**Implement a Comprehensive Banking System**

**Objective:**

To design and implement a robust banking system using Java, following an object-oriented, layered architecture. The system should manage bank accounts, transactions, rewards, and email notifications. The students will use the provided class structure and implement functionality for CRUD operations using JDBC.

**Part 1: Class Definitions**

**1. Account Class**

Represents a bank account.

**Attributes:**

* Long accountNumber: Unique account identifier.
* String name: Account holder's name.
* boolean isActive: Account status (true if active, false otherwise).
* Set<Benificiary> benificiaries: List of beneficiaries for the account.
* Address address: Account holder's address.
* int balance: Current account balance.
* String emailAddress: Account holder's email address.

**2. Address Class**

Represents the address of the account holder.

**Attributes:**

* String city: City of the account holder.
* String country: Country of the account holder.

**3. Benificiary Class**

Represents a beneficiary associated with an account.

**Attributes:**

* Long ssn: Social Security Number of the beneficiary.
* String name: Name of the beneficiary.

**4. Reward Class**

Represents rewards or benefits associated with an account.

**Attributes:**

* Long rewardConfirmationNumber: Unique confirmation number for the reward.
* int rewardAmount: Amount of the reward.
* Long accountNumber: Account number associated with the reward.

**5. TransactionDetail Class**

Represents a transaction associated with an account.

**Attributes:**

* Long transactionId: Unique transaction identifier.
* Long accountNumber: Account number associated with the transaction.
* Date transactionDate: Date of the transaction.
* int amount: Transaction amount.
* TransactionType txType: Type of transaction (debit or credit).

**6. TransactionType Enum**

Enumerates transaction types.

**Attributes:**

* DEBIT: Represents a debit transaction.
* CREDIT: Represents a credit transaction.

**Part 2: Database Integration**

* **Set up a relational database** (e.g., MySQL, PostgreSQL) with tables for account, reward, and transactiondetail.
* Create schema scripts based on the class attributes.

**Part 3: Repository Layer**

Implement repositories to interact with the database. These classes will perform CRUD operations.

**1. AccountRepository Interface**

Defines the methods to manage accounts:

* Account findAccountByNumber(Long accountNumber): Fetch account by number.
* List<Account> findAllAccounts(): Retrieve all accounts.
* void save(Account account): Save a new account.
* void update(Account account): Update account details.
* void delete(Account account): Delete an account.

**2. JdbcAccountRepositoryImpl**

Implementation of AccountRepository using JDBC.

**Key Operations**:

* **Find Account**:
* **Save Account**:
* **Update Account**:
* **Delete Account**:

**3. RewardRepository Interface**

Defines methods to manage rewards:

* void addReward(Reward reward): Add a reward.
* int getTotalRewardAmount(Long accountNumber): Fetch the total rewards for an account.
* List<Reward> getAllRewardsForAccount(Long accountNumber): Fetch all rewards for an account.

**4. JdbcRewardRepositoryImpl**

Implements RewardRepository using JDBC.

**Key Operations**:

* **Add Reward**:

**5. TransactionRepository Interface**

Defines methods to manage transactions:

* Long addTransaction(TransactionDetail transactionDetail): Add a new transaction.
* List<TransactionDetail> getAllTransactionDetailsByAccountNumber(Long accountNumber): Fetch all transactions for an account.

**6. JdbcTransactionRepositoryImpl**

Implements TransactionRepository using JDBC.

**Key Operations**:

* **Add Transaction**:

**Part 4: Service Layer**

Implement the business logic for the banking system.

**1. BankService Interface**

Defines the following methods:

* Long transfer(Long fromAccount, Long toAccount, int amount): Transfer amount between accounts.
* Long debit(int amount, Long accountNumber): Debit amount from an account.
* Long credit(int amount, Long accountNumber): Credit amount to an account.
* void createNewAccount(Account account): Create a new account.
* void deactivateAccount(Long accountNumber): Deactivate an account.
* void activateAccount(Long accountNumber): Activate an account.
* List<Account> getAllAccounts(): Fetch all accounts.

**2. BankServiceImpl**

Implements BankService and uses repositories for operations.

**Key Features**:

* Transactions (debit, credit, transfer).
* Email notifications for every transaction.
* Manage account activation/deactivation.

**Part 5: Email Notification**

Implement an EmailService to simulate email notifications:

* sendMail(String toAddress, String fromAddress, String content): Print a message to the console to simulate email sending.

**Part 6: Main Class**

Write a Main class to test the application.

**Tasks**:

1. Create new accounts.
2. Perform transactions (debit, credit, transfer).
3. Add rewards and fetch reward details.
4. Activate/deactivate accounts.
5. Fetch and display all account and transaction details.