Banking System - PlantUML Codes

ER Diagram

```
@startuml
entity "Customer" as Customer
entity "Account" as Account
entity "Transaction" as Transaction
entity "AuditLog" as AuditLog

Customer ||--o{ Account : owns
Account ||--o{ Transaction : "has"
Transaction ||--o{ AuditLog : "generates"
@enduml
```

Class Diagram

```
@startuml
class Customer {
    - id: String
    - fullName: String
    - email: String
    - password: String
    - phone: String
}
class Account {
    - id: String
    - customerId: String
    - number: String
    - balance: BigDecimal
    - status: String
    - type: String
}
class Transaction {
    - id: String
    - fromAccountId: String
    - toAccountId: String
    - type: String
    - amount: BigDecimal
    - status: String
}
class AuditLog {
    - id: String
    - txnId: String
    - accountId: String
    - actor: String
    - action: String
    - beforeBalance: BigDecimal
    - afterBalance: BigDecimal
}
Customer "1" -- "many" Account
Account "1" -- "many" Transaction
Transaction "1" -- "many" AuditLog
@enduml
```

Use Case Diagram

```
@startuml
actor Customer
Customer --> (Register)
Customer --> (Login)
Customer --> (Create Account)
Customer --> (View Balance)
Customer --> (Withdraw Money)
Customer --> (Deposit Money)
Customer --> (Transfer Money)
Customer --> (Transaction History)
Customer --> (Close Account)
Customer --> (Change Password)
@enduml
```

Activity Diagram - Register

```
@startuml
start
:Enter name, email, password, phone;
:Validate input;
if (Email already exists?) then (yes)
    :Show error;
    stop
else (no)
    :Create customer record;
    :Show success message;
endif
stop
@enduml
```

Activity Diagram - Login

```
@startuml
start
:Enter email and password;
:Validate credentials;
if (Valid?) then (yes)
    :Login success;
else (no)
    :Show error message;
endif
stop
@enduml
```

Activity Diagram - Create Account

```
@startuml
start
:Enter account type and initial deposit;
:Validate deposit > 0;
if (Valid?) then (yes)
    :Generate account;
    :Show account number;
else (no)
    :Show error;
endif
stop
@enduml
```

Activity Diagram - View Balance

```
@startuml
start
:Select account;
:Fetch balance;
:Display balance;
stop
@enduml
```

Activity Diagram - Withdraw Money

```
@startuml
start
:Enter amount;
:Check balance >= amount;
```

```
if (Sufficient?) then (yes)
    :Deduct amount;
    :Update balance;
    :Generate audit log;
    :Show success;
else (no)
    :Show insufficient funds;
endif
stop
@enduml
```

Activity Diagram - Deposit Money

```
@startuml
start
:Enter deposit amount;
:Add to balance;
:Update account;
:Generate audit log;
:Show success;
stop
@enduml
```

Activity Diagram - Transfer Money

```
@startuml
start
:Enter source, destination, amount;
:Check source balance >= amount;
if (Sufficient?) then (yes)
    :Deduct from source;
    :Credit destination;
    :Update balances;
    :Generate audit logs;
    :Show success;
else (no)
    :Show error;
endif
stop
@enduml
```

Activity Diagram - Transaction History

```
@startuml
start
:Select account;
:Fetch transactions;
:Display history;
stop
@enduml
```

Activity Diagram - Close Account

```
@startuml
start
:Select account;
:Check account status;
if (Active?) then (yes)
    :Set status CLOSED;
    :Generate audit log;
    :Show success;
else (no)
    :Show already closed;
endif
stop
@enduml
```

Activity Diagram - Change Password

```
@startuml
start
:Enter old password, new password;
:Validate old password;
if (Correct?) then (yes)
    :Update password;
    :Show success;
else (no)
    :Show error;
endif
stop
@enduml
```

Sequence Diagram - Register

```
@startuml
actor Customer
Customer -> AuthService : register(fullName, email, password, phone)
AuthService -> CustomerRepository : save(customer)
CustomerRepository --> AuthService : Customer saved
AuthService --> Customer : success
@enduml
```

Sequence Diagram - Login

```
@startuml
actor Customer
Customer -> AuthService : login(email, password)
AuthService -> CustomerRepository : findByEmail(email)
CustomerRepository --> AuthService : Customer/Empty
AuthService --> Customer : success/error
@enduml
```

Sequence Diagram - Create Account

```
@startuml
actor Customer
Customer -> AccountService : createAccount(customerId, deposit, type)
AccountService -> AccountRepository : save(account)
AccountRepository --> AccountService : account created
AccountService --> Customer : account number
@enduml
```

Sequence Diagram - View Balance

```
@startuml
actor Customer
Customer -> AccountService : viewBalance(accountNumber)
AccountService -> AccountRepository : findByNumber(accountNumber)
AccountRepository --> AccountService : Account
AccountService --> Customer : balance
@enduml
```

Sequence Diagram - Withdraw Money

```
@startuml
actor Customer
Customer -> TransactionService : withdraw(accountNumber, amount)
TransactionService -> AccountRepository : findByNumber(accountNumber)
AccountRepository --> TransactionService : Account
TransactionService -> TransactionRepository : save(transaction)
TransactionService -> AuditLogRepository : save(auditLog)
TransactionService --> Customer : success/error
@enduml
```

Sequence Diagram - Deposit Money

```
@startuml
actor Customer
Customer -> TransactionService : deposit(accountNumber, amount)
TransactionService -> AccountRepository : findByNumber(accountNumber)
AccountRepository --> TransactionService : Account
TransactionService -> TransactionRepository : save(transaction)
TransactionService -> AuditLogRepository : save(auditLog)
TransactionService --> Customer : success
@enduml
```

Sequence Diagram - Transfer Money

```
@startuml
actor Customer
Customer -> TransactionService : transfer(srcAccount, destAccount, amount)
TransactionService -> AccountRepository : findByNumber(srcAccount)
AccountRepository --> TransactionService : Source Account
TransactionService -> AccountRepository : findByNumber(destAccount)
AccountRepository --> TransactionService : Destination Account
TransactionService -> TransactionRepository : save(transaction)
TransactionService -> AuditLogRepository : save(auditLog)
TransactionService --> Customer : success/error
@enduml
```

Sequence Diagram - Transaction History

```
@startuml
actor Customer
Customer -> TransactionService : getHistory(accountNumber)
TransactionService -> TransactionRepository : findByAccountId(accountId)
TransactionRepository --> TransactionService : transactions list
TransactionService --> Customer : transactions list
@enduml
```

Sequence Diagram - Close Account

```
@startuml
actor Customer
Customer -> AccountService : closeAccount(accountNumber)
AccountService -> AccountRepository : findByNumber(accountNumber)
AccountRepository --> AccountService : Account
AccountService -> AccountRepository : save(account with CLOSED)
AccountService -> AuditLogRepository : save(auditLog)
AccountService --> Customer : success/error
@enduml
```