Java Project

Blood Bank Application

Objective

The objective of the Blood Bank Management Application is to efficiently manage blood donations, donors, and blood requests. The system aims to ensure accurate tracking of donors, reliable documentation of blood donations, and timely fulfillment of blood requests, ultimately supporting the critical mission of providing safe and sufficient blood supplies to patients in need.

1. Donor

Represents: An individual who donates blood.

Attributes:

donorId: A unique identifier for the donor.

name: The name of the donor.

bloodType: The blood group/type of the donor (e.g., A+, O-).

bloodUnits: The amount of blood units the donor has contributed.

Role: Tracks personal information and blood type for each individual donor. This information is crucial for matching donors with recipients.

2. BloodDonor

Represents: The relationship between a donor and their blood donation.

Attributes:

donorId: The unique identifier of the donor.

donationId: The unique identifier of the blood donation.

Role: Links a donor to a specific donation, establishing a clear connection between the individual and the blood they have donated. This ensures traceability and proper documentation of blood donations.

3. Donation

Represents: An instance of blood being donated by a donor.

Attributes:

donationId: A unique identifier for the donation.

bloodType: The blood group/type of the donated blood.

donationDate: The date on which the blood was donated.

Role: Tracks the details of each blood donation, including the blood type and the date it was made. This data is essential for inventory management and ensuring the availability of different blood types.

4. BloodRequest

Represents: A request made by or on behalf of a patient in need of a specific type of blood.

Attributes:

requestId: A unique identifier for the blood request.

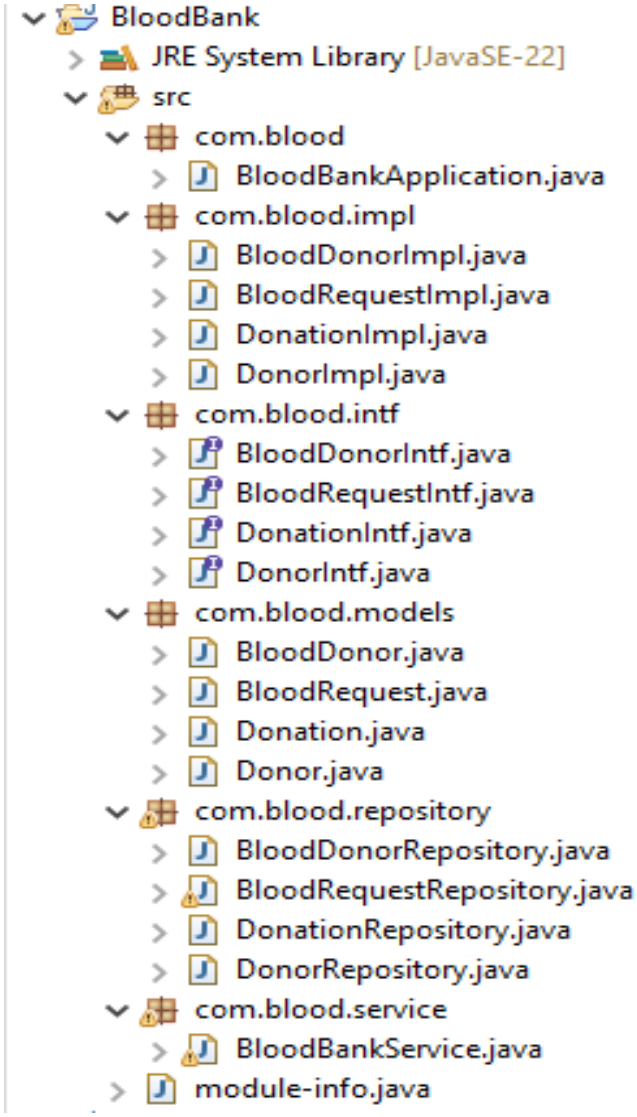
bloodType: The blood group/type requested.

quantity: The number of blood units required.

requestDate: The date on which the blood was requested.

Role: Manages the requests for blood, including the type and quantity of blood needed. This information is vital for fulfilling the needs of patients and managing the blood bank's supply.

Project Structure



Donor Module:

Donor.java

```
package com.blood.models;
public class Donor {
   private String donorId;
```

```
  private String name;
  private String bloodType;
        private String bloodUnits;
  public Donor(String donorId, String name, String bloodType, String bloodUnits ) {
     this.donorId = donorId;
     this.name = name;
     this.bloodType = bloodType;
     this.bloodUnits = bloodUnits;
  }
  public String getDonorId() {
     return donorId;
  }
  public void setDonorId(String donorId) {
     this.donorId = donorId;
  }
  public String getName() {
     return name;
  }
  public void setName(String name) {
     this.name = name;
  }
  public String getBloodType() {
     return bloodType;
  }
  public void setBloodType(String bloodType) {
     this.bloodType = bloodType;
  }
  public void setBloodUnits(String bloodUnits) {
     this.bloodUnits = bloodUnits;
  }
  public String GetBloodunit()
  {
        return bloodUnits;
  }
}
```

DonorImpl.java

```
package com.blood.impl;


import com.blood.intf.DonorIntf;
import com.blood.models.Donor;

import java.util.ArrayList;
import java.util.List;

public class DonorImpl implements DonorIntf {
    private List<Donor> donors = new ArrayList<>();

    @Override
    public void addDonor(Donor donor) {
        donors.add(donor);
    }

    @Override
    public Donor getDonor(String donorId) {
        return donors.stream()
                .filter(d -> d.getDonorId().equals(donorId))
                .findFirst()
                .orElse(null);
    }

    @Override
    public void updateDonor(Donor donor) {
        Donor existingDonor = getDonor(donor.getDonorId());
        if (existingDonor != null) {
            existingDonor.setName(donor.getName());
            existingDonor.setBloodType(donor.getBloodType());
        }
    }

    @Override
    public void deleteDonor(String donorId) {
        donors.removeIf(d -> d.getDonorId().equals(donorId));
    }

    @Override
```

```
    public List<Donor> getAllDonors() {
        return new ArrayList<>(donors);
    }
}
```

