JAVA MINI PROJECT

TOPIC : BLOOD DONOR MANAGEMENT SYSTEM

AIM: A simple blood donor management project in Java for designing a system to store, retrieve, and manage information about blood donors.

DESCRIPTION:

This Java project is a simple Blood Donor Management System that allows users to register blood donors and search for donors based on their blood group. The application uses a MySQL database to store donor information, including details such as name, blood group, age, gender, weight, phone number, and address.

Here's a brief description of the main components and functionality:

Database Initialization:

The program connects to a MySQL database specified by the DB_URL, DB_USER, and DB_PASSWORD constants. It initializes the database by creating a table named donors with columns for donor details.

Main Menu:

The main method contains an interactive menu that continuously prompts the user to choose between three options:

Register Donor (Option 1): Allows the user to input donor information, validates the input, and inserts the information into the database.

Search Donor (Option 2): Prompts the user to enter a blood group, searches the database for donors with that blood group, and displays the relevant donor information.

Exit (Option 3): Exits the program.

Input Validation:

The program includes input validation for various user inputs, such as blood group, age, gender, weight, and donor name. It ensures that entered data meets specific criteria.

Database Operations:

Database operations are encapsulated in methods like initializeDatabase, registerDonor, and searchDonor. It uses JDBC (Java Database Connectivity) to connect to the MySQL database and perform operations like creating tables and executing SQL queries.

Donor Information Display:

When searching for donors, the program retrieves and displays donor information such as name, age, gender, weight, phone number, and address.

Code Organization:

The code is structured into methods for better readability and maintainability. It follows best practices, such as closing resources using try-with-resources and using constants for database connection details.

