# **Banking Transaction System – Day 2**

**Prepared By:** Arun kumar R  
**Date:** [9/8/2025]  
**Project Name:** Banking Transaction System

# **Contents**

1. Customer Diagram
2. Account Diagram
3. Transaction Diagram

## **1. Customer Diagram**

### **Description:**

The Customer Diagram shows all relevant attributes of a bank customer and their relationships with accounts. It is typically represented as a **Class Diagram**.

### **Customer Attributes:**

* CustomerID (Primary Key)
* Name
* DateOfBirth
* Address
* Email
* PhoneNumber
* Password (secured)

### **Relationships:**

* A customer **can have multiple accounts** (1-to-many relationship).

### **Sample Diagram (Text Representation):**

Customer

----------------------

CustomerID : int

Name : string

DateOfBirth : date

Address : string

Email : string

PhoneNumber : string

Password : string

----------------------

1 Customer ----> \* Account

**Customer UML Class Diagram**

+------------------------------+

| Customer |

+------------------------------+

| - CustomerID : int |

| - Name : string |

| - DateOfBirth : date |

| - Address : string |

| - Email : string |

| - PhoneNumber : string |

| - Password : string (secured)|

+------------------------------+

| + getCustomerDetails() |

| + updateCustomerInfo() |

| + validatePassword() |

+------------------------------+

1

|

| 1-to-many

|

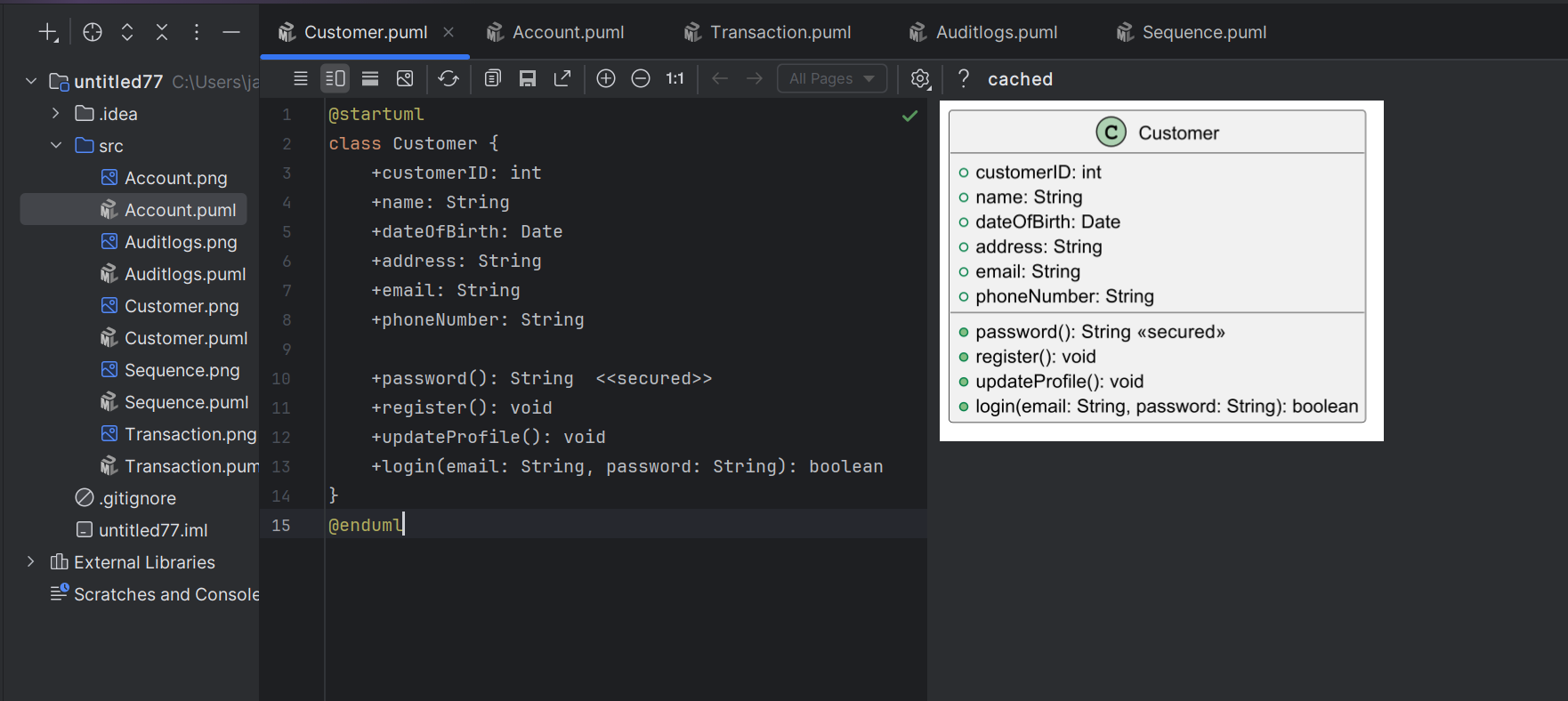
\*

+------------------------------+

| Account |

+------------------------------+

@startuml  
class Customer {  
 +customerID: int  
 +name: String  
 +dateOfBirth: Date  
 +address: String  
 +email: String  
 +phoneNumber: String  
  
 +password(): String <<secured>>  
 +register(): void  
 +updateProfile(): void  
 +login(email: String, password: String): boolean  
}  
@enduml



## **2. Account Diagram**

### **Description:**

The Account Diagram shows details of bank accounts and their association with customers and transactions.

### **Account Attributes:**

* AccountNumber (Primary Key)
* AccountType (Savings / Current)
* Balance
* BranchName
* CustomerID (Foreign Key)

### **Relationships:**

* An account **belongs to one customer**.
* An account **can have multiple transactions** (1-to-many relationship).