DonorIntf.java

```
package com.blood.intf;

import com.blood.models.Donor;

import java.util.List;

public interface DonorIntf {
    void addDonor(Donor donor);
    Donor getDonor(String donorId);
    void updateDonor(Donor donor);
    void deleteDonor(String donorId);
    List<Donor> getAllDonors();
}
```

DonorRepository.java

```
package com.blood.repository;


import com.blood.models.BloodDonor;

import java.util.ArrayList;
import java.util.List;

public class BloodDonorRepository {
    private static List<BloodDonor> bloodDonors = new ArrayList<>();



    public List<BloodDonor> getAllBloodDonors() {
        return new ArrayList<>(bloodDonors);
    }
```

```
}
```

BloodDonor Modules

BloodDonor.java

```
package com.blood.models;

public class BloodDonor {
   private String donorId;
   private String donationId;
   // Constructors
   public BloodDonor(String donorId, String donationId) {
      this.donorId = donorId;
      this.donationId = donationId;
   }
   // Getters and Setters
   public String getDonorId() { return donorId; }
   public void setDonorId(String donorId) { this.donorId = donorId; }
   public String getDonationId() { return donationId; }
   public void setDonationId(String donationId) { this.donationId = donationId; }
}
```

BloodDonorIntf.java

```
package com.blood.intf;

import com.blood.models.BloodDonor;

import java.util.List;

public interface BloodDonorIntf {
    void addBloodDonor(BloodDonor bloodDonor);
    BloodDonor getBloodDonor(String donorId);
    void updateBloodDonor(BloodDonor bloodDonor);
    void deleteBloodDonor(String donorId);
    List<BloodDonor> getAllBloodDonors();
}
```

BloodDonorImpl.java

```
package com.blood.impl;


import com.blood.intf.BloodDonorIntf;
import com.blood.models.BloodDonor;

import java.util.ArrayList;
import java.util.List;

public class BloodDonorImpl implements BloodDonorIntf {
    private List<BloodDonor> bloodDonors = new ArrayList<>();

    @Override
    public void addBloodDonor(BloodDonor bloodDonor) {
        bloodDonors.add(bloodDonor);
    }

    @Override
    public BloodDonor getBloodDonor(String donorId) {
        return bloodDonors.stream()
                .filter(b -> b.getDonorId().equals(donorId))
                .findFirst()
                .orElse(null);
    }

    @Override
    public void updateBloodDonor(BloodDonor bloodDonor) {
        BloodDonor existingBloodDonor = getBloodDonor(bloodDonor.getDonorId());
        if (existingBloodDonor != null) {
            existingBloodDonor.setDonationId(bloodDonor.getDonationId());
        }
    }

    @Override
    public void deleteBloodDonor(String donorId) {
        bloodDonors.removeIf(b -> b.getDonorId().equals(donorId));
    }
```

```
    @Override
    public List<BloodDonor> getAllBloodDonors() {
        return new ArrayList<>(bloodDonors);
    }
}
```

BloodDonorRepository.java

```
package com.blood.repository;


import com.blood.models.BloodDonor;

import java.util.ArrayList;
import java.util.List;

public class BloodDonorRepository {
    private static List<BloodDonor> bloodDonors = new ArrayList<>();



    public List<BloodDonor> getAllBloodDonors() {
        return new ArrayList<>(bloodDonors);
    }
}
```

Donation Module:

Donation.java

```
package com.blood.models;
import java.util.Date;
public class Donation {
    private String donationId;
    private String bloodType;
    private Date donationDate;
    // Constructors
    public Donation(String donationId, String bloodType, Date donationDate) {
```

```
        this.donationId = donationId;
        this.bloodType = bloodType;
        this.donationDate = donationDate;
    }
    // Getters and Setters
    public String getDonationId() { return donationId; }
    public void setDonationId(String donationId) { this.donationId = donationId; }
    public String getBloodType() { return bloodType; }
    public void setBloodType(String bloodType) { this.bloodType = bloodType; }
    public Date getDonationDate() { return donationDate; }
    public void setDonationDate(Date donationDate) { this.donationDate = donationDate; }
}
```

DonationIntf.java

```
package com.blood.intf;

import com.blood.models.Donation;

import java.util.List;

public interface DonationIntf {
    void addDonation(Donation donation);
    Donation getDonation(String donationId);
    void updateDonation(Donation donation);
    void deleteDonation(String donationId);
    List<Donation> getAllDonations();
}
```

DonationImpl.java

```
package com.blood.impl;

import com.blood.intf.DonationIntf;
import com.blood.models.Donation;

import java.util.ArrayList;
import java.util.List;
```

```
public class DonationImpl implements DonationIntf {
    private List<Donation> donations = new ArrayList<>();

    @Override
    public void addDonation(Donation donation) {
        donations.add(donation);
    }

    @Override
    public Donation getDonation(String donationId) {
        return donations.stream()
                .filter(d -> d.getDonationId().equals(donationId))
                .findFirst()
                .orElse(null);
    }

    @Override
    public void updateDonation(Donation donation) {
        Donation existingDonation = getDonation(donation.getDonationId());
        if (existingDonation != null) {
            existingDonation.setBloodType(donation.getBloodType());
            existingDonation.setDonationDate(donation.getDonationDate());
        }
    }

    @Override
    public void deleteDonation(String donationId) {
        donations.removeIf(d -> d.getDonationId().equals(donationId));
    }

    @Override
    public List<Donation> getAllDonations() {
        return new ArrayList<>(donations);
    }
}
```

