing.

Entities:

- Customer
- Account
- Transaction
- Beneficiary

Relationships Between These Entities

Customer – Account relationship

- One to Many Relationship: One customer is allowed to create many accounts.

Customer – Beneficiary Relationship

- One to Many Relationship: A customer can add multiple beneficiaries.

Account – Transaction Relationship

- Many to Many Relationship: An account can have multiple transactions and a transaction can involve multiple account numbers.

Design following class structures for Framework

1. Customer

Fields	**Access**	**Type**	**Property**
customerID	private	int	Read-Write
name	private	String	Read-Write
address	private	String	Read-Write
contact	private	String	Read-Write

Constructor	**Access**	**Parameters**
	Public	customerID, name, address, contact

Methods	**Access**	**Return Type**	**Parameters**	**Particulars**
getter and setters	public	Depends on data member	-	-
toString	public	String	-	Overridden

2. Account

Fields	Access	Type	Property
accountID	private	int	Read-Write
customerID	private	int	Read-Write
type	private	String	Read-Write
balance	private	double	Read-Write

Constructor	Access	Parameters
	Public	accountID, customerID, type, balance

Methods	Access	Return Type	Parameters	Particulars
getter and setters	public	Depends on data member	-	-
toString	public	String	-	Overridden

3. Transaction

Fields	Access	Type	Property
transactionID	private	int	Read-Write
accountID	private	int	Read-Write
type	private	String	Read-Write
amount	private	double	Read-Write
timestamp	private	LocalDateTime	Read-Write

Constructor	Access	Parameters
	Public	accountID, type, amount

Methods	Access	Return Type	Parameters	Particulars
getter and setters	public	Depends on data member	-	-
toString	public	String	-	Overridden

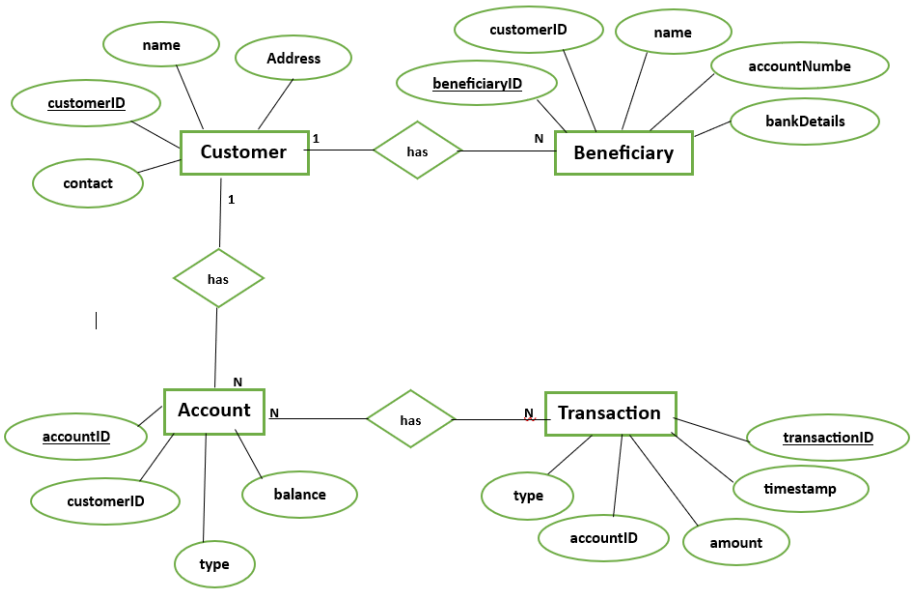
4. Beneficiary

Fields	Access	Type	Property
beneficiaryID	private	int	Read-Write
customerID	private	int	Read-Write
name	private	String	Read-Write
accountNumber	private	String	Read-Write
bankDetails	private	String	Read-Write

Constructor	Access	Parameters
	Public	beneficiaryID, customerID, name, accountNumber, bankDetails

Methods	Access	Return Type	Parameters	Particulars
getter and setters	public	Depends on data member	-	-
toString	public	String	-	Overridden

ER Diagram



Develop the banking system application:

Create service interfaces and their implementations to handle business logic and operations.

BankingService Interface

The BankingService interface will define methods for common banking operations, such as adding customers, creating accounts, processing transactions, etc.

```java
//Program to define an interface to provide Banking services
package com.tns.banking.services;

import java.util.Collection;

public interface BankingService {
    void addCustomer(Customer customer);
    void addAccount(Account account);
    void addTransaction(Transaction transaction);
    void addBeneficiary(Beneficiary beneficiary);

    Customer findCustomerById(int id);
    Account findAccountById(int id);
    Transaction findTransactionById(int id);
    Beneficiary findBeneficiaryById(int id);

    Collection<Account> getAllAccounts();
    Collection<Customer> getAllCustomers();
    Collection<Transaction> getAllTransactions();
    Collection<Beneficiary> getAllBeneficiaries();

    List<Account> getAccountsByCustomerId(int customerId);
    List<Transaction> getTransactionsByAccountId(int accountId);
    List<Beneficiary> getBeneficiariesByCustomerId(int customerId);
}
```