### **Sample Diagram (Text Representation):**

Account

----------------------

AccountNumber : int

AccountType : string

Balance : float

BranchName : string

CustomerID : int

----------------------

1 Account ----> \* Transaction

### **Account UML Class Diagram**

+------------------------------+

| Account |

+------------------------------+

| - AccountNumber : int |

| - AccountType : string |

| - Balance : float |

| - BranchName : string |

| - CustomerID : int |

+------------------------------+

| + deposit(amount: float) |

| + withdraw(amount: float) |

| + getBalance() : float |

| + updateAccountInfo() |

+------------------------------+

1

|

| 1-to-many

|

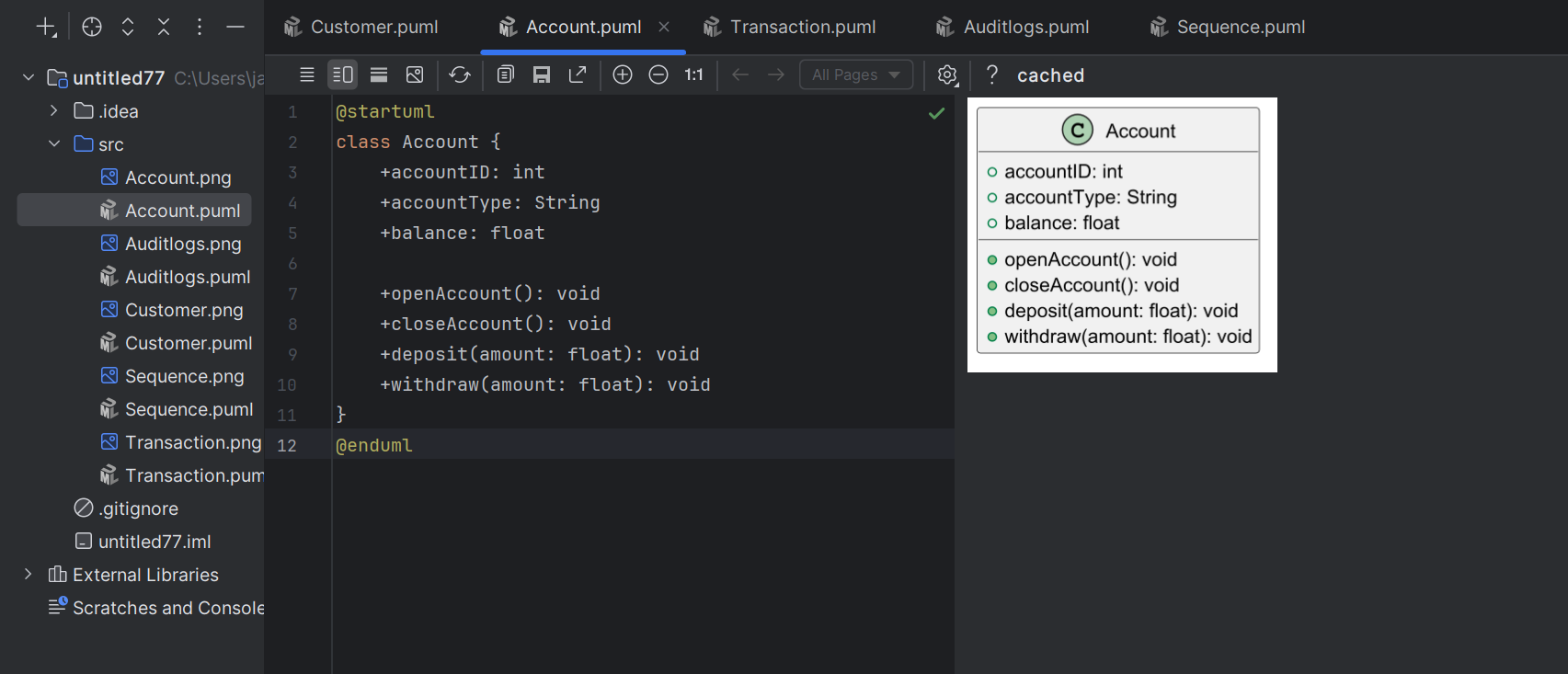
\*

+------------------------------+

| Transaction |

+------------------------------+

@startuml  
class Account {  
 +accountID: int  
 +accountType: String  
 +balance: float  
  
 +openAccount(): void  
 +closeAccount(): void  
 +deposit(amount: float): void  
 +withdraw(amount: float): void  
}  
@enduml



**3. Transaction Diagram**

### **Description:**

The Transaction Diagram represents money transfers, deposits, and withdrawals, linked to accounts.

### **Transaction Attributes:**

* TransactionID (Primary Key)
* Date
* Amount
* TransactionType (Deposit / Withdrawal / Transfer)
* SourceAccountNumber (FK)
* DestinationAccountNumber (FK, optional for deposits/withdrawals)
* Status (Success / Failed)

### **Relationships:**

* A transaction **is associated with one account** (or two for transfers).