DonationRepository.java

```
package com.blood.repository;
```

```
import com.blood.models.Donation;


import java.util.ArrayList;
import java.util.List;
import java.util.Date;

public class DonationRepository {
    private static List<Donation> donations = new ArrayList<>();

    static {
        donations.add(new Donation("DN1", "O+", new Date()));
        donations.add(new Donation("DN2", "A-", new Date()));
    }

    public List<Donation> getAllDonations() {
        return new ArrayList<>(donations);
    }
}
```

BloodRequest Module:

BloodRequest.java

```
package com.blood.models;
import java.util.Date;
public class BloodRequest {
   private String requestId;
   private String bloodType;
   private int quantity;
   private Date requestDate;
   // Constructors
   public BloodRequest(String requestId, String bloodType, int quantity, Date
requestDate) {
       this.requestId = requestId;
       this.bloodType = bloodType;
```

```
        this.quantity = quantity;
        this.requestDate = requestDate;
    }
    // Getters and Setters
    public String getRequestId() { return requestId; }
    public void setRequestId(String requestId) { this.requestId = requestId; }
    public String getBloodType() { return bloodType; }
    public void setBloodType(String bloodType) { this.bloodType = bloodType; }
    public int getQuantity() { return quantity; }
    public void setQuantity(int quantity) { this.quantity = quantity; }
    public Date getRequestDate() { return requestDate; }
    public void setRequestDate(Date requestDate) { this.requestDate = requestDate; }
}
```

BloodRequestIntf.java

```
package com.blood.intf;

import com.blood.models.BloodRequest;

import java.util.List;

public interface BloodRequestIntf {
    void addRequest(BloodRequest request);
    BloodRequest getRequest(String requestId);
    void updateRequest(BloodRequest request);
    void deleteRequest(String requestId);
    List<BloodRequest> getAllRequests();
}
```

BloodRequestImpl.java

```
package com.blood.impl;
import com.blood.intf.BloodRequestIntf;
import com.blood.models.BloodRequest;
import java.util.ArrayList;
import java.util.List;
```

```
public class BloodRequestImpl implements BloodRequestIntf {
  private List<BloodRequest> requests = new ArrayList<>();
  @Override
  public void addRequest(BloodRequest request) {
    requests.add(request);
  }
  @Override
  public BloodRequest getRequest(String requestId) {
    return requests.stream()
        .filter(r -> r.getRequestId().equals(requestId))
        .findFirst()
        .orElse(null);
  }
  @Override
  public void updateRequest(BloodRequest request) {
    BloodRequest existingRequest = getRequest(request.getRequestId());
    if (existingRequest != null) {
      existingRequest.setBloodType(request.getBloodType());
      existingRequest.setQuantity(request.getQuantity());
      existingRequest.setRequestDate(request.getRequestDate());
    }
  }
  @Override
  public void deleteRequest(String requestId) {
    requests.removeIf(r -> r.getRequestId().equals(requestId));
  }
  @Override
  public List<BloodRequest> getAllRequests() {
    return new ArrayList<>(requests);
  }
}
```

BloodRequestRepository.java

```
package com.blood.repository;

import com.blood.models.BloodRequest;

import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Date;

public class BloodRequestRepository {
    private static List<BloodRequest> requests = new ArrayList<>();


    public List<BloodRequest> getAllRequests() {
        return new ArrayList<>(requests);
    }
}
```

Service Module:

BloodBankService.java

```
package com.blood.service;



import com.blood.impl.DonorImpl;
import com.blood.impl.BloodDonorImpl;
import com.blood.impl.DonationImpl;
import com.blood.impl.BloodRequestImpl;
import com.blood.intf.DonorIntf;
import com.blood.intf.BloodDonorIntf;
import com.blood.intf.DonationIntf;
import com.blood.intf.BloodRequestIntf;
import com.blood.models.Donor;
import com.blood.models.BloodDonor;
import com.blood.models.Donation;
import com.blood.models.BloodRequest;

import java.util.Date;
import java.util.List;

public class BloodBankService {
    private DonorIntf donorService;
    private BloodDonorIntf bloodDonorService;
```

```
private DonationIntf donationService;
private BloodRequestIntf bloodRequestService;

// Constructor to initialize services
public BloodBankService() {
    this.donorService = new DonorImpl();
    this.bloodDonorService = new BloodDonorImpl();
    this.donationService = new DonationImpl();
    this.bloodRequestService = new BloodRequestImpl();
}

// Donor Operations
public void addDonor(Donor donor) {
    donorService.addDonor(donor);
}

public Donor getDonor(String donorId) {
    return donorService.getDonor(donorId);
}

public void updateDonor(Donor donor) {
    donorService.updateDonor(donor);
}

public void deleteDonor(String donorId) {
    donorService.deleteDonor(donorId);
}

public List<Donor> getAllDonors() {
    return donorService.getAllDonors();
}

// Blood Donor Operations
public void addBloodDonor(BloodDonor bloodDonor) {
    bloodDonorService.addBloodDonor(bloodDonor);
}

public BloodDonor getBloodDonor(String donorId) {
    return bloodDonorService.getBloodDonor(donorId);
}
```

```
public void updateBloodDonor(BloodDonor bloodDonor) {
    bloodDonorService.updateBloodDonor(bloodDonor);
}

public void deleteBloodDonor(String donorId) {
    bloodDonorService.deleteBloodDonor(donorId);
}

public List<BloodDonor> getAllBloodDonors() {
    return bloodDonorService.getAllBloodDonors();
}

// Donation Operations
public void addDonation(Donation donation) {
    donationService.addDonation(donation);
}

public Donation getDonation(String donationId) {
    return donationService.getDonation(donationId);
}

public void updateDonation(Donation donation) {
    donationService.updateDonation(donation);
}

public void deleteDonation(String donationId) {
    donationService.deleteDonation(donationId);
}

public List<Donation> getAllDonations() {
    return donationService.getAllDonations();
}

// Blood Request Operations
public void addBloodRequest(BloodRequest request) {
    bloodRequestService.addRequest(request);
}

public BloodRequest getBloodRequest(String requestId) {
```

```
        return bloodRequestService.getRequest(requestId);
    }

    public void updateBloodRequest(BloodRequest request) {
        bloodRequestService.updateRequest(request);
    }

    public void deleteBloodRequest(String requestId) {
        bloodRequestService.deleteRequest(requestId);
    }

    public List<BloodRequest> getAllBloodRequests() {
        return bloodRequestService.getAllRequests();
    }
}
```

