**UML DIAGRAM DAY-2**

**Main Entities (Classes)**

1. **Customer**
   * Holds customer information like id, name, email, phone, address, city, country, and status.
   * status comes from **CustomerStatus** (enum not shown but would be things like ACTIVE, INACTIVE, BLOCKED).
   * Relationship: A **customer** can have one or more **accounts**.
2. **Account**
   * Represents a bank account.
   * Attributes:
     + accountId, accountNumber (unique identifiers)
     + accountType (CHECKING, SAVINGS, LOAN, etc.)
     + balance, currency
     + status (ACTIVE, FROZEN, CLOSED)
     + openedAt, closedAt (timestamps)
     + customerId (links to Customer)
   * Relationship: An **account** can have multiple **transactions**.
3. **Transaction**
   * Represents any money movement.
   * Attributes:
     + transactionId, type (DEPOSIT, WITHDRAWAL, TRANSFER, etc.)
     + amount, currency
     + sourceAccountId, destinationAccountId (depending on transaction type)
     + status (PENDING, COMPLETED, FAILED/REVERSED)
     + description, createdAt, createdByUserId
   * Relationship: A **transaction** belongs to one account (but in case of transfers, two accounts are involved).
4. **Audit**
   * Keeps a record of important actions (for accountability).
   * Attributes:
     + auditId
     + action (CREATE, UPDATE, DELETE, LOGIN, etc.)
     + entityType, entityId (what was changed)
     + actorId, actorType (who did it: system or user)
     + performedAt (timestamp)
   * Relationship: Tracks activities on accounts, customers, or transactions.

**📌 Enums (Fixed Values)**

1. **AccountType**
   * CHECKING
   * SAVINGS
   * FIXED\_DEPOSIT
   * LOAN
2. **AccountStatus**
   * ACTIVE
   * FROZEN
   * CLOSED
3. **TransactionType**
   * DEPOSIT
   * WITHDRAWAL
   * TRANSFER
   * FEE, INTEREST, REVERSAL
4. **TransactionStatus**
   * PENDING
   * COMPLETED
   * FAILED / REVERSED
5. **AuditAction**
   * CREATE
   * UPDATE
   * DELETE
   * LOGIN

**📌 Notes (Special Rules)**

* **Deposit** → sourceAccountId = null (since money comes from outside).
* **Withdrawal** → destinationAccountId = null (money leaves the bank).
* **Transfer** → both sourceAccountId and destinationAccountId are set.
* Balance updates must be **atomic** (safe, consistent updates).
* Currency should be consistent unless **foreign exchange (FX)** is supported.
* Every record must have a **timestamp**.

**📌 Relationships**

* **Customer → Account**: One customer can have multiple accounts.
* **Account → Transaction**: Each account can have many transactions.
* **Audit** is connected to all entities for logging actions.