UITS

Future will be better than thy past

An initiative of PHP Family

University of Information Technology & Sciences

Lab Report-01

Drawing a Class Diagram

Course Title: Software Engineering and System Analysis Lab

Course Code: CSE 356

Submitted To

Khandoker Nosiba Arifin

Lecturer, Department of CSE, UITS

Submitted By

Name: Yeasmin Kabir Keya

ID: 0432220005101107

Batch: 52

Section: 6C

Experiment No: 02

Experiment Name: Study of designing Class Diagram of Blood Donation Management System (RedRescue)

Objectives

· To understand the role of UML Class Diagrams in software design. · To design a structured Class Diagram for a Blood Donation Management System. · To analyze relationships between different system entities.
· To enhance object-oriented modeling skills.

Apparatus

· Computer with UML modeling software (draw.io).
· Software Engineering course guidelines.

Theory:

A Class Diagram is a fundamental part of UML (Unified Modeling Language) used to represent the structure of a system by showing its classes, attributes, operations, and relationships among objects.

Blood Donation Management System (RedRescue) is designed to manage the process of blood donation efficiently. The system includes various entities such as donors, blood banks, hospitals, patients, and transaction records. The key functionalities include donor registration, blood storage, blood issuing, donation requests, and hospital orders.

Main classes in the diagram:

1. **BloodBankManager:** Manages blood banks.

2. **Receptionist:** Registers donors and handles blood orders.

3. **Donor:** Provides blood donation details.

4. **Blood:** Represents donated blood and its properties.

5. **BloodBank:** Stores and issues blood as per hospital requests.

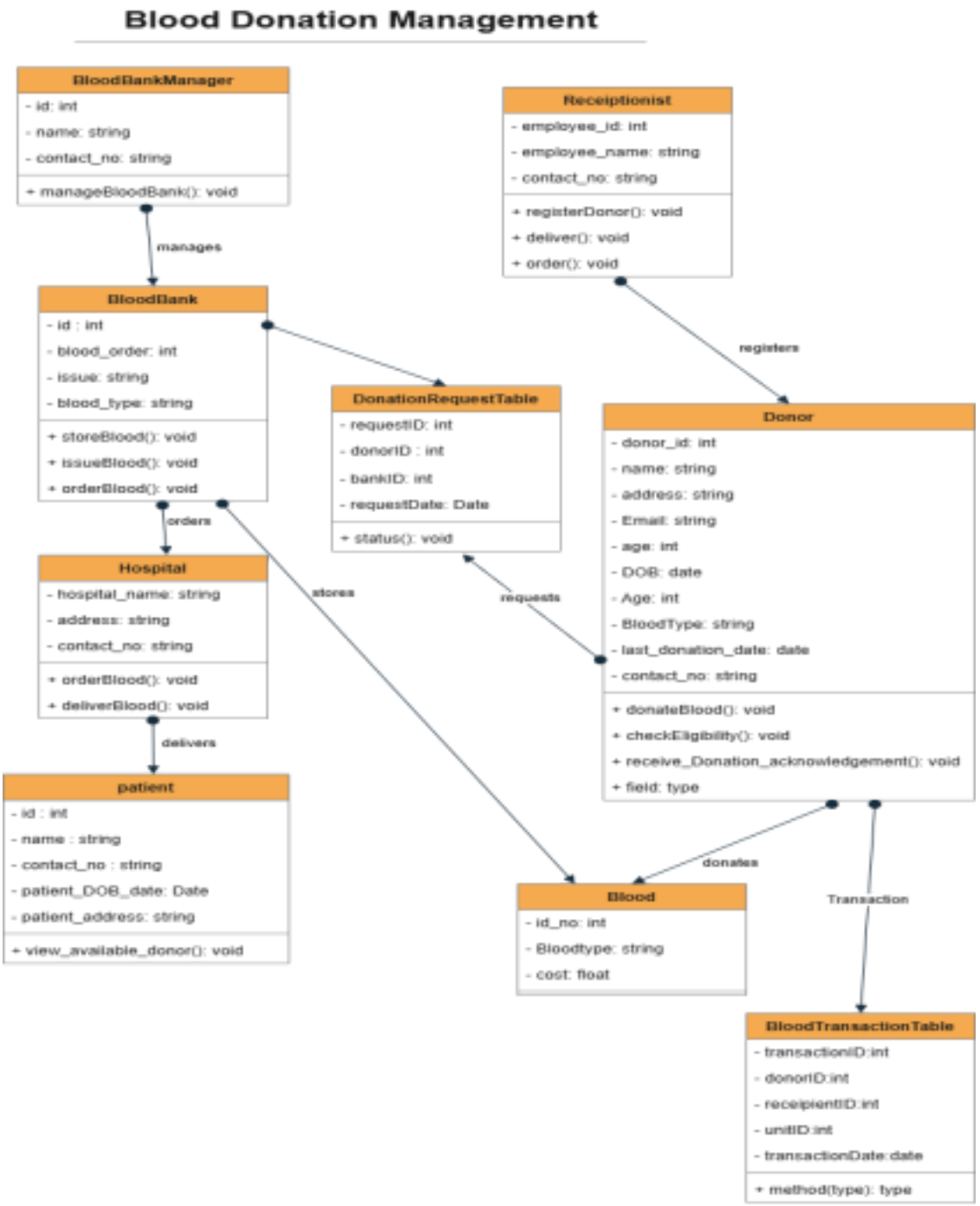
6. **Hospital:** Orders and receives blood for patients.

7. **Patient:** Checks available donors and receives blood.

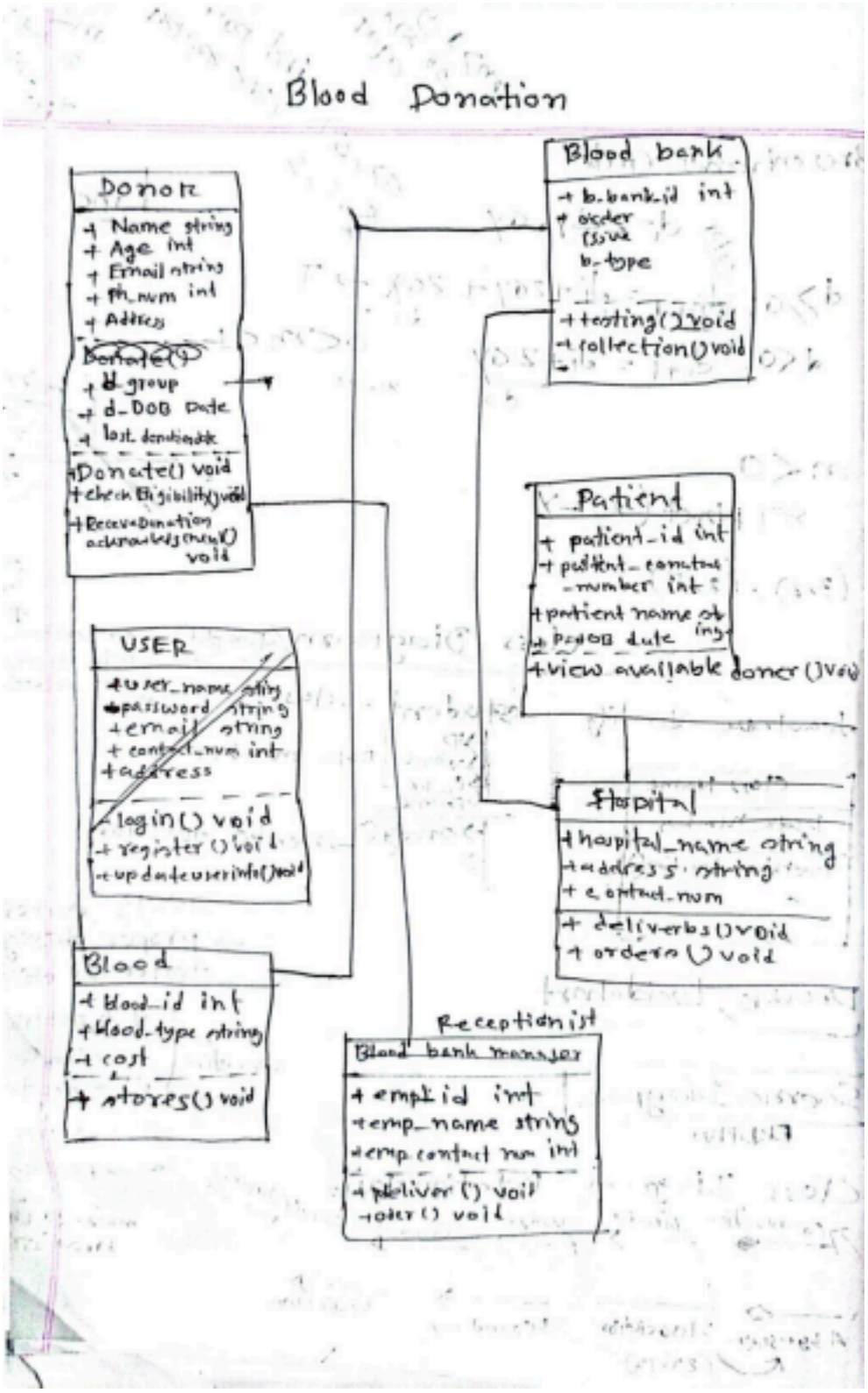
8. **DonationRequestTable:** Handles blood donation requests from donors to blood banks.

9. **BloodTransactionTable:** Records transaction details, including donor, recipient, unit of blood, transaction date, and method type.

Class Diagram:


Blood Donation Management
BloodBankManager
- id: int
- name: string
- contact_no: string
+ manageBloodBank(): void
manages
BloodBank
- id : int
- blood_order: int
- issue: string
- blood_type: string
+ storeBlood(): void
+ issueBlood(): void
+ orderBlood(): void
orders
Hospital
- hospital_name: string
- address: string
- contact_no: string
+ orderBlood(): void
+ deliverBlood(): void
delivers
patient
- id : int
- name : string
- contact_no : string
- patient_DOB_date: Date
- patient_address: string
+ view_available_donor(): void
stores
DonationRequestTable
- requestID: int
- donorID : int
- bankID: int
- requestDate: Date
+ status(): void
requests
Receiptionist
- employee_id: int
- employee_name: string
- contact_no: string
+ registerDonor(): void
+ deliver(): void
+ order(): void
registers
Donor
- donor_id: int
- name: string
- address: string
- Email: string
- age: int
- DOB: date
- Age: int
- BloodType: string
- last_donation_date: date
- contact_no: string
+ donateBlood(): void
+ checkEligibility(): void
+ receive_Donation_acknowledgement(): void
+ field: type
donates
Blood
- id_no: int
- Bloodtype: string
- cost: float
Transaction
BloodTransactionTable
- transactionID:int
- donorID:int
- receipientID:int
- unitID:int
- transactionDate:date
+ method(type): type

Class Work:



Result and Discussion:

- The class diagram now includes additional components such as

DonationRequestTable

- and BloodTransactionTable, which enhance the tracking of blood donation requests and transactions.
- The relationships such as “manages” “registers” “orders” “stores” “delivers” “requests” “donates” ensure smooth workflow representation.
- Encapsulation is maintained by defining attributes as private (-) and methods as public • (+) for controlled access.
- The system ensures efficient blood management, from donor registration to hospital delivery and transaction tracking.

Conclusion:

The Blood Donation Management System class diagram provides a structured approach to understanding how different components interact. It helps in designing an efficient system thatsimplifies donor registration, blood storage, ordering, donation request tracking, and delivery processes. The addition of DonationRequestTable and BloodTransactionTable further enhances transparency and efficiency. The use of UML class diagrams ensures clarity in software development, making system implementation more manageable.