Main Method

BloodBankApplication.java

```
package com.blood;

import com.blood.impl.DonorImpl;
import com.blood.impl.BloodDonorImpl;
import com.blood.impl.DonationImpl;
import com.blood.impl.BloodRequestImpl;
import com.blood.intf.DonorIntf;
import com.blood.intf.BloodDonorIntf;
import com.blood.intf.DonationIntf;
import com.blood.intf.BloodRequestIntf;
import com.blood.models.Donor;
import com.blood.models.BloodDonor;
import com.blood.models.Donation;
import com.blood.models.BloodRequest;

import java.util.Date;
import java.util.Scanner;
```

```
public class BloodBankApplication {

    public static void main(String[] args) {
        // Create service instances
        DonorIntf donorService = new DonorImpl();
        BloodDonorIntf bloodDonorService = new BloodDonorImpl();
        DonationIntf donationService = new DonationImpl();
        BloodRequestIntf bloodRequestService = new BloodRequestImpl();

        Scanner scanner = new Scanner(System.in);
        boolean exit = false;

        while (!exit) {
            System.out.println("\nBlood Bank Management System");
            System.out.println("1. Add Donor");
            System.out.println("2. View All Donors");
            System.out.println("3. Update Donor");
            System.out.println("4. Delete Donor");
            System.out.println("5. View All Donations");
            System.out.println("6. View All Blood Donors");
            System.out.println("7. View All Blood Requests");
            System.out.println("8. Create Blood Request");
            System.out.println("9. Exit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
            case 1:
                // Add Donor
                System.out.println("\nEnter Donor Details:");
                System.out.print("Donor ID: ");
                String donorId = scanner.nextLine();
                System.out.print("Name: ");
                String name = scanner.nextLine();
                System.out.print("Blood Type: ");
                String bloodType = scanner.nextLine();
                System.out.print("Blood Units Received: ");
                String bloodunits = scanner.nextLine();
```

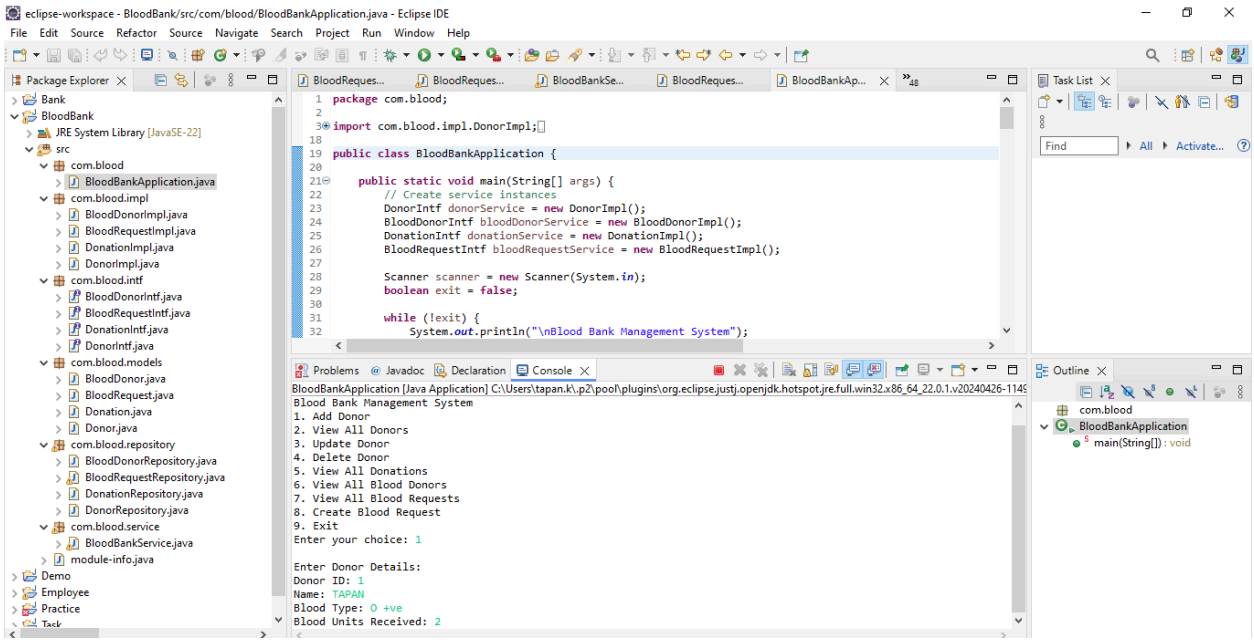
```
            Donor donor = new Donor(donorId, name, bloodType, bloodunits);
            donorService.addDonor(donor);
            System.out.println("Donor added successfully.");

            // Automatically link donor to donation
           //  System.out.println("\nEnter Donation Details:");
            String donationId = "DN" + donorId; // Example logic to create a unique
donation ID
            Donation donation = new Donation(donationId, bloodType, new Date());
            donationService.addDonation(donation);
            BloodDonor bloodDonor = new BloodDonor(donorId, donationId);
            bloodDonorService.addBloodDonor(bloodDonor);
            System.out.println("Donor linked to donation successfully.");
            break;

        case 2:
            // View All Donors
            System.out.println("\nAll Donors:");
            System.out.println("| ID | Name | BloodType | Units |");
            donorService.getAllDonors().forEach(d ->
                System.out.println(d.getDonorId() + "   -  " + d.getName() + "   - " +
d.getBloodType()+ "     -  " +d.GetBloodunit())//+ "   - " +d.GetBloodunit()
            );
            break;

        case 3:
            // Update Donor
            System.out.println("\nEnter Donor ID to update:");
            String updateDonorId = scanner.nextLine();
            Donor existingDonor = donorService.getAllDonors().stream()
                .filter(d -> d.getDonorId().equals(updateDonorId))
                .findFirst()
                .orElse(null);

            if (existingDonor != null) {
                System.out.println("Current Name: " + existingDonor.getName());
                System.out.print("New Name: ");
                String newName = scanner.nextLine();
```

```
                System.out.println("Current Blood Type: " + existingDonor.getBloodType());
                System.out.print("New Blood Type: ");
                String newBloodType = scanner.nextLine();

                System.out.println("Current Blood Units: " + existingDonor.GetBloodunit());
                System.out.print("New Blood Units: ");
                String newBloodunits= scanner.nextLine();


                existingDonor.setName(newName);
                existingDonor.setBloodType(newBloodType);
                existingDonor.setBloodUnits(newBloodunits);

                System.out.println("Donor updated successfully.");
            } else {
                System.out.println("Donor with ID " + updateDonorId + " not found.");
            }
            break;

        case 4:
            // Delete Donor
            System.out.println("\nEnter Donor ID to delete:");
            String deleteDonorId = scanner.nextLine();

            Donor donorToDelete = donorService.getDonor(deleteDonorId);

            if (donorToDelete != null) {
                donorService.deleteDonor(deleteDonorId); // Use the deleteDonor method
from the service
                System.out.println("Donor removed successfully.");
            } else {
                System.out.println("Donor with ID " + deleteDonorId + " not found.");
            }
            break;



        case 5:
            // View All Donations
```

