



and Project

ECS2301 Software Engineering

Mid term project (100 marks)

Instructions:

- ★ If your SLTC student number ends with 0,1,2,3 then you must attempt question 1.
- ★ If your SLTC student number ends with 4,5,6 then you must attempt question 2. ★ If your SLTC student number ends with 7,8,9 then you must attempt question 3.
- ★ Submit your answers as a single file (**.ZIP**) on or before the deadline provided in the LMS.
- ★ Submission must include this document explaining your code, output screens and lessons learnt
- ★ Late submission will not be considered for the marking.
- ★ Make sure to include this document in your submission

Important notice:

Please note that you need to write minimal code (stub functions). This is to prove your logic and navigation is working. You do NOT need to complete the calculations. That will be completed during the final project.

Full name: M.I Rasan Ahmed

Student index: 22ug2-0065

Date of submission: 1/14/2024

Q2. Blood pressure monitor

Your blood pressure is recorded as two numbers:

- Systolic blood pressure (the first number) – indicates how much pressure your blood is exerting against your artery walls when the heart contracts.
- Diastolic blood pressure (the second number) – indicates how much pressure your blood is exerting against your artery walls while the heart muscle is resting between contractions.

Blood pressure categories

The five blood pressure ranges as recognized by the American Heart Association are:

1. Normal

Blood pressure numbers of less than 120/80 mm Hg (millimeters of mercury) are considered within the normal range.

2. Elevated

Elevated blood pressure is when readings consistently range from 120-129 systolic and less than 80 mm Hg diastolic.

3. Hypertension Stage 1

Hypertension Stage 1 is when blood pressure consistently ranges from 130 to 139 systolic or 80 to 89 mm Hg diastolic.

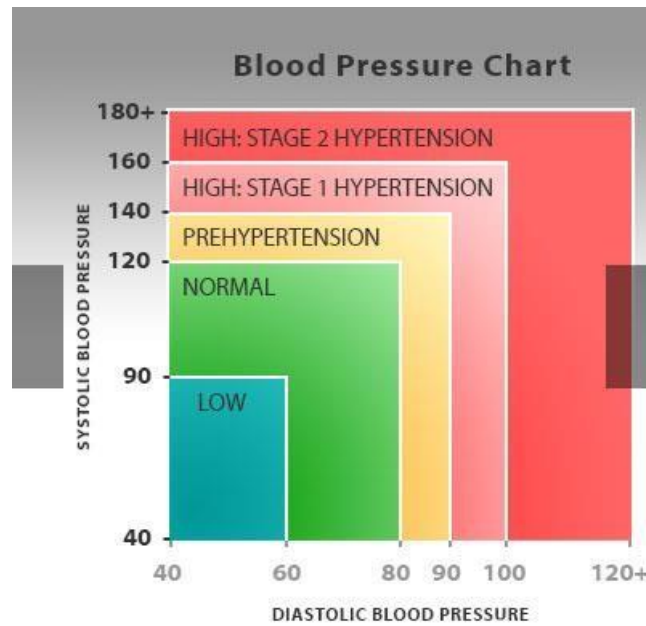
4. Hypertension Stage 2




Hypertension Stage 2 is when blood pressure consistently is 140/90 mm Hg or higher.

5. Hypertensive crisis

This stage of high blood pressure requires medical attention.

Watch this video for more details: <https://youtu.be/o8DX89jm710>



	Age	Min	Normal	Max
	1 to 12 months	75 / 50	90 / 60	100 / 75
	1 to 5 years	80 / 55	95 / 65	110 / 79
	6 to 13 years	90 / 60	105 / 70	115 / 80
	14 to 19 years	105 / 73	117 / 77	120 / 81
	20 to 24 years	108 / 75	120 / 79	132 / 83
	25 to 29 years	109 / 76	121 / 80	133 / 84
	30 to 34 years	110 / 77	122 / 81	134 / 85
	35 to 39 years	111 / 78	123 / 82	135 / 86
	40 to 44 years	112 / 79	125 / 83	137 / 87
	45 to 49 years	115 / 80	127 / 84	139 / 88
	50 to 54 years	116 / 81	129 / 85	142 / 89
	55 to 59 years	118 / 82	131 / 86	144 / 90
	60 to 64 years	121 / 83	134 / 87	147 / 91

©idealbloodpressureinfo.com

Blood pressure shown as systolic/diastolic

Requirements:

Tester
-id (int) -name (String) -yob (year of birth) -systolic (int) -diastolic (int)
+main() +displayMenu() +index() +view(int id) +create() +delete() +exit()

Tester class diagram

BloodPressure
-id (int) -name (String) -yob (year of birth) -systolic (int) -diastolic (int)
+bloodPressure() (constructor) +setters/getters for properties +calculate() +display()

Blood pressure class diagram

Write an application to store blood pressure of 5 users. It should be a menu driven application. You need to show the evidence of using classes, objects, methods, properties, setters/getters, constructors, abstraction, inheritance, polymorphism and java collections.