### **Sample Diagram (Text Representation):**

Transaction

----------------------

TransactionID : int

Date : datetime

Amount : float

TransactionType : string

SourceAccountNumber : int

DestinationAccountNumber : int

Status : string

----------------------

\* Transaction <---- 1 Account

### **Transaction UML Class Diagram**

+-------------------------------------+

| Transaction |

+-------------------------------------+

| - TransactionID : int |

| - Date : datetime |

| - Amount : float |

| - TransactionType : string |

| - SourceAccountNumber : int |

| - DestinationAccountNumber : int |

| - Status : string |

+-------------------------------------+

| + processTransaction() |

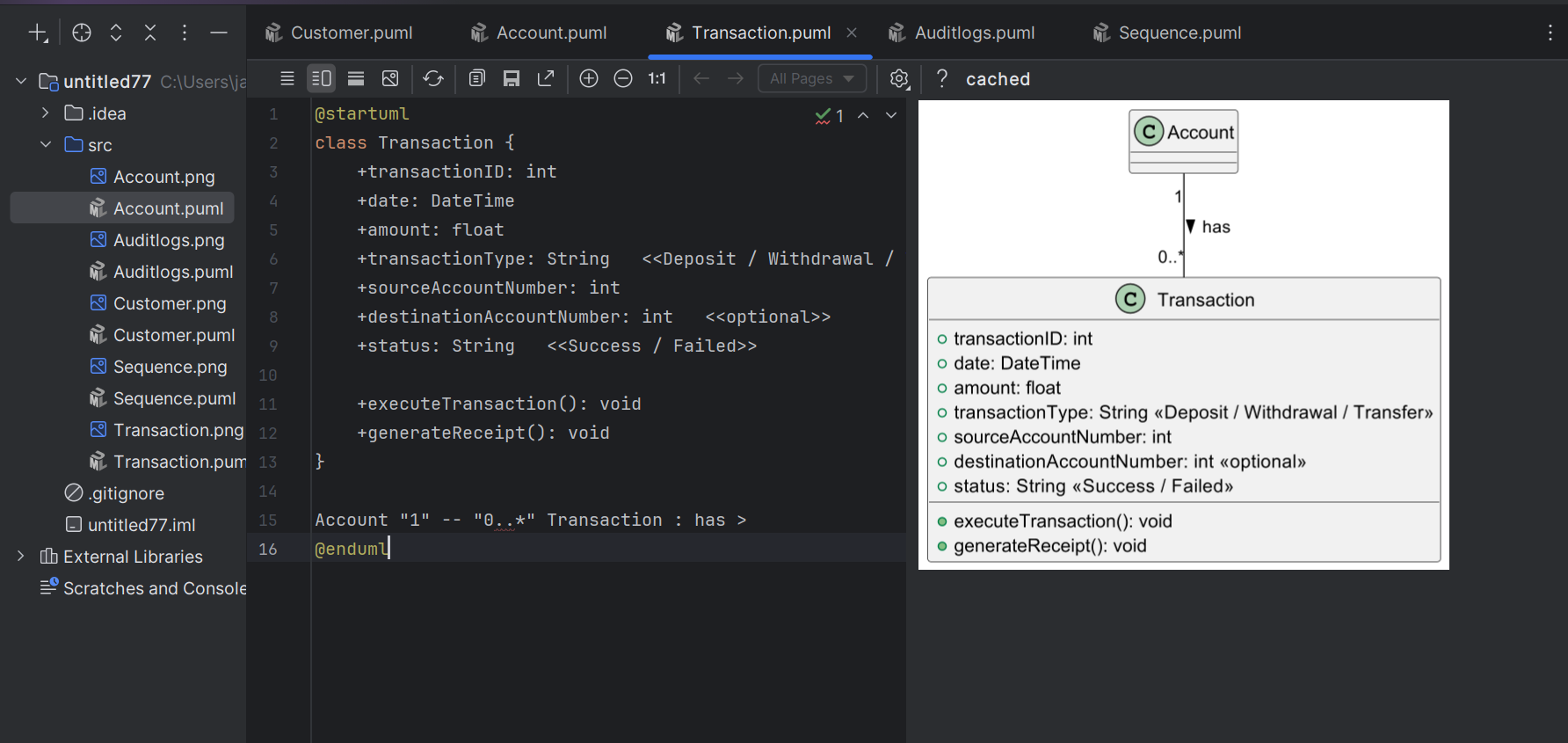
| + validateTransaction() |

| + getTransactionDetails() |

+-------------------------------------+

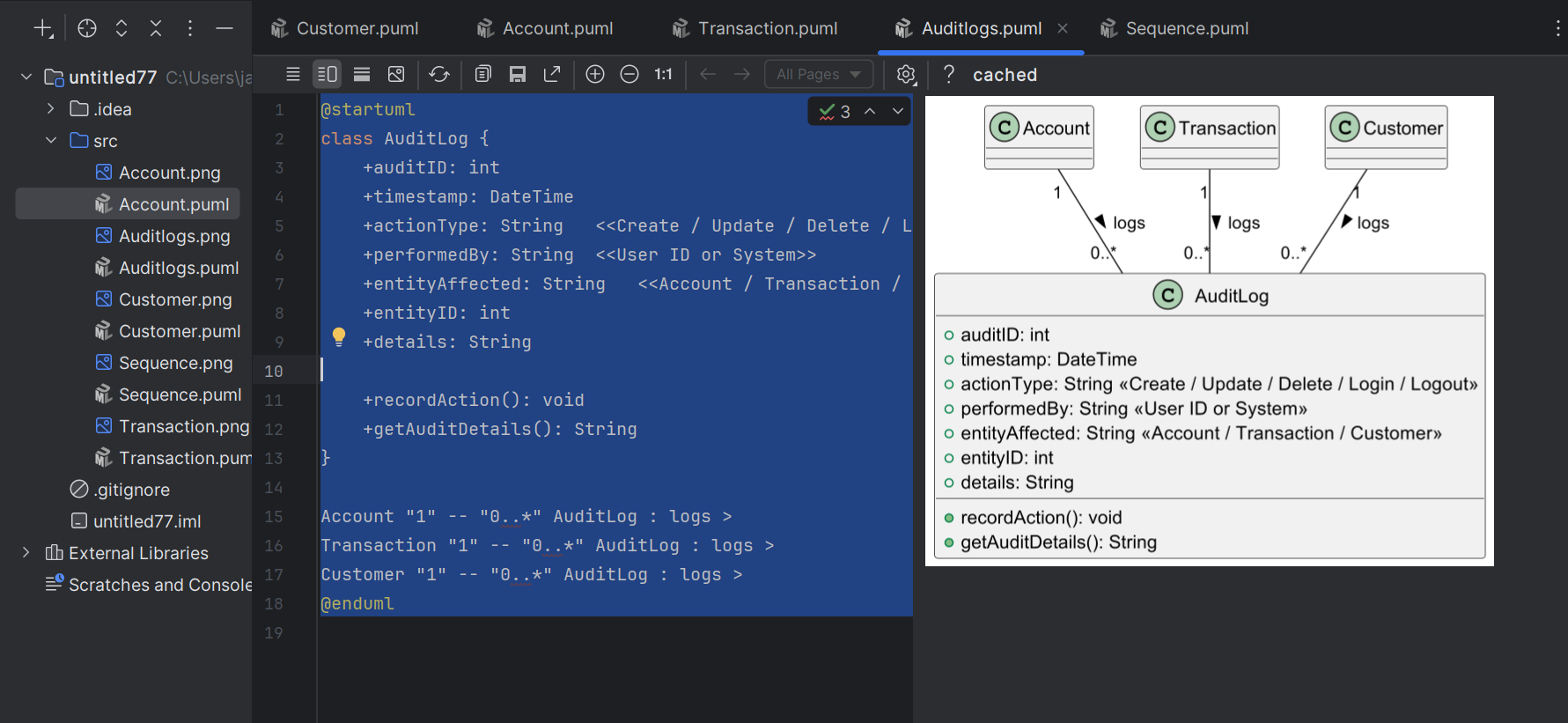
### 

@startuml  
class Transaction {  
 +transactionID: int  
 +date: DateTime  
 +amount: float  
 +transactionType: String <<Deposit / Withdrawal / Transfer>>  
 +sourceAccountNumber: int  