BankingServiceImpl class -

The BankingServiceImpl class will implement the methods defined in the BankingService interface. This class will manage the collections of entities and perform operations based on the requirements.

```
public class BankingServiceImple implements BankingService {

        private Map<Integer, Customer> customers = new HashMap<>();

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

        private Map<Integer, Transaction> transactions = new HashMap<>();

        private Map<Integer, Beneficiary> beneficiaries = new HashMap<>();

        @Override

        public void addCustomer(Customer customer) {

                // add Customer to customers map, Key - customerID

        }

        @Override

        public void addAccount(Account account) {

                // add Account to accounts map, Key - accountID

        }

        @Override
```

```
public void addTransaction(Transaction transaction) {
        // add transaction to transactions map, key - transactionID
        // and based on transaction type(deposit or withdraw)update the account balance
}
@Override
public void addBeneficiary(Beneficiary beneficiary) {
        // add beneficiary to beneficiaries map, key - beneficiaryID
}
@Override
public Customer findCustomerById(int id) {
        return customers.get(id);
}
@Override
public Account findAccountById(int id) {
        return accounts.get(id);
}
@Override
public Transaction findTransactionById(int id) {
        return transactions.get(id);
}
@Override
public Beneficiary findBeneficiaryById(int id) {
        return beneficiaries.get(id);
}
@Override
public List<Account> getAccountsByCustomerId(int customerId) {
        List<Account> result = new ArrayList<>();
        // retrieve accounts from accounts map of given customerID and add into result
        return result;
}
```

```
	@Override
	public List<Transaction> getTransactionsByAccountId(int accountId) {
		List<Transaction> result = new ArrayList<>();
		// retrieve transactions from transactions map of given accountID and add into result
		return result;
	}
	@Override
	public List<Beneficiary> getBeneficiariesByCustomerId(int customerId) {
		List<Beneficiary> result = new ArrayList<>();
		// retrieve beneficiaries from beneficiaries map of given customerID and add into
		//result
		return result;
	}
	@Override
	public Collection<Account> getAllAccounts() {
		return accounts.values();
	}
	@Override
	public Collection<Customer> getAllCustomers() {
		return customers.values();
	}
	@Override
	public Collection<Transaction> getAllTransactions() {
		return transactions.values();
	}
	@Override
	public Collection<Beneficiary> getAllBeneficiaries() {

		return beneficiaries.values();
	}
}
```

Create the Menu Driven Driver class BankingSystemApp to demonstrates how to use the BankingService to manage customers, accounts, transactions, and beneficiaries.

SAMPLE OUTPUT :

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

1

Enter Customer Details

Customer Id : 1

Name : Aniket

Address : Pune

Contact No. : 7868678899

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

1

Enter Customer Details

Customer Id : 2

Name : Nikhil

Address : Mumbai

Contact No. : 7878330331

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

2

Enter Account Details

Account Id : 101

Customer Id : 1

Account Type Saving/ Current : Saving

Balance : 70000

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

2

Enter Account Details

Account Id : 102

Customer Id : 2

Account Type Saving/ Current : Current

Balance : 80000

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

3

Enter Beneficiary Details

Customer Id : 1

Benefiary Id : 1001

Beneficiary Name : Mahesh

Beneficiary Account No. : 6722212

Beneficiary Bank details : SBI

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

5

Cutomer ID: 1, Name: Aniket

Cutomer ID: 2, Name: Nikhil

Customer Id : 1

Customer: Aniket

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

6

Account ID: 101, Customer ID : 1, Balance: 70000.0

Account ID: 102, Customer ID : 2, Balance: 80000.0

Customer Id : 1

Accounts for Customer ID :1

Account ID: 101, Balance: 70000.0

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

7

Account Id : 101

Transactions for Account ID :101

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

4

Enter Transaction Details

Account Id : 101

Tyep (Deposit/Withdrawal : Deposit

Account Amount : 10000

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

7

Account Id : 101

Transactions for Account ID :101

Transaction ID: 1, Type: Deposit, Amount: 10000.0, Timestamp: 2024-08-09T17:47:43.479473300

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

6

Account ID: 101, Customer ID : 1, Balance: 80000.0

Account ID: 102, Customer ID : 2, Balance: 80000.0

Customer Id : 101

Accounts for Customer ID :101

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

8

Beneficiary ID: 1, Name: Mahesh

Customer Id : 1

Beneficiaries for Customer ID :1

Beneficiary ID: 1, Name: Mahesh

Banking System

1. Add Customers

2. Add Accounts

3. Add Beneficiary

4. Add Transaction

5. Find Customer by Id

6. List all Accounts of specific Customer

7. List all transactions of specific Account

8. List all beneficiaries of specific customer

9. Exit

Enter your choice :

9

Thank you!