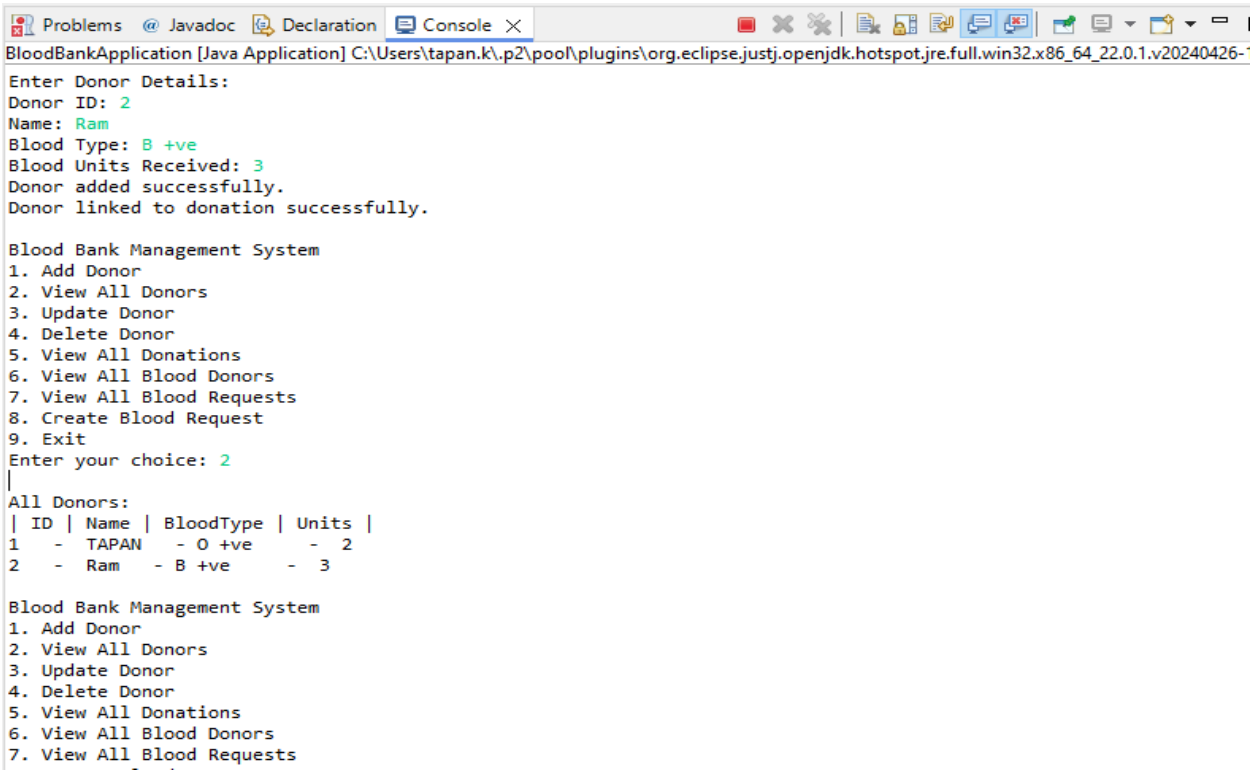
```
                System.out.println("\nAll Donations:");
                donationService.getAllDonations().forEach(d ->
                    System.out.println(d.getDonationId() + " - " + d.getBloodType() + " - " +
d.getDonationDate())
                );
                break;


            case 6:
                    // View All Blood Donors
                System.out.println("\nAll Blood Donors:");
                bloodDonorService.getAllBloodDonors().forEach(bloodDonorEntry ->
                    System.out.println(bloodDonorEntry.getDonorId() + " - " +
bloodDonorEntry.getDonationId())
                );
                break;

            case 7:
                // View All Blood Requests
                System.out.println("\nAll Blood Requests:");
                bloodRequestService.getAllRequests().forEach(request ->
                    System.out.println(request.getRequestId() + " - " + request.getBloodType()
+ " - " + request.getQuantity() + " units - " + request.getRequestDate())
                );
                break;



                case 8:
                    // Create Blood Request
                    System.out.println("\nEnter Blood Request Details:");
                    System.out.print("Request ID: ");
                    String requestId = scanner.nextLine();
                    System.out.print("Blood Type: ");
                    String bloodTypes = scanner.nextLine();
                    System.out.print("Quantity (units): ");
                    int quantity = scanner.nextInt();
                    scanner.nextLine(); // Consume newline

                    BloodRequest request = new BloodRequest(requestId, bloodTypes,
quantity, new Date());
```

