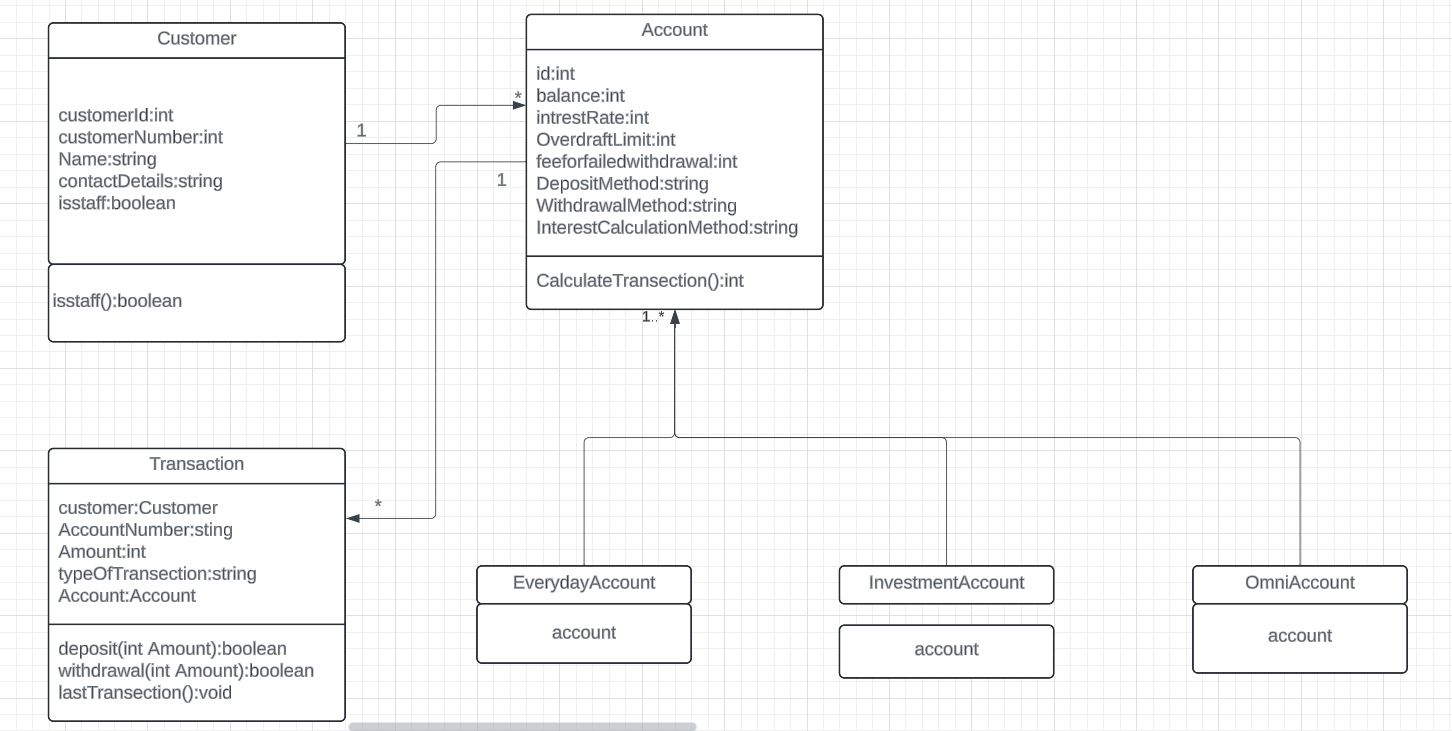
**IT7742 Advanced Programming - Q4 2024 Q1 2025**

Student id- 20220674

UML diagram



**Classes and Their Roles**

**1. Customer Class**

* Represents the user (customer) of the banking system.
* **Attributes**:
  + customerId: Unique identifier for each customer.
  + customerNumber: A specific account number for the customer.
  + name: The customer's name.
  + contactDetails: Contact information such as email or phone.
  + isStaff: Boolean to determine if the customer is a staff member.
* **Methods**:
  + isStaff(): Checks if the customer is staff or not.
* **Relationships**:
  + **1-to-Many Relationship** with Account (1..\*): One customer can own multiple accounts.

**2. Account Class**

* Represents a bank account with essential details and operations.
* **Attributes**:
  + id: Unique ID for the account.
  + balance: Current balance in the account.
  + interestRate: Rate at which interest is applied (if applicable).
  + overdraftLimit: Maximum overdraft allowed for the account.
  + feeForFailedWithdrawal: Fee charged for failed withdrawals.
  + depositMethod: Describes how deposits are made (e.g., online or cash).
  + withdrawalMethod: Describes how withdrawals are performed (e.g., ATM or teller).
  + interestCalculationMethod: Method used for calculating interest.
* **Methods**:
  + calculateTransaction(): Calculates fees, interest, or balances based on the transaction.
* **Relationships**:
  + **1-to-Many Relationship** with Transaction (1..\*): An account can have multiple transactions.
  + **Generalization Relationship** with subclasses (EverydayAccount, InvestmentAccount, and OmniAccount): These are specialized account types derived from the parent Account class.

**3. Transaction Class**

* Represents financial activities (deposit or withdrawal) linked to an account.
* **Attributes**:
  + accountNumber: The account on which the transaction occurs.
  + amount: The amount involved in the transaction.
  + typeOfTransaction: Indicates whether the transaction is a deposit or withdrawal.
  + account: Links the transaction to a specific account.
* **Methods**:
  + deposit(int amount): Processes deposits into the account.
  + withdrawal(int amount): Handles withdrawals from the account.
  + lastTransaction(): Tracks or retrieves the most recent transaction for the account.
* **Relationships**:
  + **Many-to-One Relationship** with Account (\*..1): Multiple transactions are linked to a single account.

**4. EverydayAccount Class**

* Subclass of Account that represents a checking account for daily use.
* **Relationships**:
  + **Inheritance**: Inherits attributes and methods from the Account class.

**5. InvestmentAccount Class**

* Subclass of Account designed for savings or investment purposes.
* **Relationships**:
  + **Inheritance**: Inherits attributes and methods from the Account class.

**6. OmniAccount Class**

* Subclass of Account offering combined features of checking and savings accounts.
* **Relationships**:
  + **Inheritance**: Inherits attributes and methods from the Account class.

**Relationships in the UML Diagram**

1. **Customer ↔ Account**:
   * A **Customer** can own multiple **Accounts** (1..\*), but each account belongs to only one customer (1).
2. **Account ↔ Transaction**:
   * An **Account** can be associated with multiple **Transactions** (1..\*), but each transaction involves one account (1).
3. **Account Subclasses**:
   * EverydayAccount, InvestmentAccount, and OmniAccount are specialized types of **Account**, connected via a **generalization relationship** (inheritance).