- a) In your main class create a displayMenu() method and show the following choices. The program must continuously run until 'exit' is selected (10 marks) :
 - i. Create a record. (Ask for user id, then ask data for name, year of birth, systolic & diastolic)
 - ii. Show blood pressure data for all users
 - iii. Show blood pressure data for a selected user.
 - iv. Delete all
 - v. Exit application
- b) Write methods for all actions above, inside main class (10 marks for each)
 - i. index() : to show all records. Call BloodPressure.display() here.
 - ii. view(int id) : to show one record for the given id. Call BloodPressure.display() here.
 - iii. create() : create a new record. Call BloodPressure constructor here.
 - iv. delete() : delete all records
 - v. exit() : exit to system
- c) A section on the lessons learnt during this exercise (20 marks)
- d) Generate JavaDoc files (10 marks)
- e) Upload your complete, commented, tested code as a Git Hub repository to /midterm branch. Include the link in the document (10 marks)

Paste the code here

GitHub repo url:

```
import java.util.ArrayList;
import java.util.Scanner;
```

```
class BloodPressure{
    private int id,yob,systolic,diastolic;
    private String name;

    public BloodPressure(int ID,String Name,int Yob,int Systolic,int Diastolic){

        this.id = ID;
        this.yob = Yob;
        this.diastolic = Diastolic;
        this.systolic = Systolic;
        this.name = Name;
    }
    public int getId(){
        return id;
    }

    public String getName(){
        return name;
    }
    public int getyob(){
        return yob;
    }

    public int getsystolic(){
        return systolic;
    }

    public int getDiastolic(){
        return diastolic;
    }

    public void setId(int i){
        id = i;
    }

    public void setName(String N){
        name = N;
    }

    public void setyob(int Y){
```

```

        yob = Y;
    }

    public void setsystolic(int S){
        systolic = S;
    }

    public void setDiastolic(int D){
        diastolic = D;
    }

    public double calculate(int age){
        age = 2024-yob;
        return age;
    }

    public void display(){
        System.out.println("Id : "+id);
        System.out.println("Name : "+name);
        System.out.println("Year of Birth : "+yob);
        System.out.println("Systolic : "+systolic);
        System.out.println("Diastolic : "+diastolic);
    }

}

/**
 *
 * @author Rasan
 */
public class Tester {
    private static BloodPressure record(){
        Scanner R = new Scanner(System.in);
        System.out.print("Enter Your ID: ");
        int id = R.nextInt();
        System.out.print("Enter Your Name: ");
        String name = R.next();
        System.out.print("Enter Your Year of Birth: ");
        int yob = R.nextInt();
        System.out.print("Enter The Systolic : ");
        int systolic = R.nextInt();
        System.out.print("Enter The Diastolic : ");
        int diastolic = R.nextInt();
        return new BloodPressure(id,name,yob,systolic, diastolic);
    }

}

```

```

private static int getId() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter ID: ");
    return scanner.nextInt();
}

private static BloodPressure getbpBYId(ArrayList<BloodPressure> bloodpressureRecords, int id) {
    for (BloodPressure bloodpressure : bloodpressureRecords) {
        if (bloodpressure.getId() == id) {
            return bloodpressure;
        }
    }
    return null;
}

public static void displayMenu(){
    System.out.println(" Blood pressure monitor");
    System.out.println("*-----*");
    System.out.println("1. Create a record.");
    System.out.println("2. Show blood pressure data for all users.");
    System.out.println("3. Show blood pressure data for a selected user.");
    System.out.println("4. Delete all.");
    System.out.println("5. Exit application.");

    System.out.print("Enter Your Choise : ");
}

public static void index(){
    System.out.println("to show all records");
}
public static void view(int Id){
    System.out.println(" to show one record for the given id.");
}

public static void create(){
    System.out.println("create a new record");
}

public static void delete(){
    System.out.println("All Blood pressure deleted successfully");
}

public static void exit(){
    System.out.println("Thank you for using The Blood pressure monitor");
}

```

```

        System.exit(0);
    }

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    ArrayList <BloodPressure> bloodpressure = new ArrayList<>();
    int choice;
    Scanner in = new Scanner(System.in);
    do{
        displayMenu();
        choice = in.nextInt();
        switch(choice){
            case 1:
                create();
                for(int j=0 ; j < 5 ; j++){
                    bloodpressure.add(record());
                    System.out.println("-----");
                }
                break;

            case 2:
                index();
                for(BloodPressure BP : bloodpressure){
                    BP.display();
                    System.out.println("-----");
                }

                break;

            case 3:
                int id = getId();
                view(id);
                BloodPressure bloodpressureToDisplay = getbpBYId(bloodpressure, id);
                if (bloodpressureToDisplay != null) {
                    bloodpressureToDisplay.display();
                    System.out.println("-----");
                }

                else {
                    System.out.println("No bloodpressure record found for the given ID.");
                }
            }
        }
    }
}

```



```
        break;

    case 4:
        delete();
        bloodpressure.clear();
        break;

    case 5:
        exit();
        break;

    default:
        System.out.println("Invalid Choise");

    }

    }
while(true);

}

}
```

Paste the output screens here

```
1. Create a record.
2. Show blood pressure data for all users.
3. Show blood pressure data for a selected user.
4. Delete all.
5. Exit application.
Enter Your Choise : 1
create a new record
Enter Your ID: 0
Enter Your Name: rasan
Enter Your Year of Birth: 2001
Enter The Systolic : 120
Enter The Diastolic : 79
-----
Enter Your ID: 1
Enter Your Name: asvi
Enter Your Year of Birth: 2001
Enter The Systolic : 108
Enter The Diastolic : 75
-----
Enter Your ID: 2
Enter Your Name: rais
Enter Your Year of Birth: 2000
Enter The Systolic : 120
Enter The Diastolic : 81
-----
Enter Your ID: 3
Enter Your Name: atheef
Enter Your Year of Birth: 2012
Enter The Systolic : 105
Enter The Diastolic : 70
-----
```

```
Id : 4
Name : waseem
Year of Birth : 1960
Systolic : 147
Diastolic : 91
-----
Blood pressure monitor
*-----*
1. Create a record.
2. Show blood pressure data for all users.
3. Show blood pressure data for a selected user.
4. Delete all.
5. Exit application.
Enter Your Choise : 3
Enter ID: 2
to show one record for the given id.
Id : 2
Name : rais
Year of Birth : 2000
Systolic : 120
Diastolic : 81
-----
Blood pressure monitor
*-----*
1. Create a record.
2. Show blood pressure data for all users.
3. Show blood pressure data for a selected user.
4. Delete all.
5. Exit application.
Enter Your Choise : 4
All Blood pressure deleted successfully
Blood pressure monitor
*-----*
```

```

1. Create a record.
2. Show blood pressure data for all users.
3. Show blood pressure data for a selected user.
4. Delete all.
5. Exit application.
Enter Your Choise : 3
Enter ID: 2
    to show one record for the given id.
Id : 2
Name : rais
Year of Birth : 2000
Systolic : 120
Diastolic : 81
-----
    Blood pressure monitor
*-----*
1. Create a record.
2. Show blood pressure data for all users.
3. Show blood pressure data for a selected user.
4. Delete all.
5. Exit application.
Enter Your Choise : 4
All Blood pressure deleted successfully
    Blood pressure monitor
*-----*
1. Create a record.
2. Show blood pressure data for all users.
3. Show blood pressure data for a selected user.
4. Delete all.
5. Exit application.
Enter Your Choise : 5
Thank you for using The Blood pressure monitor
BUILD SUCCESSFUL (total time: 3 minutes 44 seconds)
|

```

Lessons learnt

In your words, what did you learnt during this exercise?

Developing a blood pressure monitoring application in Java has imparted several key lessons in software design, object-oriented programming (OOP), and Java features. The exercise emphasized the importance of modular code organization, user-friendly interfaces, effective use of Java collections, and the practical application of OOP principles.

One crucial lesson was the significance of modular code. By creating distinct classes like `BloodPressure`, the code became more readable and maintainable. OOP principles, such as encapsulation and abstraction, were instrumental in achieving a modular and flexible design. This lesson highlighted the importance of structuring code for scalability and future modifications.

The exercise underscored the value of user-focused design through a menu-driven interface. Continuous execution until the user chooses to exit ensures a seamless and user-friendly experience, emphasizing the need to prioritize end-user perspectives in application development.

The application also introduced the practical use of Java collections, specifically the `HashMap`. Understanding and leveraging collections, in this case, facilitated efficient data management by mapping user IDs to corresponding blood pressure records. This lesson highlighted the importance of choosing appropriate data structures for effective information organization.

User input handling, employing the `Scanner` class, was a crucial aspect of the exercise. This not only improved user interaction but also necessitated input validation for reliability. Error handling mechanisms, such as checking for the existence of user IDs, contributed to the application's robustness.

While not explicitly implemented, the concepts of inheritance and polymorphism were implicitly emphasized. These concepts, integral to OOP, were recognized as powerful tools for creating modular and extensible code, allowing for shared functionalities and enhanced code reuse in more complex applications.

In summary, the blood pressure monitoring application exercise provided valuable insights into modular code organization, user-focused design, effective use of Java collections, and practical application of OOP principles. These lessons serve as foundational knowledge for future software development endeavors.