 +destinationAccountNumber: int <<optional>>  
 +status: String <<Success / Failed>>  
  
 +executeTransaction(): void  
 +generateReceipt(): void  
}  
  
Account "1" -- "0..\*" Transaction : has >  
@enduml



Audit Logs

@startuml  
class AuditLog {  
 +auditID: int  
 +timestamp: DateTime  
 +actionType: String <<Create / Update / Delete / Login / Logout>>  
 +performedBy: String <<User ID or System>>  
 +entityAffected: String <<Account / Transaction / Customer>>  
 +entityID: int  
 +details: String  
  
 +recordAction(): void  
 +getAuditDetails(): String  
}  
  
Account "1" -- "0..\*" AuditLog : logs >  
Transaction "1" -- "0..\*" AuditLog : logs >  
Customer "1" -- "0..\*" AuditLog : logs >  
@enduml

.

Sequence Diagram

@startuml  
actor Customer  
Customer -> Account : requestTransfer(fromAcc, toAcc, amount)  
Account -> MoneyTransfer : initiateTransfer(fromAcc, toAcc, amount)  
MoneyTransfer -> Account : withdraw(fromAcc, amount)  
Account -> Account : deposit(toAcc, amount)  
MoneyTransfer -> Transaction : recordTransfer()  
Transaction --> Customer : receipt  
@enduml