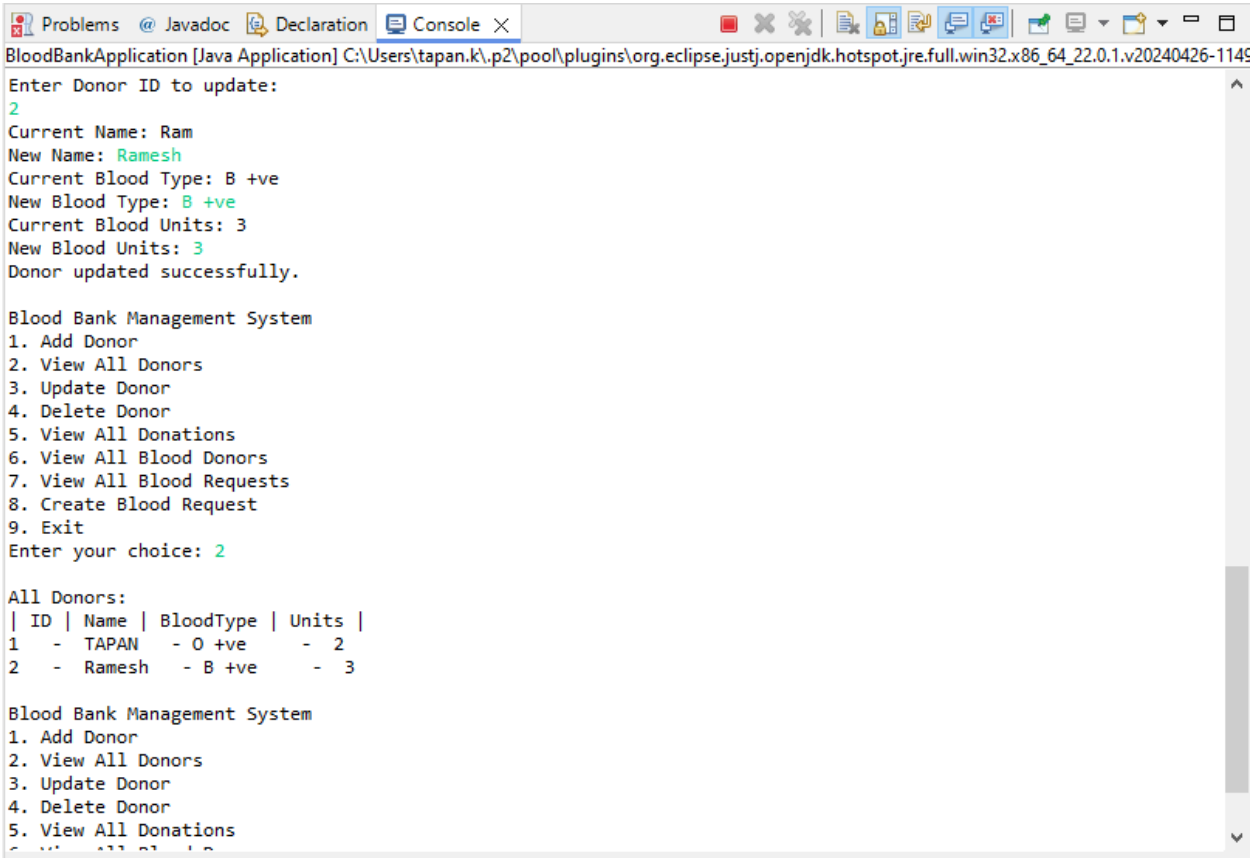
```
                bloodRequestService.addRequest(request);
                System.out.println("Blood request added successfully.");
                break;

            case 9:
                // Exit
                System.out.println("Exiting the application.");
                exit = true;
                break;

            default:
                System.out.println("Invalid choice. Please try again.");
                break;
        }
    }

    scanner.close();
  }
}
```

