ECS2301 Software Engineering and Project

Final project (100 marks)

Instructions:

- This is a continuation of your mid-term exam project. You will complete the logic, input validation and output formatting here.
- 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

Full name: Achini Senanayake

Student index: 22ug2-0203

Date of submission: 2024/03/27

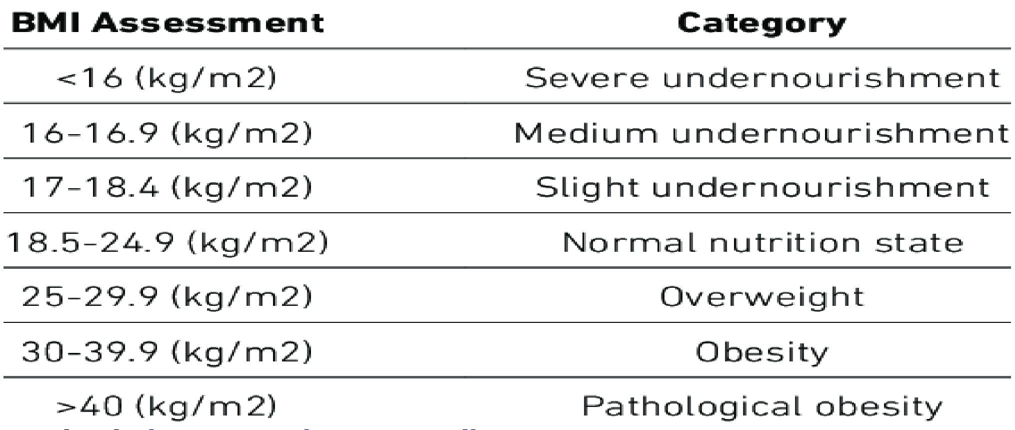
Q1. BMI calculator

BMI, or Body Mass Index, is a numerical value of a person's weight in relation to their height. It is a commonly used screening tool to categorize individuals into different weight status categories, such as underweight, normal weight, overweight, and obesity.

BMI is calculated using the following formula:

$$BMI = \frac{\text{weight in kilograms}}{(\text{height in meters})^2}$$

Here's a breakdown of the BMI categories:



BMI Assessment	Category
<16 (kg/m2)	Severe undernourishment
16-16.9 (kg/m2)	Medium undernourishment
17-18.4 (kg/m2)	Slight undernourishment
18.5-24.9 (kg/m2)	Normal nutrition state
25-29.9 (kg/m2)	Overweight
30-39.9 (kg/m2)	Obesity
>40 (kg/m2)	Pathological obesity

Watch this video for more details: https://youtu.be/t8sIioCX0lk.

Requirements:

Tester
-id (int) -name (String) -yob (year of birth) -height (int) -weight (int)

+main()
+displayMenu()
+index()
+view(int id)
+create()
+delete()
+exit()

Tester class diagram

Bmi
-id (int) -name (String) -yob (year of birth) -height (int) -weight (int)

+bmi() (constructor)
+setters/getters for properties
+calculate()
+display()

Bmi class diagram

Write an application to store BMI 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, height & weight)

    ii. Show BMI data for all users.

    iii. Show BMI 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 Bmi.display() here.

    ii. view(int id) : to show one record for the given id. Include BMI category and recommendations if there are any. Call Bmi.display() here.

    iii. create() : create a new record with appropriate input validation. Call Bmi constructor here.

    iv. delete() : delete all records.

    v. exit() : exit to system

c) Use a suitable Java collection to store 5 users (10 marks)

d) Calculate BMI accurately, taking into consideration height in cm (5 marks)

e) Calculate the age with year of birth and current year (5 marks)

f) A section on the lessons learnt during this exercise (10 marks)

g) Generate JavaDoc files (5 marks)

h) Upload your complete, commented, tested code as a Git Hub repository to `/final` branch. Include the link in the document (5 marks)

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.