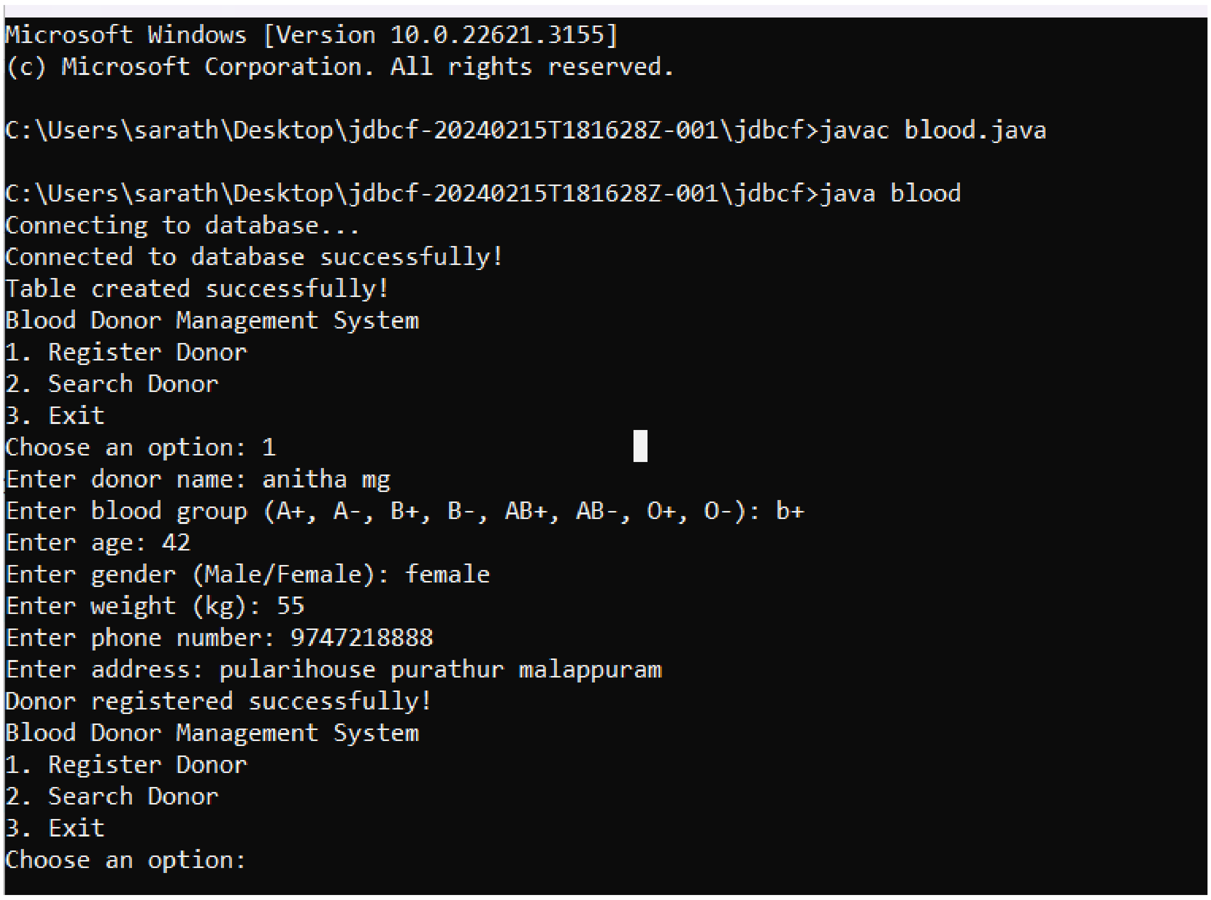
CODE:

```
import java.sql.*;

import java.util.Scanner;


public class blood {


  private static final String DB_URL =
"jdbc:mysql://localhost:3306/sarath_db?characterEncoding=utf8";

  private static final String DB_USER = "root";

  private static final String DB_PASSWORD = "Sarath@9747";


  public static void main(String[] args) {

    initializeDatabase();

    Scanner scanner = new Scanner(System.in);


    while (true) {

      System.out.println("Blood Donor Management System");

      System.out.println("1. Register Donor");

      System.out.println("2. Search Donor");

      System.out.println("3. Exit");

      System.out.print("Choose an option: ");


      int choice = scanner.nextInt();


      switch (choice) {

        case 1:

          registerDonor(scanner);

          break;

        case 2:
                searchDonor(scanner);
                break;
            case 3:
                System.out.println("Exiting...");
                System.exit(0);
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
}

private static void initializeDatabase() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        System.out.println("Connecting to database...");
        Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        System.out.println("Connected to database successfully!");

        String createTableSQL = "CREATE TABLE IF NOT EXISTS donors (name VARCHAR(255),
bloodgroup VARCHAR(5), gender VARCHAR(10), weight FLOAT, phonenumber
VARCHAR(15), address VARCHAR(255), age INT)";
        try (Statement statement = connection.createStatement()) {
            statement.executeUpdate(createTableSQL);
            System.out.println("Table created successfully!");
        }
        connection.close();
    } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
    }
```

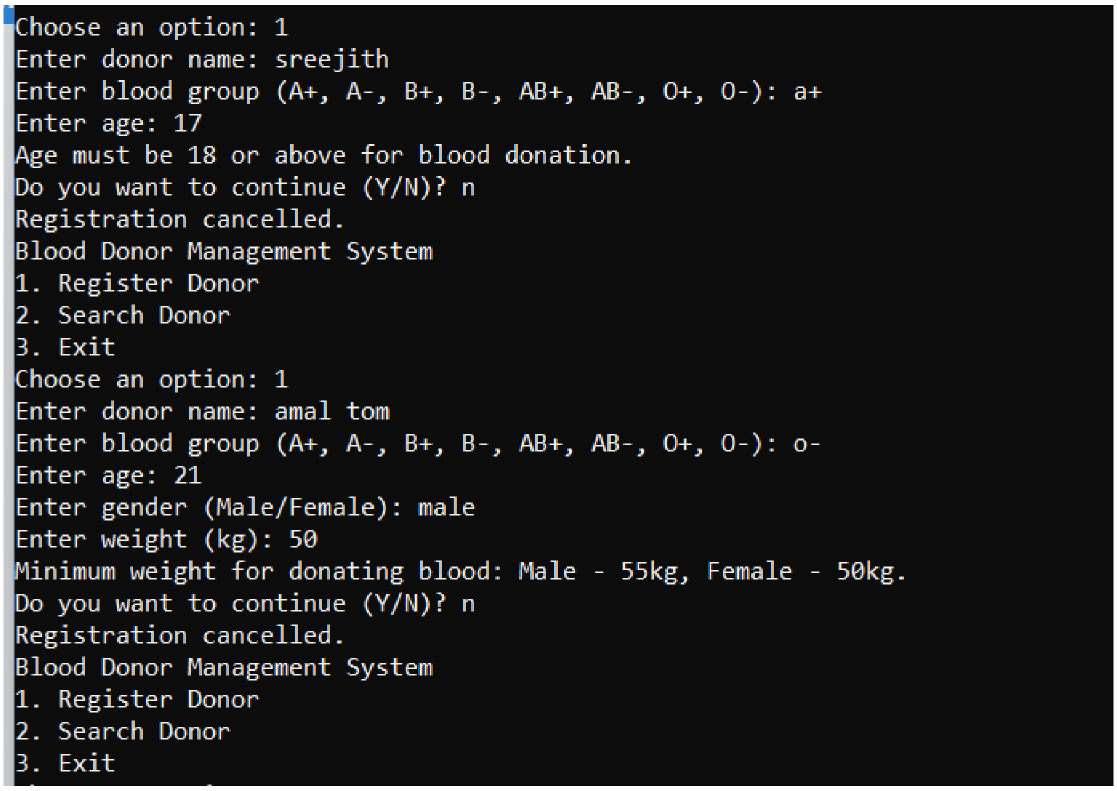
```java
}

private static void registerDonor(Scanner scanner) {
    System.out.print("Enter donor name: ");
    // Consume the newline character left after the previous nextInt() call
    scanner.nextLine();
    String name = scanner.nextLine().toUpperCase();

    // Validate blood group
    String bloodGroup;
    while (true) {
        System.out.print("Enter blood group (A+, A-, B+, B-, AB+, AB-, O+, O-): ");
        bloodGroup = scanner.next().toUpperCase();
        if (isValidBloodGroup(bloodGroup)) {
            break;
        } else {
            System.out.println("Invalid blood group. Please enter a relevant blood group.");
        }
    }

    // Validate age
    int age;
    while (true) {
        System.out.print("Enter age: ");
        age = scanner.nextInt();
        if (age >= 18) {
            break;
        } else {
            System.out.println("Age must be 18 or above for blood donation.");
            System.out.print("Do you want to continue (Y/N)? ");
            String continueChoice = scanner.next().toUpperCase();
            if (!continueChoice.equals("Y")) {
                System.out.println("Registration cancelled.");
                return;
            }
        }
    }

    // Validate gender
    String gender;
    while (true) {
        System.out.print("Enter gender (Male/Female): ");
        gender = scanner.next().toUpperCase();
        if (gender.equals("MALE") || gender.equals("FEMALE")) {
            break;
        } else {
            System.out.println("Invalid gender. Please enter Male or Female.");
        }
    }

    // Validate weight based on gender
    float weight;
    while (true) {
        System.out.print("Enter weight (kg): ");
        weight = scanner.nextFloat();
        if ((gender.equals("MALE") && weight >= 55) || (gender.equals("FEMALE") &&
weight >= 50)) {
            break;
        } else {
            System.out.println("Minimum weight for donating blood: Male - 55kg, Female - 50kg.");
            System.out.print("Do you want to continue (Y/N)? ");
            String continueChoice = scanner.next().toUpperCase();
            if (!continueChoice.equals("Y")) {
                System.out.println("Registration cancelled.");
                return;
            }
        }
    }

    System.out.print("Enter phone number: ");
    String phoneNumber = scanner.next();

    System.out.print("Enter address: ");
    scanner.nextLine();
    String address = scanner.nextLine().toUpperCase();

    try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD)) {
        String insertSQL = "INSERT INTO donors (name, bloodgroup, gender, weight, phonenumber, address, age) VALUES (?, ?, ?, ?, ?, ?, ?)";
        try (PreparedStatement preparedStatement = connection.prepareStatement(insertSQL)) {
            preparedStatement.setString(1, name);
            preparedStatement.setString(2, bloodGroup);
            preparedStatement.setString(3, gender);
            preparedStatement.setFloat(4, weight);
            preparedStatement.setString(5, phoneNumber);
            preparedStatement.setString(6, address);
            preparedStatement.setInt(7, age);
        preparedStatement.executeUpdate();
        System.out.println("Donor registered successfully!");
      }
    } catch (SQLException e) {
      e.printStackTrace();
    }
}

private static void searchDonor(Scanner scanner) {
    System.out.print("Enter blood group to search: ");
    String searchBloodGroup = scanner.next().toUpperCase();

    try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
        String selectSQL = "SELECT * FROM donors WHERE bloodgroup = ?";
        try (PreparedStatement preparedStatement =
connection.prepareStatement(selectSQL)) {
            preparedStatement.setString(1, searchBloodGroup);
            ResultSet resultSet = preparedStatement.executeQuery();

            System.out.println("Donors with Blood Group " + searchBloodGroup + ":");
            System.out.println("\n");
            while (resultSet.next()) {
                System.out.println("Name: " + resultSet.getString("name") +"\n"+
                " Age: " + resultSet.getInt("age")+"\n"+
                " Gender: " + resultSet.getString("gender") +"\n"+
                " Weight: " + resultSet.getFloat("weight") +"\n"+
                " Phone Number: " + resultSet.getString("phonenumber") +"\n"+
                ", Address: " + resultSet.getString("address"));
        // Print one-line gap for better separation between each line of donor information
        System.out.println();
        System.out.println("\n");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static boolean isValidBloodGroup(String bloodGroup) {
    String[] relevantBloodGroups = {"A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-"};
    for (String bg : relevantBloodGroups) {
        if (bg.equals(bloodGroup)) {
            return true;
        }
    }
    return false;
}
}
```