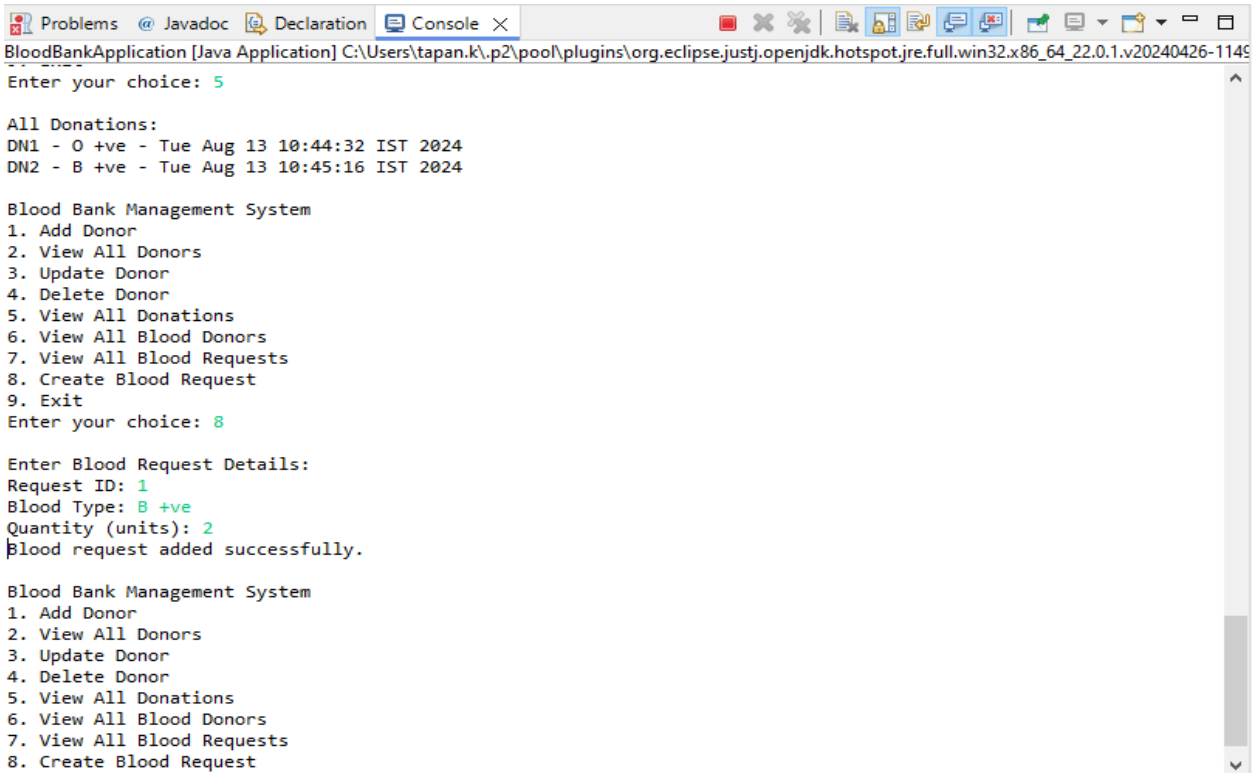
Output :-



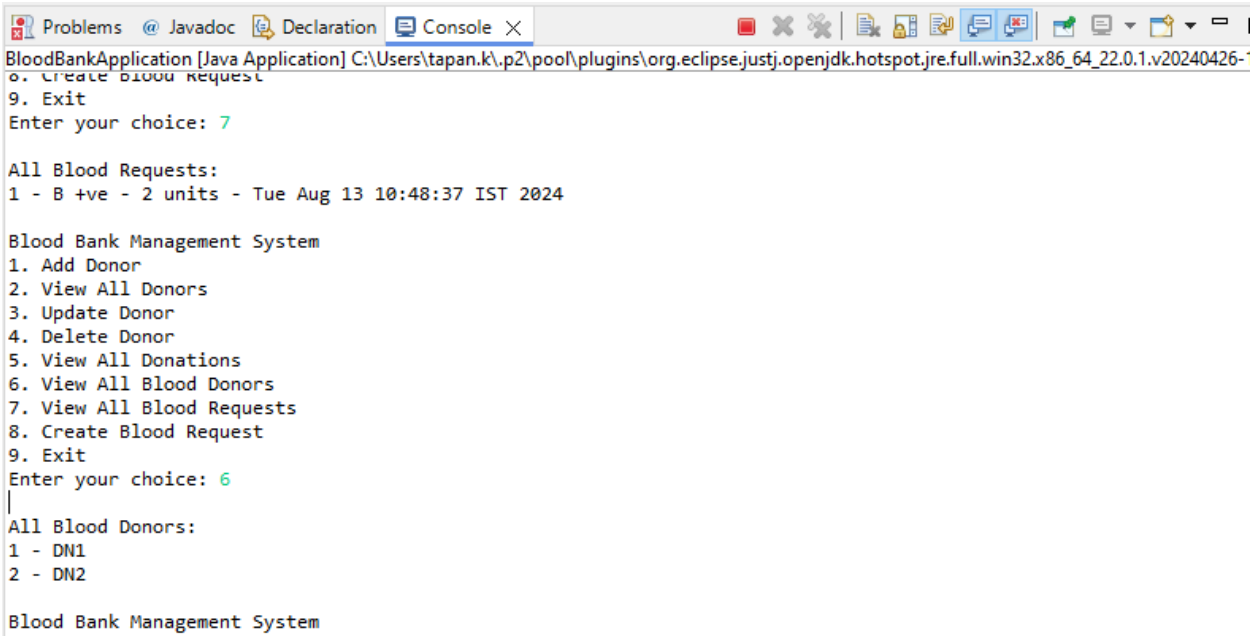
```
```

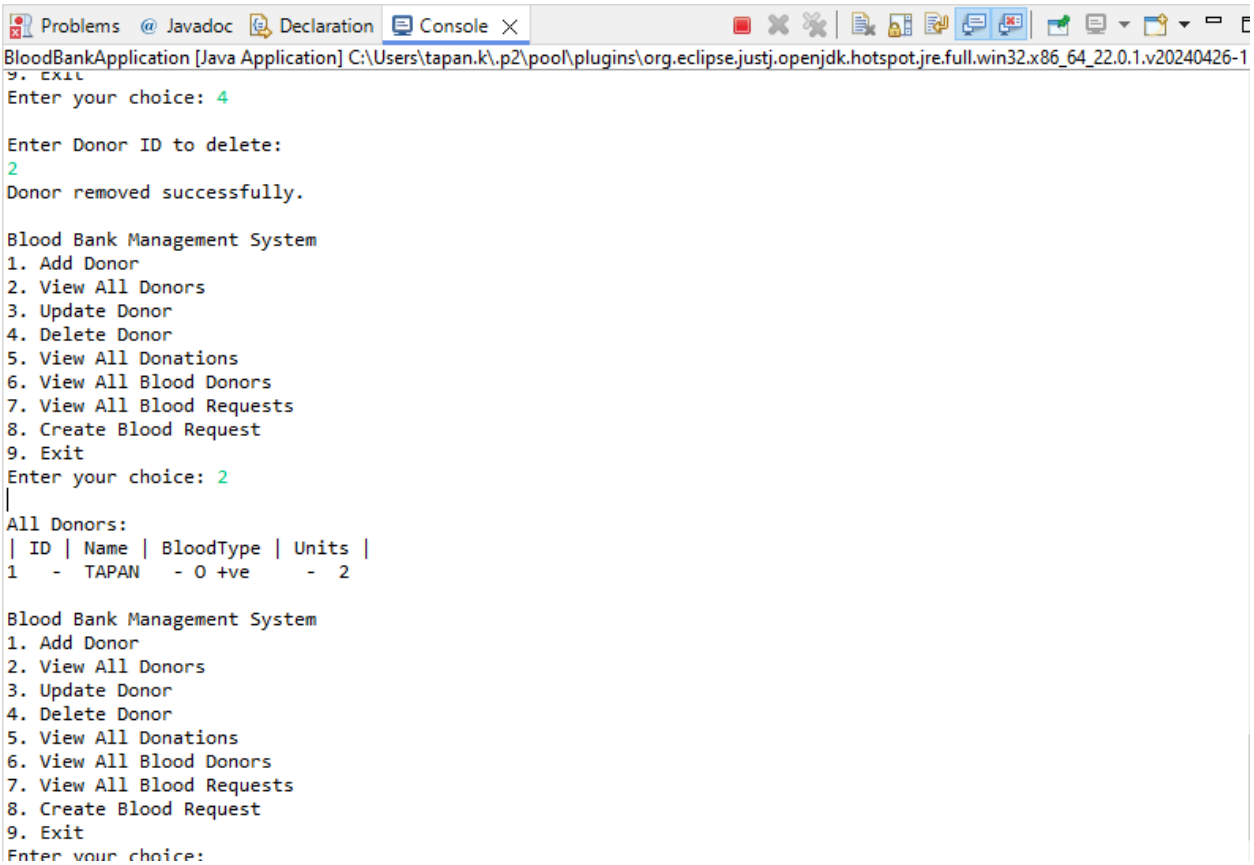

Problems
Javadoc
Declaration
Console
BloodBankApplication [Java Application] C:\Users\tapan.k\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-
Enter Donor Details:
Donor ID: 2
Name: Ram
Blood Type: B +ve
Blood Units Received: 3
Donor added successfully.
Donor linked to donation successfully.
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations
6. View All Blood Donors
7. View All Blood Requests
8. Create Blood Request
9. Exit
Enter your choice: 2
All Donors:
| ID | Name | BloodType | Units |
1 - TAPAN - O +ve - 2
2 - Ram - B +ve - 3
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations
6. View All Blood Donors
7. View All Blood Requests


Problems
Javadoc
Declaration
Console
BloodBankApplication [Java Application] C:\Users\tapan.k\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149
Enter Donor ID to update:
2
Current Name: Ram
New Name: Ramesh
Current Blood Type: B +ve
New Blood Type: B +ve
Current Blood Units: 3
New Blood Units: 3
Donor updated successfully.
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations
6. View All Blood Donors
7. View All Blood Requests
8. Create Blood Request
9. Exit
Enter your choice: 2
All Donors:
| ID | Name | BloodType | Units |
1 - TAPAN - O +ve - 2
2 - Ramesh - B +ve - 3
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations


Problems
Javadoc
Declaration
Console
BloodBankApplication [Java Application] C:\Users\tapan.k\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149
Enter your choice: 5
All Donations:
DN1 - O +ve - Tue Aug 13 10:44:32 IST 2024
DN2 - B +ve - Tue Aug 13 10:45:16 IST 2024
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations
6. View All Blood Donors
7. View All Blood Requests
8. Create Blood Request
9. Exit
Enter your choice: 8
Enter Blood Request Details:
Request ID: 1