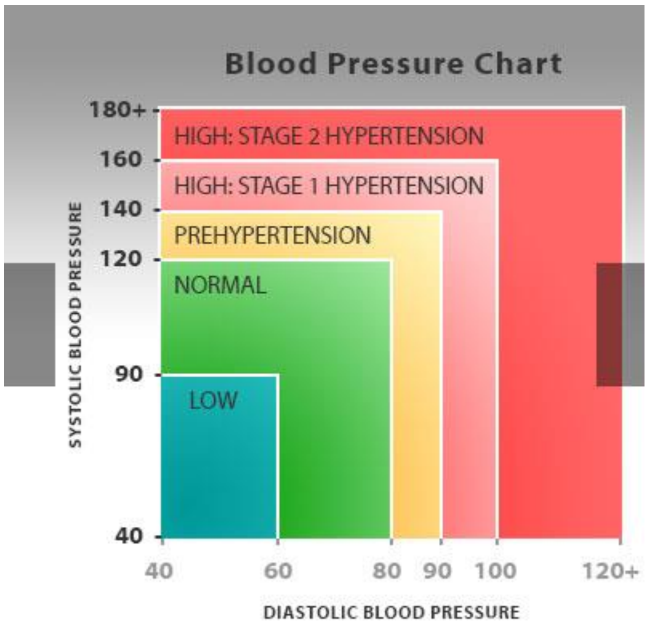
4. Hypertension Stage 2

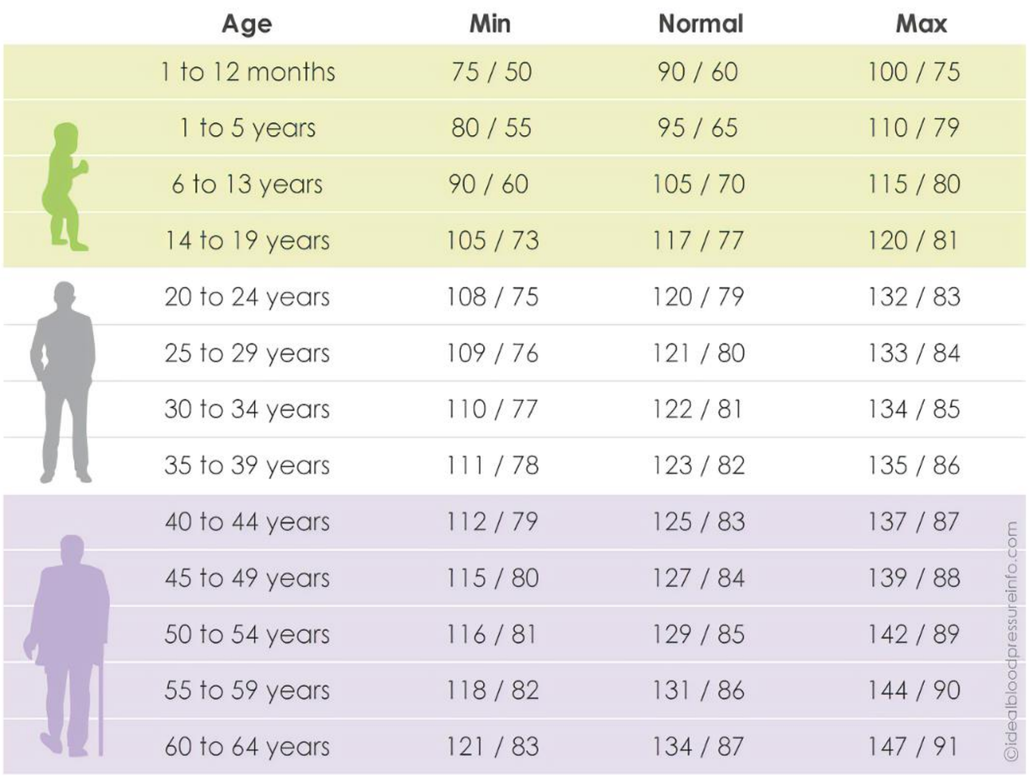
Hypertension Stage 2 is when blood pressure consistently is 140/90 mm Hg or higher.

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

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 data.)

   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. Include blood pressure category and recommendations if there are any for their age. Call BloodPressure.display() here.

   iii. create() : create a new record with appropriate input validation. Call BloodPressure constructor here.

   iv. delete() : delete all records

   v. exit() : exit to system

c) Use a suitable Java collection to store 5 users (10 marks)

d) Calculate the age with year of birth and current year (10 marks)

e) A section on the lessons learnt during this exercise (10 marks)

f) Generate JavaDoc files (5 marks)

g) Upload your complete, commented, tested code as a Git Hub repository to `/final` branch. Include the link in the document (5 marks)

Q3. Blood sugar monitor

Glucose comes from the food we eat and its sugar content. When a person consumes a food with high sugar content, that is turned into glucose. The glucose is then absorbed into the bloodstream with the support of insulin. This is then distributed between the body's cells and used as energy.

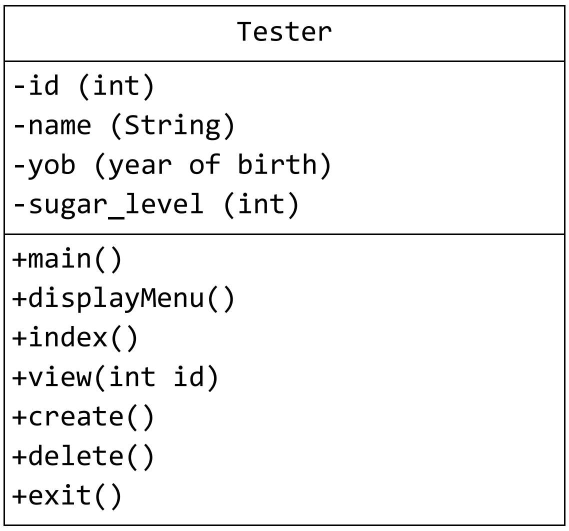
Diabetes is a chronic health condition that occurs when the body is unable to properly regulate blood sugar (glucose) levels. According to the International Diabetes Federation (IDF) and the World Health Organization (WHO), diabetes was estimated to be responsible for around 4 million deaths globally in the year 2019.

This chart details goals for specific groups of people with diabetes.

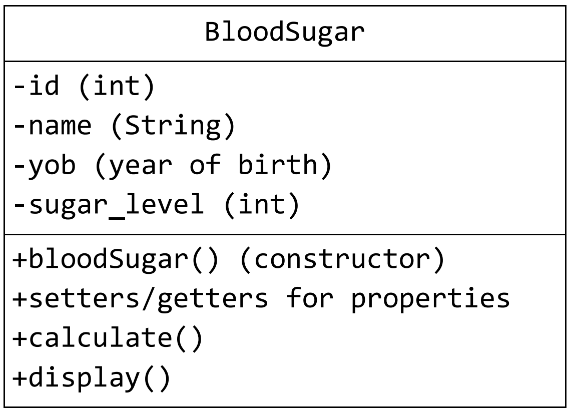
	Before meals (fasting)	**After meals (post-prandial)**	**Other**
Adults with type 1 diabetes	80–130 mg/dL	< 180 mg/dL (1 or 2 hours after)	
Adults with type 2 diabetes	80–130mg/dL	< 180 mg/dL and (1 or 2 hours after)	
Children with type 1 diabetes	90-130 mg/dL		90–150 mg/dL at bedtime/overnight
Pregnant people (T1D, gestational diabetes)	< 95 mg/dL	140 mg/dL (1 hour after)	120 mg/dL (2 hours after)
65 or older	80–180 mg/dL		80–200 mg/dL for those in poorer health, assisted living, end of life
Without diabetes	99 mg/dL or below	140 mg/dL or below	

Watch this video for more details: https://youtu.be/O7l3qg0Z4GE

Requirements:



Tester class diagram



Blood sugar class diagram

Write an application to store blood sugar 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 and blood sugar level)

   ii. Show blood sugar data for all users

   iii. Show blood sugar 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 BloodSugar.display() here.

   ii. view(int id) : to show one record for the given id. Include blood sugar category and recommendations if there are any. Call BloodSugar.display() here.

   iii. create() : create a new record with appropriate input validation. Call BloodSugar constructor here.

   iv. delete() : delete all records

   v. exit() : exit to system

c) Use a suitable Java collection to store 5 users (10 marks)

d) Calculate the age with year of birth and current year (10 marks)

e) A section on the lessons learnt during this exercise (10 marks)

f) Generate JavaDoc files (5 marks)

g) Upload your complete, commented, tested code as a Git Hub repository to `/final` branch. Include the link in the document (5 marks)

Paste the code here

GitHub repo url:

<u>Question number 1</u>

- **<u>Tester.java codes</u>**

```
package package1;


import java.lang.ref.Cleaner;


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


public class Tester {


    public static int id;
    static String name;
    static String yob;
    static int height;
    static int weight;
    static Bmi bmi;
    public static List<Bmi> users = new ArrayList<>();
    public static Scanner scanner = new Scanner(System.in);


    public static void main(String[] args) {
```

```
    int choice;

    do {
        displayMenu();
        choice = scanner.nextInt();
        switch (choice) {
            case 1:
                create();
                break;
            case 2:
                index();
                break;
            case 3:
                System.out.println("Enter any user id: ");
                int choseId = scanner.nextInt();
                view(choseId);
                break;
            case 4:
                delete();
                break;
            case 5:
                exit();
                break;
            default:
                System.out.println("Please enter a correct choice");
                ;
        }
    } while (true);

}

public static void displayMenu() {
        System.out.println("Welcome: BMI CALCULATOR");
        System.out.println("");
        System.out.println("1.Create a record");
        System.out.println("2.Show BMI data for all users. ");
        System.out.println("3.Show BMI data for a selected user");
        System.out.println("4.Delete all records ");
        System.out.println("5.Exit ");
        System.out.println("");
        System.out.println("Please Enter Your choice ");

    }

    public static void view(int choseId) {

        String stringId = String.valueOf(choseId);

        for (Bmi user : users) {
            if (String.valueOf(user.getId()).equals(stringId)) {
                System.out.println("Showing data for user id: " + choseId);
                System.out.println("");
                System.out.println("User name: " + user.getName());
                System.out.println("User age: " + user.getAge());
                System.out.println("User BMI: " + user.getBmi());
                System.out.println("User BMI category is: " + user.getCategory());
                System.out.println("Recommendation :" + user.getRecommendation());

                System.out.println("");

            } else {
                System.out.println("No matching user found. Try another id");
            }
        }
        bmi.display();

    }

    public static void create() {

        System.out.println("Enter your id:");
        id = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Enter your name:");
        name = scanner.nextLine();
        System.out.println("Enter your yob:");
        yob = scanner.nextLine();
        System.out.println("Enter your height in centi meters:");
        height = scanner.nextInt();
        System.out.println("Enter your weight in kg:");
        weight = scanner.nextInt();
        System.out.println("");
        if (id <= 0 || id >= 6) {
            System.out.println("Id must be greater than 0 and less than 5");
            System.out.println("Enter your id:");
            id = scanner.nextInt();

            if (Integer.parseInt(yob) >= 2025) {
                System.out.println("Year of birth must be less than 2025");
                System.out.println("Enter your yob:");
                int yobnum = scanner.nextInt();
                yob = String.valueOf(yobnum);
                bmi = new Bmi(id, name, yob, height, weight);
                bmi.calculate();
                bmi.display();
            users.add(bmi);
        } else {
            bmi = new Bmi(id, name, yob, height, weight);
            bmi.calculate();
            bmi.display();
            users.add(bmi);

        }

    } else if (Integer.parseInt(yob) >= 2025) {
        System.out.println("Year of birth must be less than 2025");
        System.out.println("Enter your yob:");
        int yobnum = scanner.nextInt();
        yob = String.valueOf(yobnum);
        bmi = new Bmi(id, name, yob, height, weight);
        bmi.calculate();
        bmi.display();
        users.add(bmi);

    } else {
        bmi = new Bmi(id, name, yob, height, weight);
        bmi.calculate();
        bmi.display();
        users.add(bmi);
    }

}

public static void index() {
    if (!String.valueOf(users.size()).equalsIgnoreCase("0")) {
        for (Bmi user : users) {
            System.out.println("Showind all data of " + user.getName());
            System.out.println("User id: " + user.getId());
            System.out.println("User name: " + user.getName());
            System.out.println("User age: " + user.getAge());
            System.out.println("User BMI: " + user.getBmi());
            System.out.println("User BMI category is: " + user.getCategory());
            System.out.println("Recommendation :" + user.getRecommendation());
            System.out.println("");
        }
    } else {
        System.out.println("No users found.Please create a record first.");
    }
    bmi.display();
}

public static void delete() {
    users.clear();
    System.out.println("Deleted all user records");

}

public static void exit() {
    System.out.println("Exiting system");
    System.exit(0);
}

}
```

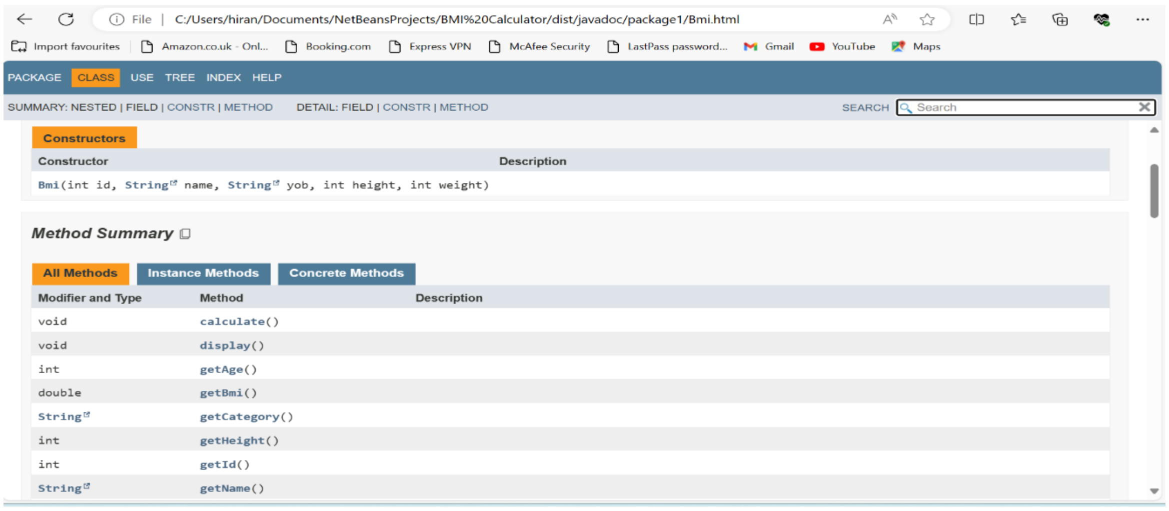
- **Bmi.java code**

```
package package1;

import java.time.Year;
```

```java
public class Bmi {

    int id;
    String name;
    String yob;
    int height;
    int weight;
    double bmi;
    String category;
    String recommendation;
    int age;

    public Bmi(int id, String name, String yob, int height, int weight) {
        this.id = id;
        this.name = name;
        this.yob = yob;
        this.height = height;
        this.weight = weight;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getYob() {
        return yob;
    }

    public void setYob(String yob) {
        this.yob = yob;
    }

    public int getHeight() {
        return height;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    public double getBmi() {
        return bmi;
    }

    public String getCategory() {
        return category;
    }
```

```
    public int getAge() {
        return age;
    }

    public String getRecommendation() {
        return recommendation;
    }



    public void calculate() {
        double heightInM = height / 100;
        double heightSq = heightInM * heightInM;
        bmi = weight / heightSq;

        age = 2024 - Integer.parseInt(yob);
    }

    public void display() {
        if (bmi < 16) {
            category = "Severe undernourishment";
            recommendation = "you need Immediate medical attention and nutritional support";
        } else if (bmi > 16 && bmi < 16.9) {
            category = "Medium undernourishment";
            recommendation ="you need balanced diet rich in nutrients, supplemented with regular
meals and hydration";
        } else if (bmi > 17 && bmi < 18.4) {
            category = "Slight undernourishment";
            recommendation = "focus on increasing nutrient-dense foods in your diet";
        } else if (bmi > 18.5 && bmi < 24.9) {
            category = "Normal nutrition state";
            recommendation = "continue to prioritize a balanced diet consisting";
        } else if (bmi > 25 && bmi < 29.9) {
            category = "Overweight";
            recommendation = "need regular physical activity and seeking support from healthcare
professionals ";
        } else if (bmi > 30 && bmi < 39.9) {
            category = "Obesity";
            recommendation = "do regular exercise and make behavioral changes";
        } else if (bmi > 40) {
            category = "Pathological obesity";
            recommendation = "seek immediate medical attention and guidance from healthcare
professionals";
        }
    }

}
```

Paste the output screens here


File | C:/Users/hiran/Documents/NetBeansProjects/BMI%20Calculator/dist/javadoc/package1/Bmi.html
Import favourites
Amazon.co.uk - Onl...
Booking.com
Express VPN
McAfee Security
LastPass password...
Gmail
YouTube
Maps
PACKAGE CLASS USE TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH Search

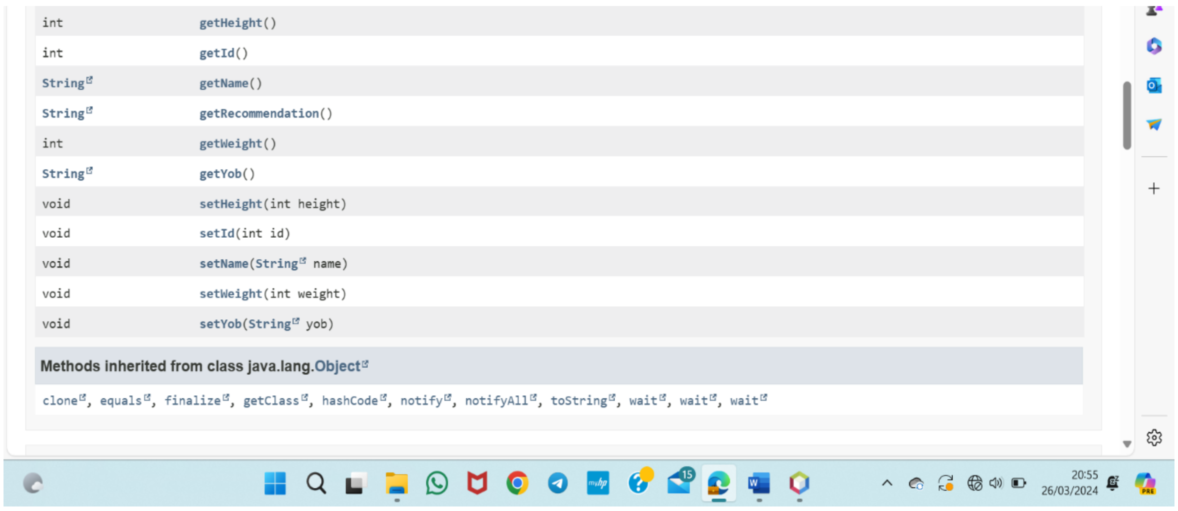
Constructors

Constructor	Description
Bmi(int id, String name, String yob, int height, int weight)	

Method Summary

All Methods | Instance Methods | Concrete Methods

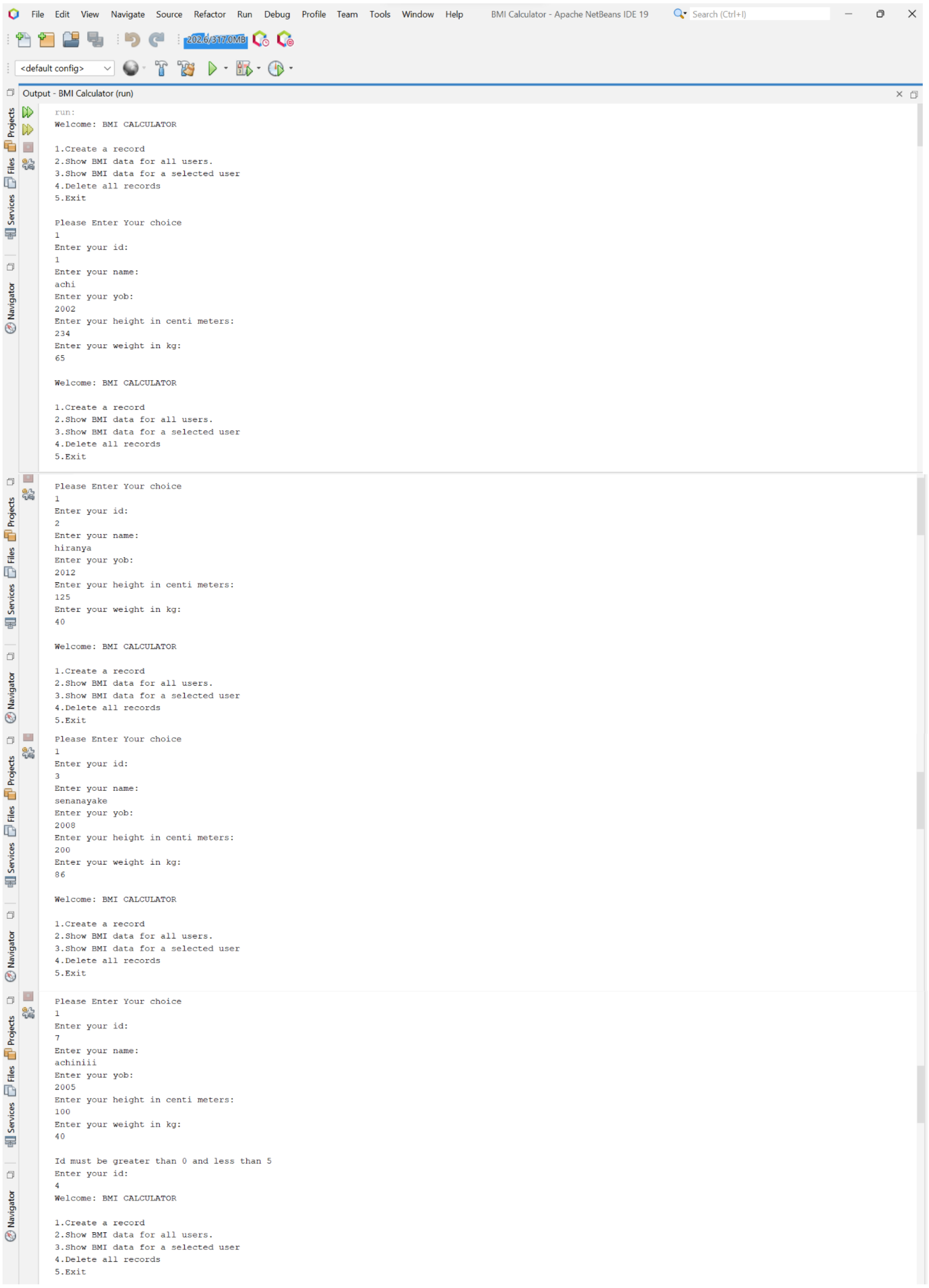
Modifier and Type	Method	Description
void	calculate()	
void	display()	
int	getAge()	
double	getBmi()	
String	getCategory()	
int	getHeight()	
int	getId()	
String	getName()	



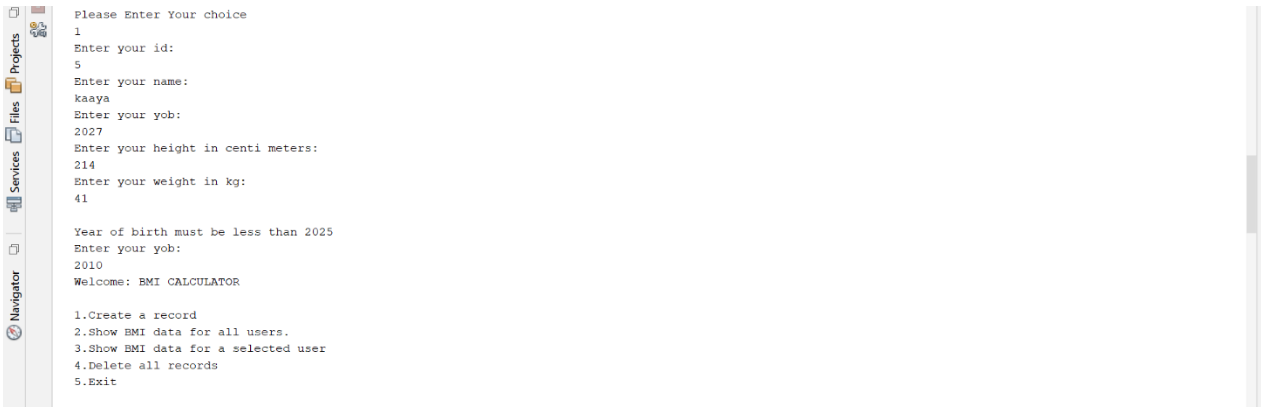
int	getHeight()
int	getId()
String	getName()
String	getRecommendation()
int	getWeight()
String	getYob()
void	setHeight(int height)
void	setId(int id)
void	setName(String name)
void	setWeight(int weight)
void	setYob(String yob)

Methods inherited from class java.lang.Object

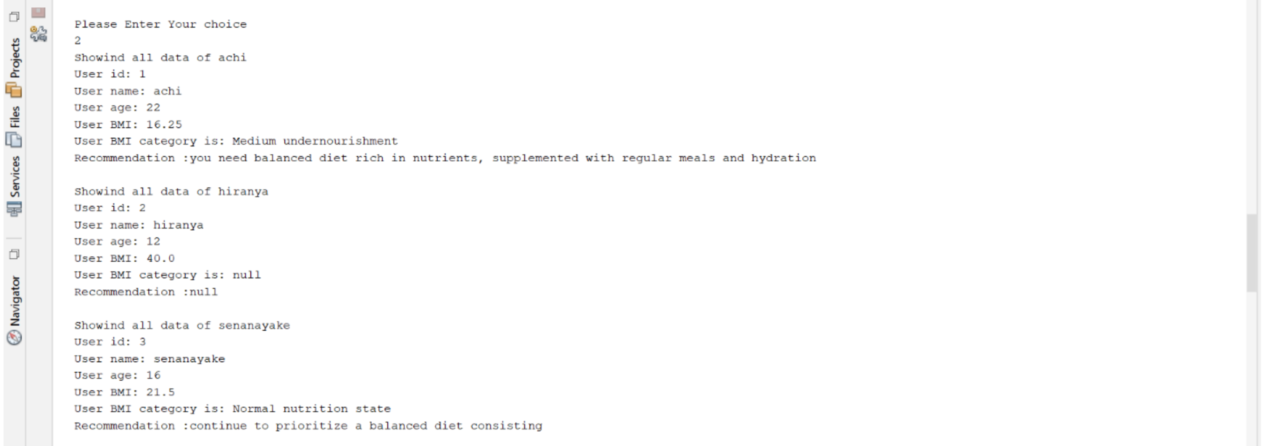
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
BMI Calculator - Apache NetBeans IDE 19
Search (Ctrl+I)
<default config>
Output - BMI Calculator (run)
Projects
Files
Services
Navigator

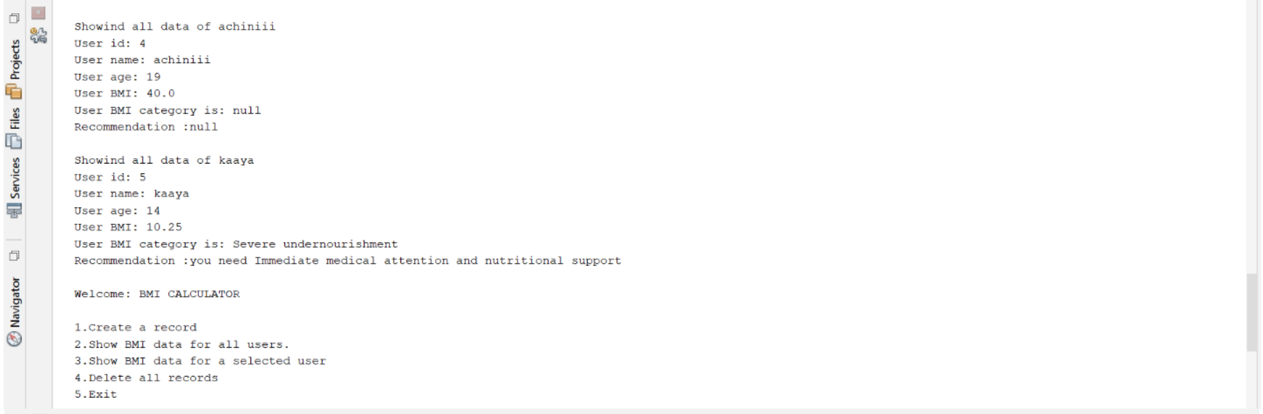
```
run:
Welcome: BMI CALCULATOR

1.Create a record
2.Show BMI data for all users.
3.Show BMI data for a selected user
4.Delete all records
5.Exit

Please Enter Your choice
1
Enter your id:
1
Enter your name:
achi
Enter your yob:
2002
Enter your height in centi meters:
234
Enter your weight in kg:
65

Welcome: BMI CALCULATOR

1.Create a record
2.Show BMI data for all users.
3.Show BMI data for a selected user
4.Delete all records
5.Exit

Please Enter Your choice
1
Enter your id:
2
Enter your name:
hiranya
Enter your yob:
2012
Enter your height in centi meters:
125
Enter your weight in kg:
40

Welcome: BMI CALCULATOR

1.Create a record
2.Show BMI data for all users.
3.Show BMI data for a selected user
4.Delete all records
5.Exit

Please Enter Your choice
1
Enter your id:
3
Enter your name:
senanayake
Enter your yob:
2008
Enter your height in centi meters:
200
Enter your weight in kg:
86

Welcome: BMI CALCULATOR

1.Create a record
2.Show BMI data for all users.
3.Show BMI data for a selected user
4.Delete all records
5.Exit

Please Enter Your choice
1
Enter your id:
7
Enter your name:
achiniii
Enter your yob:
2005
Enter your height in centi meters:
100
Enter your weight in kg:
40

Id must be greater than 0 and less than 5
Enter your id:
4
Welcome: BMI CALCULATOR

1.Create a record
2.Show BMI data for all users.
3.Show BMI data for a selected user
4.Delete all records
5.Exit
```



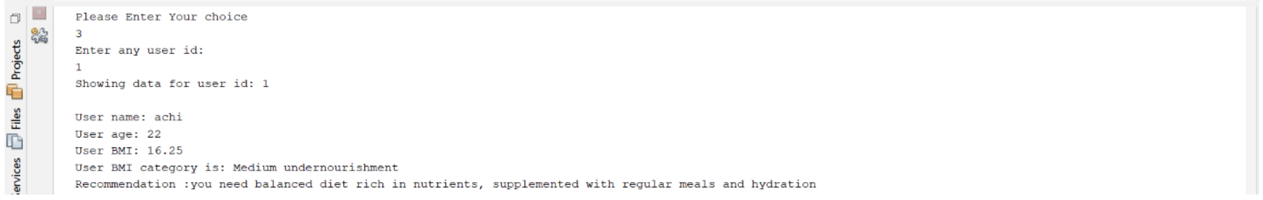
```
Please Enter Your choice
1
Enter your id:
5
Enter your name:
kaaya
Enter your yob:
2027
Enter your height in centi meters:
214
Enter your weight in kg:
41

Year of birth must be less than 2025
Enter your yob:
2010
Welcome: BMI CALCULATOR

1.Create a record
2.Show BMI data for all users.
3.Show BMI data for a selected user
4.Delete all records
5.Exit
```



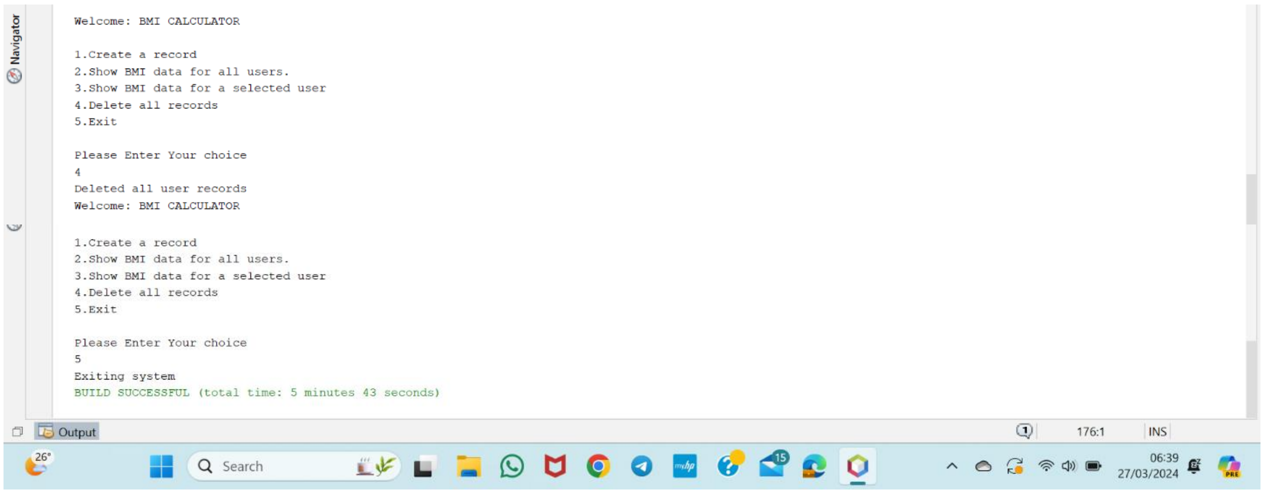
```
Please Enter Your choice
2
Showind all data of achi
User id: 1
User name: achi
User age: 22
User BMI: 16.25
User BMI category is: Medium undernourishment
Recommendation :you need balanced diet rich in nutrients, supplemented with regular meals and hydration

Showind all data of hiranya
User id: 2
User name: hiranya
User age: 12
User BMI: 40.0
User BMI category is: null
Recommendation :null

Showind all data of senanayake
User id: 3
User name: senanayake
User age: 16
User BMI: 21.5
User BMI category is: Normal nutrition state
Recommendation :continue to prioritize a balanced diet consisting
```



```
Showind all data of achiniii
User id: 4
User name: achiniii
User age: 19
User BMI: 40.0
User BMI category is: null
Recommendation :null

Showind all data of kaaya
User id: 5
User name: kaaya
User age: 14
User BMI: 10.25
User BMI category is: Severe undernourishment
Recommendation :you need Immediate medical attention and nutritional support

Welcome: BMI CALCULATOR

1.Create a record
2.Show BMI data for all users.
3.Show BMI data for a selected user
4.Delete all records
5.Exit
```



```
Please Enter Your choice
3
Enter any user id:
1
Showing data for user id: 1

User name: achi
User age: 22
User BMI: 16.25
User BMI category is: Medium undernourishment
Recommendation :you need balanced diet rich in nutrients, supplemented with regular meals and hydration
```



```
Welcome: BMI CALCULATOR

1.Create a record
2.Show BMI data for all users.
3.Show BMI data for a selected user
4.Delete all records
5.Exit

Please Enter Your choice
4
Deleted all user records
Welcome: BMI CALCULATOR

1.Create a record
2.Show BMI data for all users.
3.Show BMI data for a selected user
4.Delete all records
5.Exit

Please Enter Your choice
5
Exiting system
BUILD SUCCESSFUL (total time: 5 minutes 43 seconds)
```

Lessons learnt

- Scanner class
  I learnt about the scanner class. How to initiate it and how to get the user’s input data.

- Object creation
  By creating Bmi class in the tester class I learned how to create objects, how to use methods and variables from the created instance.

- Static and non-static modifier
  While trying to access methods and variables inside different methods and classes I learnt that what should and should not be modified as static and non-static.

- Parameter passing
  I learnt that the methods that need a parameter can never be used without passing a parameter upon calling. If not I will have to overload the methods as needed.

- Data types
  I had to use different data types throughout the program. I also had to convert integres and doubles values in to String data type. I learned that some methods can only be used with a certain data type.

- Constructors
  I learnt how to create constructors by creating the constructor in Bmi class. Then when I had to create the object I had to pass data in order to create the object.

- Class diagrams
  When referring to the questions I had to learn about class diagrams to understand the question better

- Minimal coding
  By creating a skeleton like behavior of the final code, I learnt how to do a minimalcode and I also learned some good practices of programming.

  I encapsulated the variables using getters and setters. Then I used getter methods inmy Tester class when needed.

- Collections
  I learnt about different types of collections. I used Arraylists in my program and learnthow to add, get, and delete data from an arraylist.

- Flow statements
  By using if else statements, enhanced for loops and switch cases I learnt about andhow to use different flow statements in related scenarios.

  - Input validations

New input validations were added when adding data by the users. The logics had to be figured out when doing this.

GitHub link

https://github.com/hiranyasena/final-.git