OUTPUT:

DONOR REGISTRATION :

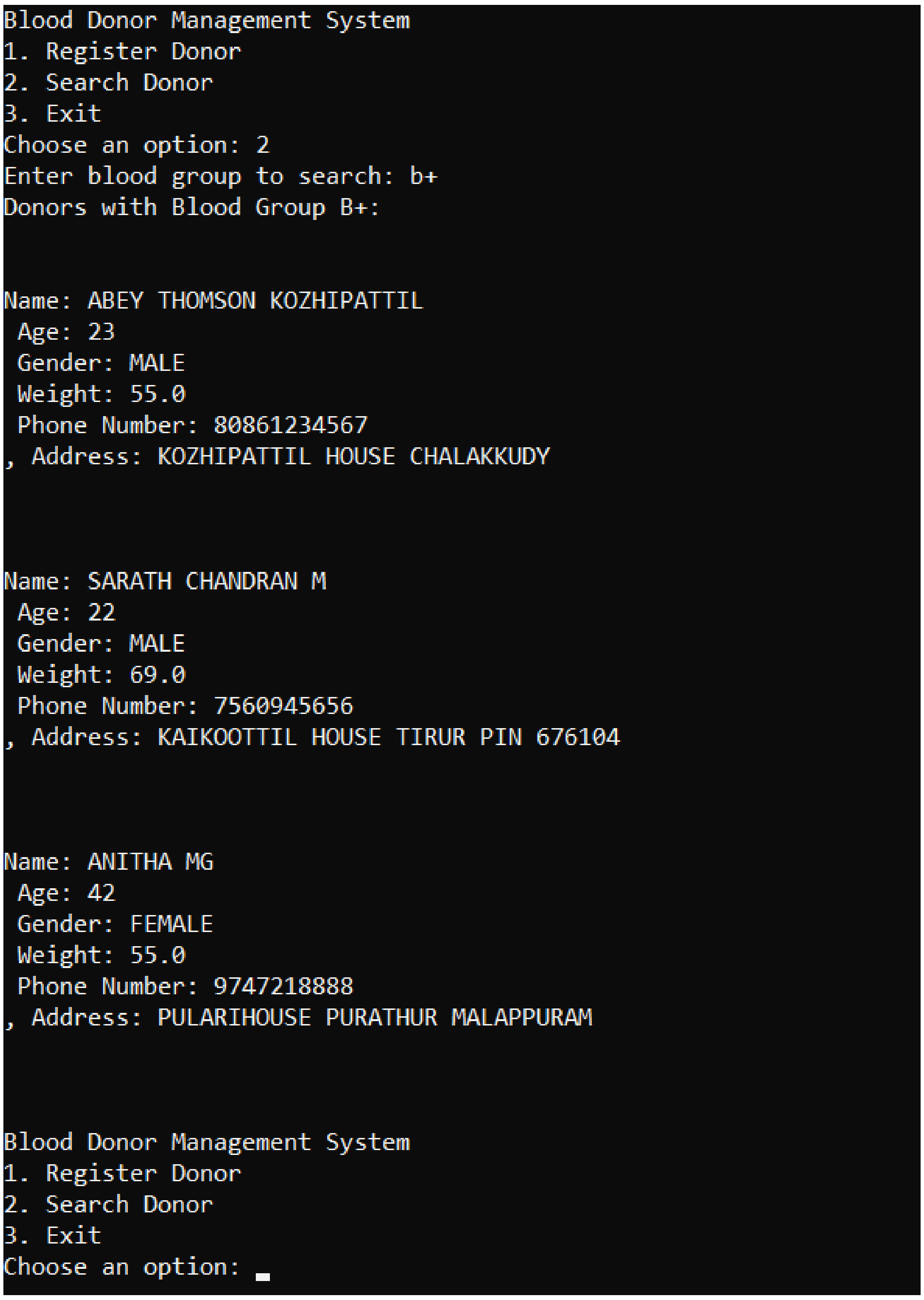


```
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sarath\Desktop\jdbcf-20240215T181628Z-001\jdbcf>javac blood.java

C:\Users\sarath\Desktop\jdbcf-20240215T181628Z-001\jdbcf>java blood
Connecting to database...
Connected to database successfully!
Table created successfully!
Blood Donor Management System
1. Register Donor
2. Search Donor
3. Exit
Choose an option: 1
Enter donor name: anitha mg
Enter blood group (A+, A-, B+, B-, AB+, AB-, O+, O-): b+
Enter age: 42
Enter gender (Male/Female): female
Enter weight (kg): 55
Enter phone number: 9747218888
Enter address: pularihouse purathur malappuram
Donor registered successfully!
Blood Donor Management System
1. Register Donor
2. Search Donor
3. Exit
Choose an option:
```



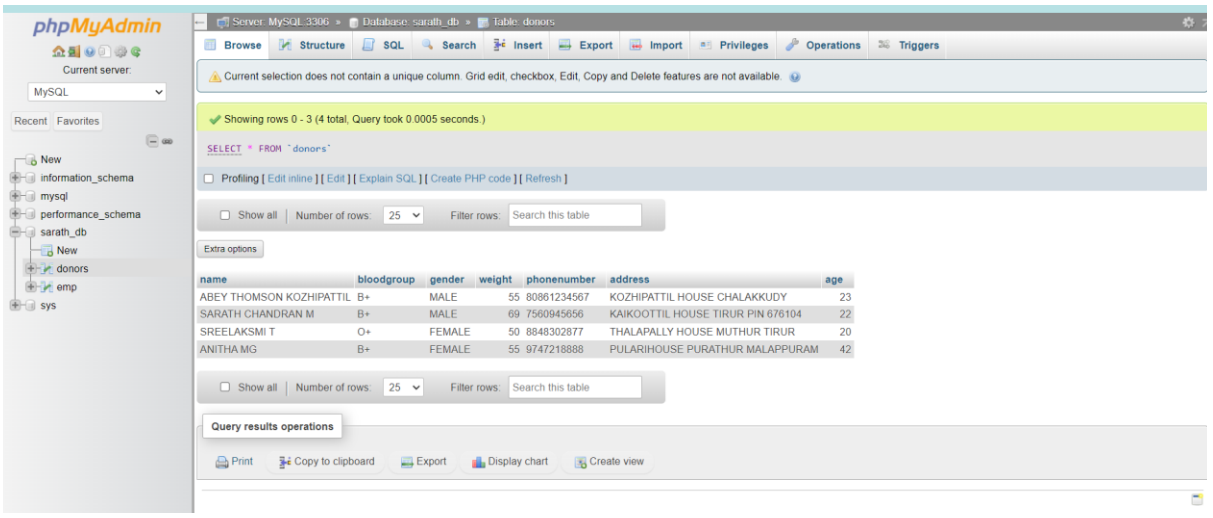
```
Choose an option: 1
Enter donor name: sreejith
Enter blood group (A+, A-, B+, B-, AB+, AB-, O+, O-): a+
Enter age: 17
Age must be 18 or above for blood donation.
Do you want to continue (Y/N)? n
Registration cancelled.
Blood Donor Management System
1. Register Donor
2. Search Donor
3. Exit
Choose an option: 1
Enter donor name: amal tom
Enter blood group (A+, A-, B+, B-, AB+, AB-, O+, O-): o-
Enter age: 21
Enter gender (Male/Female): male
Enter weight (kg): 50
Minimum weight for donating blood: Male - 55kg, Female - 50kg.
Do you want to continue (Y/N)? n
Registration cancelled.
Blood Donor Management System
1. Register Donor
2. Search Donor
3. Exit
```

RETRIEVING DONOR INFORMATION :



```
Blood Donor Management System
1. Register Donor
2. Search Donor
3. Exit
Choose an option: 2
Enter blood group to search: b+
Donors with Blood Group B+:


Name: ABEY THOMSON KOZHIPATTIL
 Age: 23
 Gender: MALE
 Weight: 55.0
 Phone Number: 80861234567
, Address: KOZHIPATTIL HOUSE CHALAKKUDY



Name: SARATH CHANDRAN M
 Age: 22
 Gender: MALE
 Weight: 69.0
 Phone Number: 7560945656
, Address: KAIKOOTTIL HOUSE TIRUR PIN 676104



Name: ANITHA MG
 Age: 42
 Gender: FEMALE
 Weight: 55.0
 Phone Number: 9747218888
, Address: PULARIHOUSE PURATHUR MALAPPURAM



Blood Donor Management System
1. Register Donor
2. Search Donor
3. Exit
Choose an option: _
```

DATABASE :



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 3 (4 total, Query took 0.0005 seconds.)

```
SELECT * FROM `donors`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

name	bloodgroup	gender	weight	phonenumber	address	age
ABEY THOMSON KOZHIPATTIL	B+	MALE	55	80861234567	KOZHIPATTIL HOUSE CHALAKKUDY	23
SARATH CHANDRAN M	B+	MALE	69	7560945656	KAIKOOTTIL HOUSE TIRUR PIN 676104	22
SREELAKSMI T	O+	FEMALE	50	8848302877	THALAPALLY HOUSE MUTHUR TIRUR	20
ANITHA MG	B+	FEMALE	55	9747218888	PULARIHOUSE PURATHUR MALAPPURAM	42

Show all | Number of rows: 25 Filter rows: Search this table