Blood Type: B +ve
Quantity (units): 2
Blood request added successfully.
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations
6. View All Blood Donors
7. View All Blood Requests
8. Create Blood Request


Problems
Javadoc
Declaration
Console
BloodBankApplication [Java Application] C:\Users\tapan.k\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1
Enter your choice: 5
All Donations:
DN1 - O +ve - Tue Aug 13 10:44:32 IST 2024
DN2 - B +ve - Tue Aug 13 10:45:16 IST 2024
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations
6. View All Blood Donors
7. View All Blood Requests
8. Create Blood Request
9. Exit
Enter your choice: 8
Enter Blood Request Details:
Request ID: 1
Blood Type: B +ve
Quantity (units): 2
Blood request added successfully.


Problems
Javadoc
Declaration
Console
BloodBankApplication [Java Application] C:\Users\tapan.k\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1
8. Create Blood Request
9. Exit
Enter your choice: 7
All Blood Requests:
1 - B +ve - 2 units - Tue Aug 13 10:48:37 IST 2024
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations
6. View All Blood Donors
7. View All Blood Requests
8. Create Blood Request
9. Exit
Enter your choice: 6
All Blood Donors:
1 - DN1
2 - DN2
Blood Bank Management System


Problems
Javadoc
Declaration
Console
BloodBankApplication [Java Application] C:\Users\tapan.k\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1
9. Exit
Enter your choice: 4
Enter Donor ID to delete:
2
Donor removed successfully.
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations
6. View All Blood Donors
7. View All Blood Requests
8. Create Blood Request
9. Exit
Enter your choice: 2
All Donors:
| ID | Name | BloodType | Units |
1 - TAPAN - O +ve - 2
Blood Bank Management System
1. Add Donor
2. View All Donors
3. Update Donor
4. Delete Donor
5. View All Donations
6. View All Blood Donors
7. View All Blood Requests
8. Create Blood Request
9. Exit
Enter your choice: