UNIVERSITI TEKNOLOGI MALAYSIA

PENGATURCARAAN BERORENTASIKAN OBJEK (OBJECT ORIENTED PROGRAMMING (SECJ2154)

SEMESTER 2 2023/2024

GROUP PROJECT

Nutrition Tracker

CHAN QING YEE (A22EC0040)

LIM SI NI (A22EC0070)

ONG KAI XUEN (A22EC0100)

SOH FEI ZHEN (A22EC0272)

2/SECRH

SECTION 04

Lecturer:

MADAM LIZAWATI MI YUSUF

21st JUNE 2024

The work in this page has been done by: **Soh Fei Zhen** *Date:* **21/06**

SECTION A: PROJECT DESCRIPTION

Synopsis:

Nowadays. importance of health getting aware of people. However, the rise of fast food, processed snacks, and busy lifestyles has led to a significant increase in diet-related health issues such as obesity, diabetes, and heart disease. Therefore, people struggle to keep track of their nutrition intake for analysis purposes. A nutrition tracker system is the solution to this problem.

Our developed nutrition tracker system provides an interface for users to enter their calorie intake daily and record it in the database. Users can view their records anytime, anywhere. Plus, our system will also calculate the BMI based on the information provided by users to recommend the calories that should be taken daily.

In our developed system, there are 5 modules:

1. User Module:
   - This module consists User, Regular User and Admin class.
   - This module is used to create and delete users.

2. Food Module:
   - This module consists FoodItem, FoodItemEnum and Meal class
   - This module is used for users to create meals with food items manually or use the predefined food items with pre-set calories as well as edit the meals

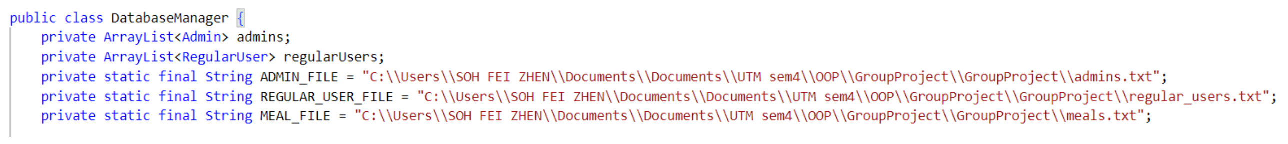
3. Tracking Module:
   - This module consists NutritionTracker class.
   - This module is used to calculate the daily total and recommended calorie intake of users and displays the details of their recorded meals.

4. Utility Module:
   - This module consists DatabaseManager and InvalidFoodException class
   - This module is used to handle data storage and provide exceptions for the food items.

5. Application Module
   - This module consists Main class
   - This module is used to integrate all the modules and consists main function.

The work in this page has been done by: **Soh Fei Zhen** *Date:* **21/06**

Also, we have used 3 .txt files as our database. Please note that to run our system, you should change the value of ADMIN_FILE, REGULAR_USER_FILE and MEAL_FILE instance variables in the DatabaseManager class with your respective path of admin.txt, regular_users.txt and meals.txt on your device.



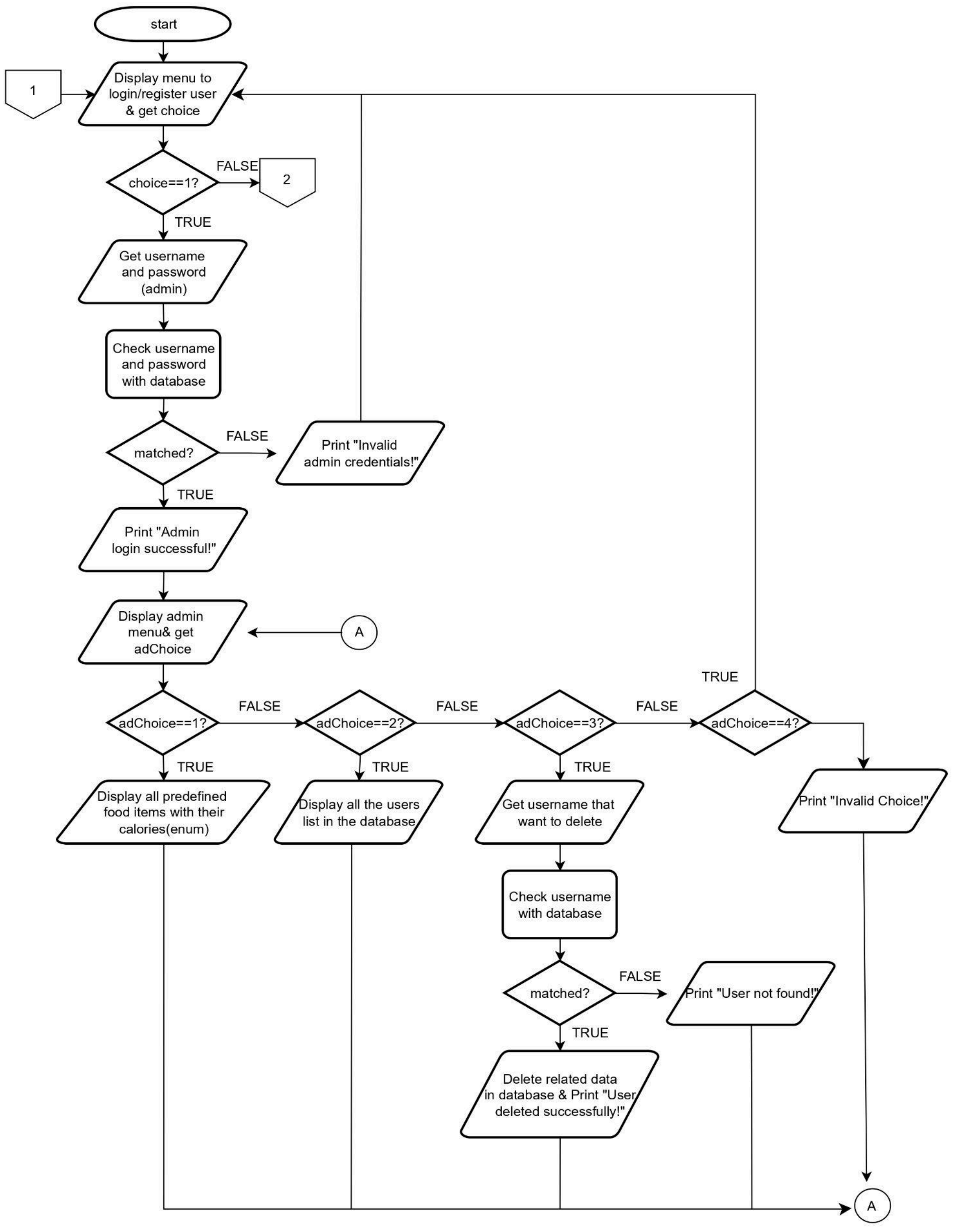
```
public class DatabaseManager {
    private ArrayList<Admin> admins;
    private ArrayList<RegularUser> regularUsers;
    private static final String ADMIN_FILE = "C:\\Users\\SOH FEI ZHEN\\Documents\\Documents\\UTM sem4\\OOP\\GroupProject\\GroupProject\\admins.txt";
    private static final String REGULAR_USER_FILE = "C:\\Users\\SOH FEI ZHEN\\Documents\\Documents\\UTM sem4\\OOP\\GroupProject\\GroupProject\\regular_users.txt";
    private static final String MEAL_FILE = "C:\\Users\\SOH FEI ZHEN\\Documents\\Documents\\UTM sem4\\OOP\\GroupProject\\GroupProject\\meals.txt";
```

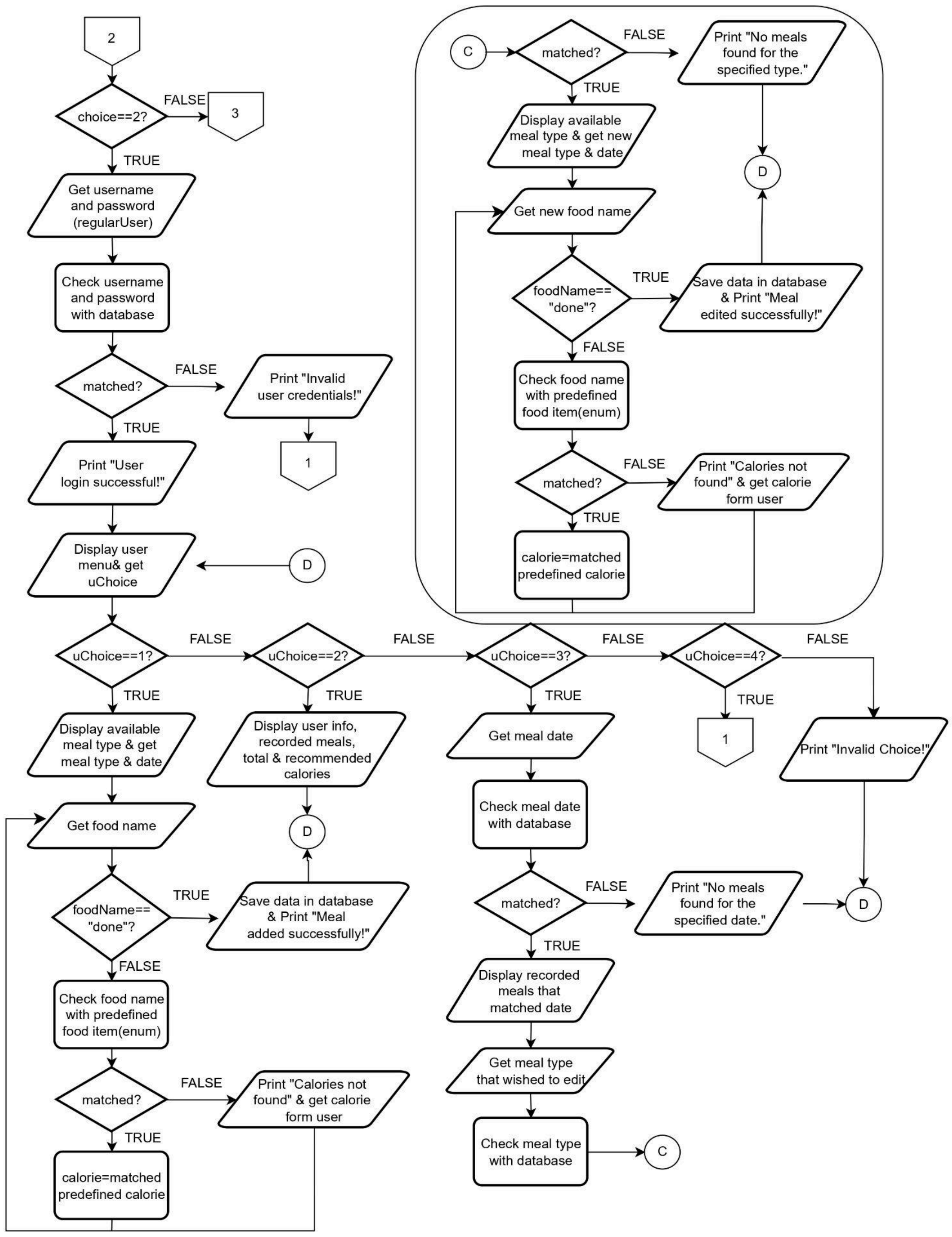
Objective and Scope:

1. To implement encapsulation and data hiding in the system
2. To implement association and composition in the system
3. To implement inheritance, polymorphism and abstract class in the system
4. To implement exception handling in the system

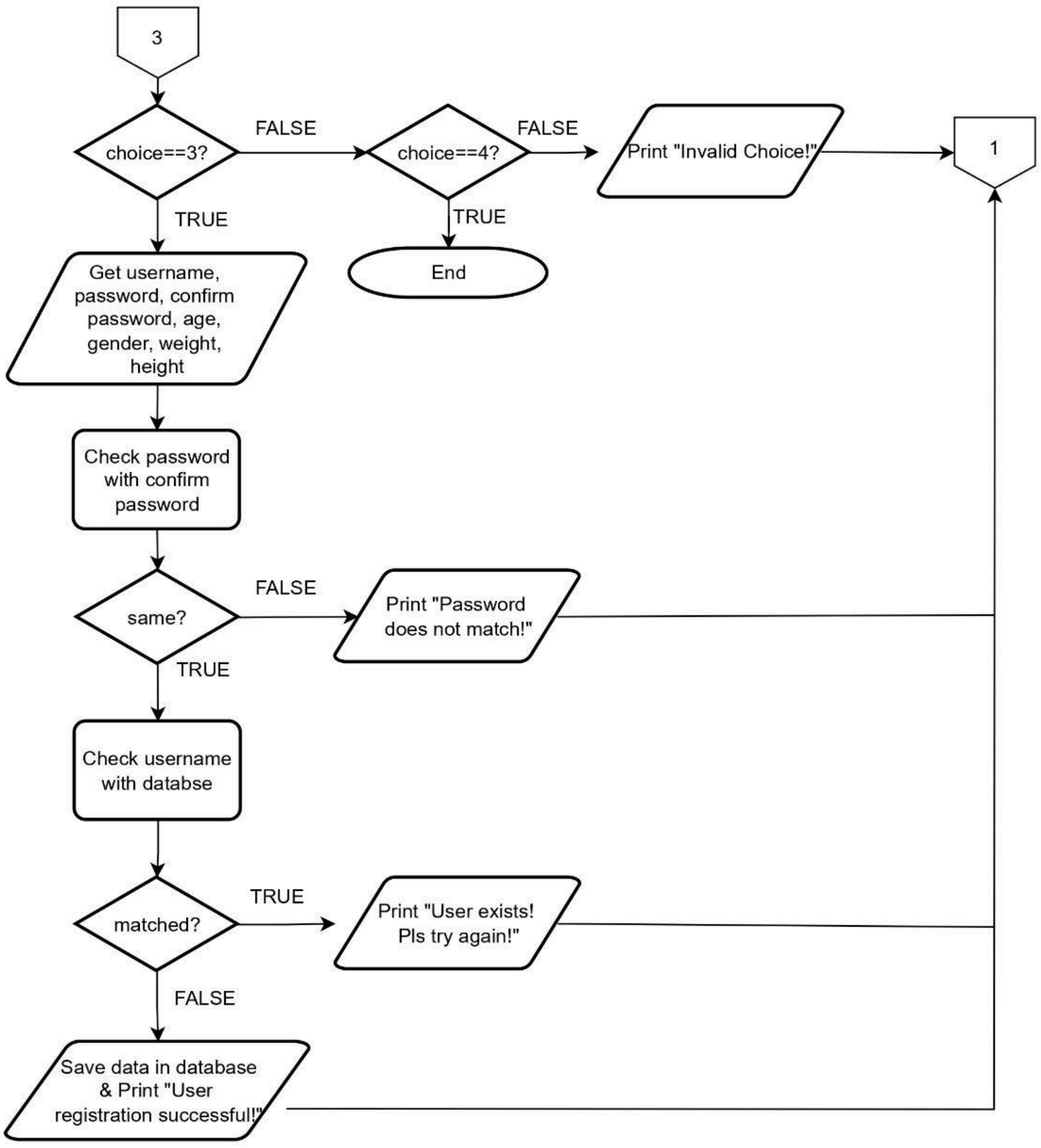
The work in this page has been done by: **Soh Fei Zhen** *Date:* **21/06**

Work flow:


start
1
Display menu to login/register user & get choice
choice==1?
FALSE
2
TRUE
Get username and password (admin)
Check username and password with database
matched?
FALSE
Print "Invalid admin credentials!"
TRUE
Print "Admin login successful!"
Display admin menu& get adChoice
A
adChoice==1?
FALSE
adChoice==2?
FALSE
adChoice==3?
FALSE
adChoice==4?
TRUE
TRUE
Display all predefined food items with their calories(enum)
TRUE
Display all the users list in the database
TRUE
Get username that want to delete
Check username with database
matched?
FALSE
Print "User not found!"
TRUE
Delete related data in database & Print "User deleted successfully!"
Print "Invalid Choice!"
A


2
choice==2?
FALSE
3
TRUE
Get username and password (regularUser)
Check username and password with database
matched?
FALSE
Print "Invalid user credentials!"
1
TRUE
Print "User login successful!"
Display user menu& get uChoice
D
uChoice==1?
FALSE
uChoice==2?
FALSE
uChoice==3?
FALSE
uChoice==4?
FALSE
Print "Invalid Choice!"
D
TRUE
1
TRUE
Display available meal type & get meal type & date
Get food name
foodName== "done"?
TRUE
Save data in database & Print "Meal added successfully!"
D
FALSE
Check food name with predefined food item(enum)
matched?
FALSE
Print "Calories not found" & get calorie form user
TRUE
calorie=matched predefined calorie
TRUE
Display user info, recorded meals, total & recommended calories
D
TRUE
Get meal date
Check meal date with database
matched?
FALSE
Print "No meals found for the specified date."
D
TRUE
Display recorded meals that matched date
Get meal type that wished to edit
Check meal type with database
C
C
matched?
FALSE
Print "No meals found for the specified type."
D
TRUE
Display available meal type & get new meal type & date
Get new food name
foodName== "done"?
TRUE
Save data in database & Print "Meal edited successfully!"
FALSE
Check food name with predefined food item(enum)
matched?
FALSE
Print "Calories not found" & get calorie form user
TRUE
calorie=matched predefined calorie

The work in this page has been done by: **Soh Fei Zhen** *Date:* **21/06**



3
choice==3?
FALSE
choice==4?
FALSE
Print "Invalid Choice!"
1
TRUE
TRUE
End
Get username, password, confirm password, age, gender, weight, height
Check password with confirm password
same?
FALSE
Print "Password does not match!"
TRUE
Check username with databse
matched?
TRUE
Print "User exists! Pls try again!"
FALSE
Save data in database & Print "User registration successful!"

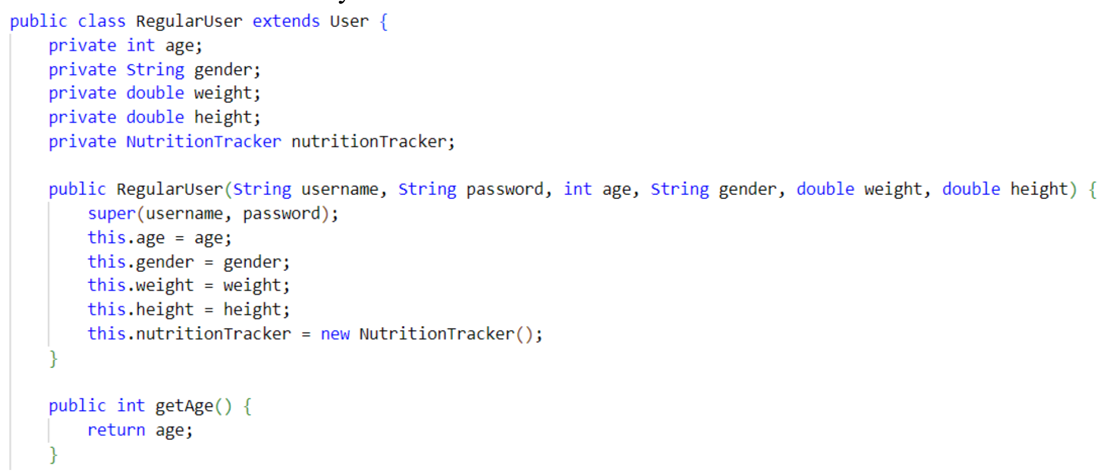
The work in this page has been done by: **Chan Qing Yee** *Date:* **21/06**

OO Concepts:

1. **Encapsulation and data hiding**

   In our developed system, we have used Java language which implements object-oriented programming. That means each class we declare combines data and methods via encapsulation. If the other objects want to access or manipulate the data, they need to access or manipulate it via public methods. This indirect access is known as a programming interface. By encapsulating object data, we maximize reusability, reduce data dependency, and minimize debugging time.

   Also, by implementing the encapsulation concept, we have implemented data hiding too. This is because every object has its own data and method, only objects themselves can access the private data unless by public method. Data hiding is important as it protects attributes from accidental corruption by outside objects, allows the maintainer of the object to modify the internal functioning of the object without "breaking" other's code as well as better security.



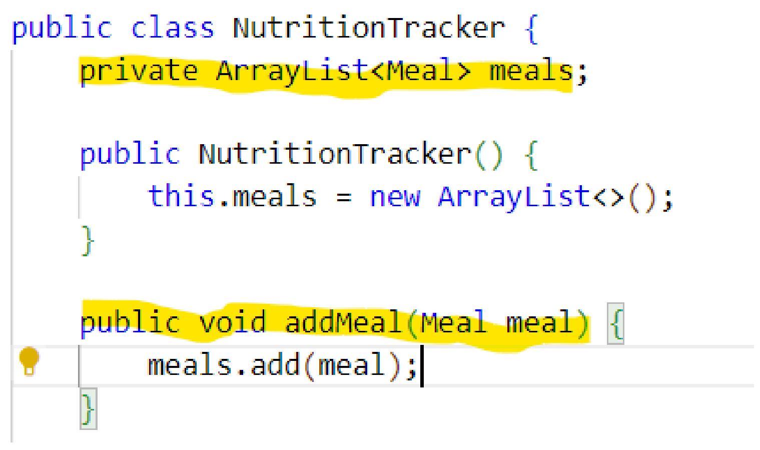
```java
public class RegularUser extends User {
    private int age;
    private String gender;
    private double weight;
    private double height;
    private NutritionTracker nutritionTracker;

    public RegularUser(String username, String password, int age, String gender, double weight, double height) {
        super(username, password);
        this.age = age;
        this.gender = gender;
        this.weight = weight;
        this.height = height;
        this.nutritionTracker = new NutritionTracker();
    }

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

   The above figure shows the RegularUser class has implemented encapsulation by providing accessors such as getAge() for other objects to indirectly access data.

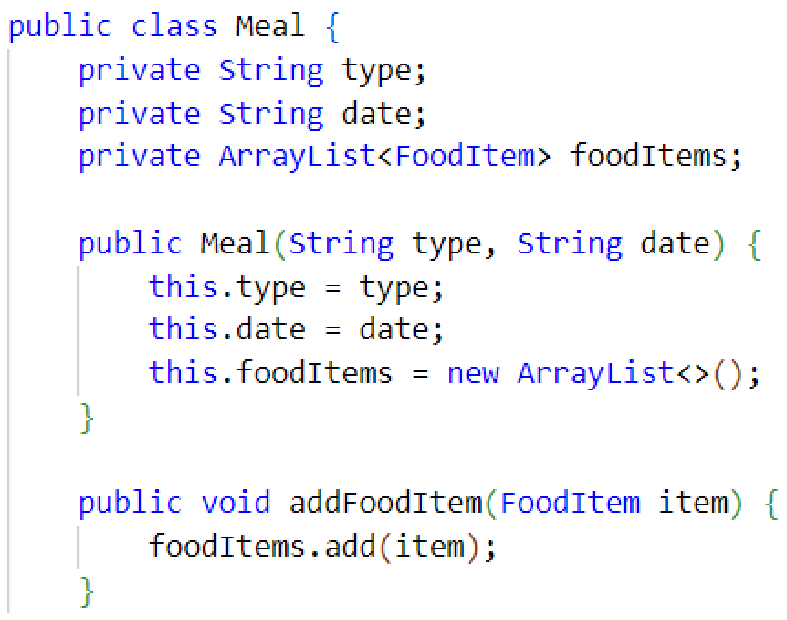
The work in this page has been done by: **Chan Qing Yee** *Date:* **21/06**

2. Association

Association is a general binary relationship between two classes. This relationship allows objects to call methods in other objects. In our system, we only implement the unidirectional association.



```
public class NutritionTracker {
    private ArrayList<Meal> meals;

    public NutritionTracker() {
        this.meals = new ArrayList<>();
    }

    public void addMeal(Meal meal) {
        meals.add(meal);
    }
```



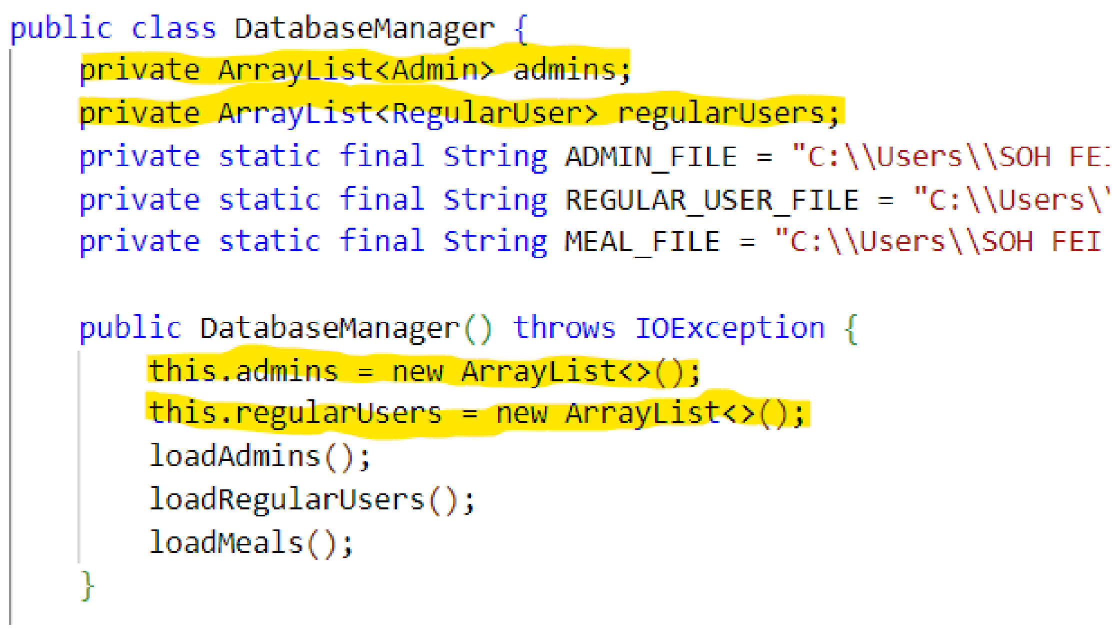
```
public class Meal {
    private String type;
    private String date;
    private ArrayList<FoodItem> foodItems;

    public Meal(String type, String date) {
        this.type = type;
        this.date = date;
        this.foodItems = new ArrayList<>();
    }

    public void addFoodItem(FoodItem item) {
        foodItems.add(item);
    }
```

The above figures show the unidirectional association between the Meal class and NutritionTracker class. This can be proven by the instance variables of Meal datatype objects as well as methods to add Meal objects in NutritionTracker. However, in the Meal class, there is none of the instance variables or methods about the NutritionTracker class. Also, the NutritionTracker class is an enclosing class whereas the Meal class is an enclosed class. The enclosing class exclusively owns enclosed classes. That means the existence of objects are independent where when a NutritionTracker object is destroyed, the Meal objects owned by this NutritionTracker object will not be destroyed together.

The work in this page has been done by: **Chan Qing Yee** *Date:* **21/06**

3. Composition

Composition is a special form of association, which represents an ownership relationship between two classes.



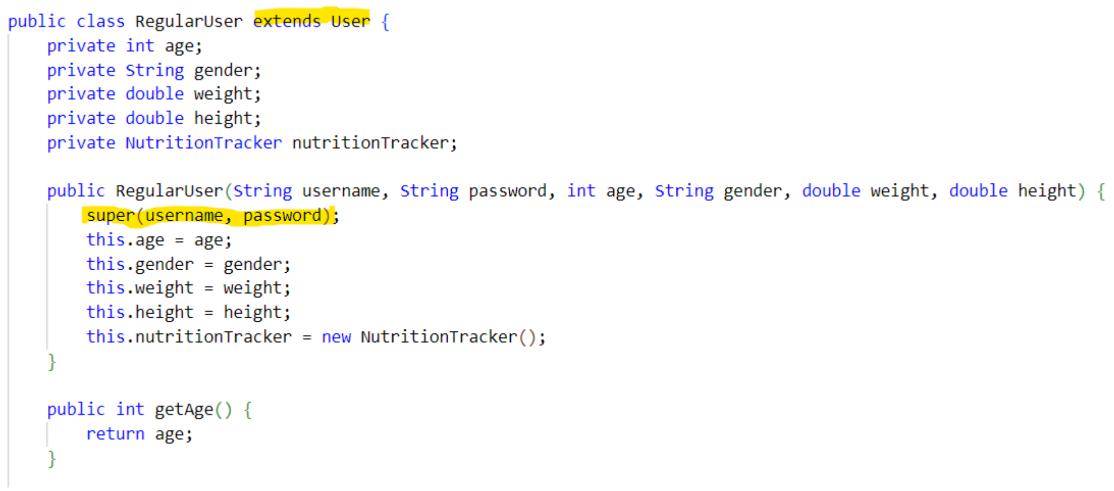
```
public class DatabaseManager {
    private ArrayList<Admin> admins;
    private ArrayList<RegularUser> regularUsers;
    private static final String ADMIN_FILE = "C:\\Users\\SOH FE:
    private static final String REGULAR_USER_FILE = "C:\\Users\'
    private static final String MEAL_FILE = "C:\\Users\\SOH FEI

    public DatabaseManager() throws IOException {
        this.admins = new ArrayList<>();
        this.regularUsers = new ArrayList<>();
        loadAdmins();
        loadRegularUsers();
        loadMeals();
    }
```

The above figure shows the composition between the DatabaseManager class with Admin class and the RegularUser class. In this case, the DatabaseManager class is an enclosing class whereas the Admin class and RegularUser class are enclosed classes. The enclosed classes are the parts of the enclosed class. That means the existence of objects is dependent where when a DatabaseManager object is destroyed, the Admin objects and RegularUsers objects owned by this DatabaseManager object will be destroyed too.

The work in this page has been done by: **Chan Qing Yee** *Date:* **21/06**

4. Inheritance

Inheritance provides a way to create a new class from an existing class. That means classes can inherit attributes and methods from other classes, and add extra attributes and methods of their own.



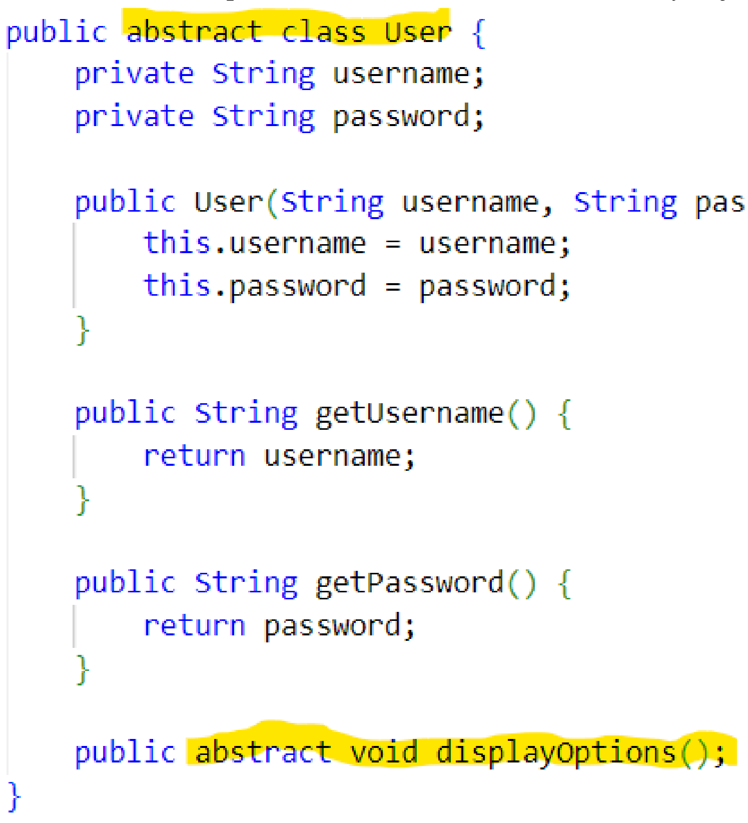
```java
public class RegularUser extends User {
    private int age;
    private String gender;
    private double weight;
    private double height;
    private NutritionTracker nutritionTracker;

    public RegularUser(String username, String password, int age, String gender, double weight, double height) {
        super(username, password);
        this.age = age;
        this.gender = gender;
        this.weight = weight;
        this.height = height;
        this.nutritionTracker = new NutritionTracker();
    }

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

The above figure shows the inheritance relationship between RegularUser and User where RegularUser is a child class, User is a parent class. A child class will own all the attributes and methods declared in the parent class and add on their specific attribute and methods. The username and password attributes shown above are the attributes inherited from the parent class, User. To initialize them, we need to invoke the parent class constructor by super() method. The instance variables and methods are declared in the RegularUser class making it a specialized form of the User class.

The work in this page has been done by: **Chan Qing Yee** *Date:* **21/06**

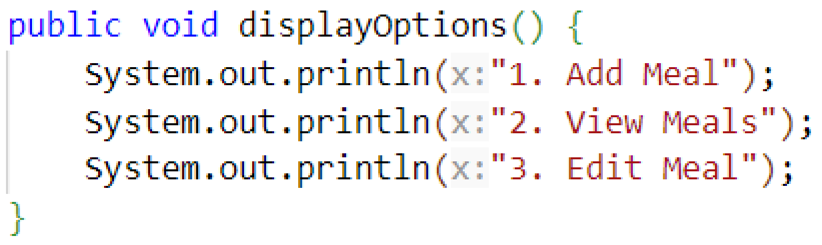
5. Abstract class and polymorphism

An abstract class is a class with one or more abstract methods where the methods should not have implementation. We also cannot have any object of abstract class.



```
public abstract class User {
    private String username;
    private String password;

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public abstract void displayOptions();
}
```

The above figure shows the abstract class in our system. As you see, there is one abstract method in this class, so this class will automatically become abstract and must be extended by the child class. Yet, the abstract method must be overridden by the subclass so that the subclass will not transform to an abstract class too.

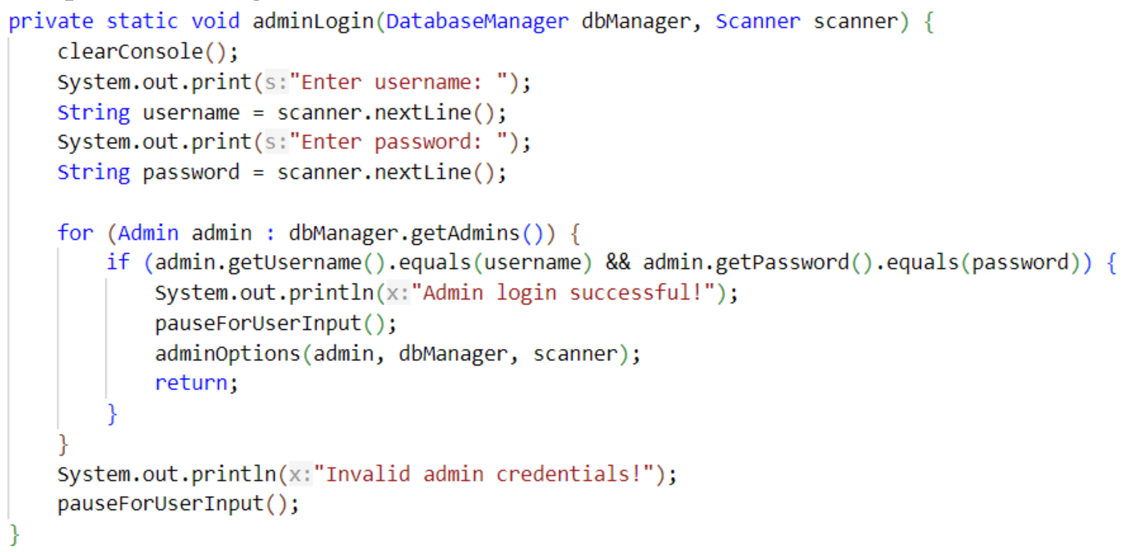


```
public void displayOptions() {
    System.out.println(x:"1. Add Meal");
    System.out.println(x:"2. View Meals");
    System.out.println(x:"3. Edit Meal");
}
```

The figure above shows the overridden method in the RegularUser class where it is the child class of the User class. To override the method, we should have exactly the same function name, parameter lists and return datatype. If a User datatype variable reference to a RegularUser, the polymorphism is implemented as the implementation of displayOptions() will be used either in User class or RegularUser class will be determined during run time.

The work in this page has been done by: **Chan Qing Yee** *Date:* **21/06**

6. **Exception handling**



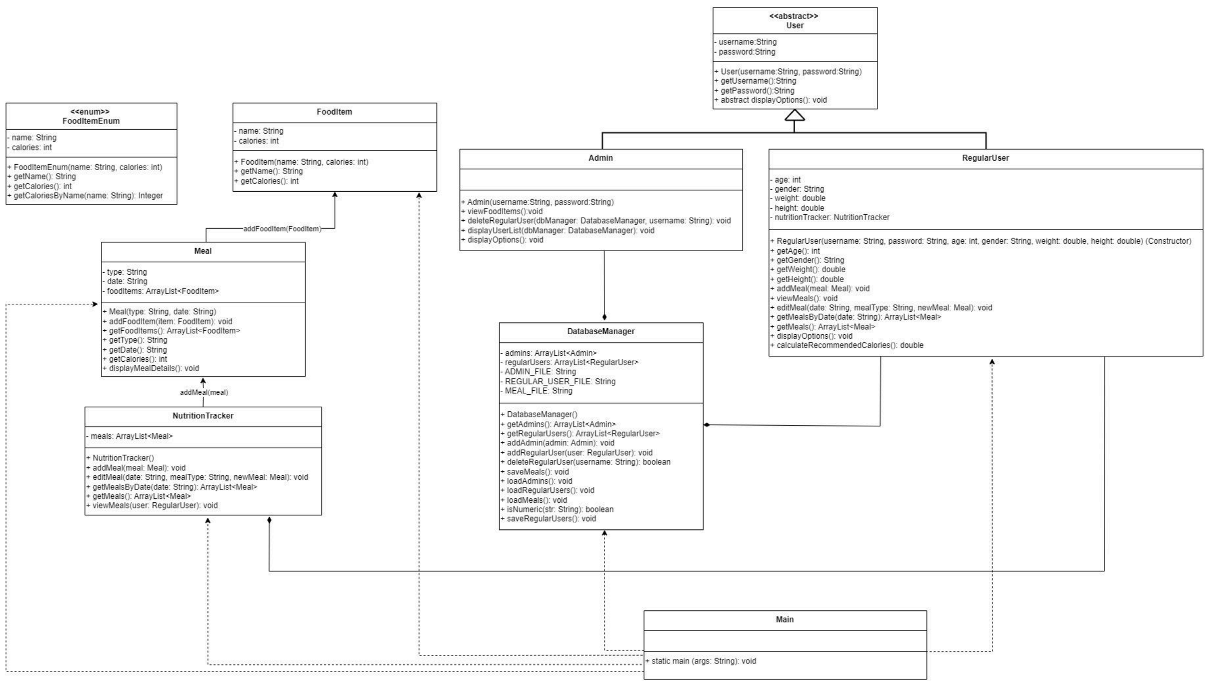
```
private static void adminLogin(DatabaseManager dbManager, Scanner scanner) {
    clearConsole();
    System.out.print(s:"Enter username: ");
    String username = scanner.nextLine();
    System.out.print(s:"Enter password: ");
    String password = scanner.nextLine();

    for (Admin admin : dbManager.getAdmins()) {
        if (admin.getUsername().equals(username) && admin.getPassword().equals(password)) {
            System.out.println(x:"Admin login successful!");
            pauseForUserInput();
            adminOptions(admin, dbManager, scanner);
            return;
        }
    }
    System.out.println(x:"Invalid admin credentials!");
    pauseForUserInput();
}
```

The above figure shows one of the exception handling we had in our developed system. The exception we try to handle is the user entering an invalid input of username and password. If the user entered the username and password which is not matched with the database, an error message “Invalid admin credentials” will be displayed to the user to indicates their invalid inputs.

The work in this page has been done by: **LIM SI NI** *Date:* **21/06**

SECTION B: CLASS DIAGRAMS

The following diagram shows class diagram for the "Nutrition Tracker".



<<abstract>>
User
- username:String
- password:String
+ User(username:String, password:String)
+ getUsername():String
+ getPassword():String
+ abstract displayOptions(): void
<<enum>>
FoodItemEnum
- name: String
- calories: int
+ FoodItemEnum(name: String, calories: int)
+ getName(): String
+ getCalories(): int
+ getCaloriesByName(name: String): Integer
FoodItem
- name: String
- calories: int
+ FoodItem(name: String, calories: int)
+ getName(): String
+ getCalories(): int
Admin
+ Admin(username:String, password:String)
+ viewFoodItems():void
+ deleteRegularUser(dbManager: DatabaseManager, username: String): void
+ displayUserList(dbManager: DatabaseManager): void
+ displayOptions(): void
RegularUser
- age: int
- gender: String
- weight: double
- height: double
- nutritionTracker: NutritionTracker
+ RegularUser(username: String, password: String, age: int, gender: String, weight: double, height: double) (Constructor)
+ getAge(): int
+ getGender(): String
+ getWeight(): double
+ getHeight(): double
+ addMeal(meal: Meal): void
+ viewMeals(): void
+ editMeal(date: String, mealType: String, newMeal: Meal): void
+ getMealsByDate(date: String): ArrayList<Meal>
+ getMeals(): ArrayList<Meal>
+ displayOptions(): void
+ calculateRecommendedCalories(): double
addFoodItem(FoodItem)
Meal
- type: String
- date: String
- foodItems: ArrayList<FoodItem>
+ Meal(type: String, date: String)
+ addFoodItem(item: FoodItem): void
+ getFoodItems(): ArrayList<FoodItem>
+ getType(): String
+ getDate(): String
+ getCalories(): int
+ displayMealDetails(): void
DatabaseManager
- admins: ArrayList<Admin>
- regularUsers: ArrayList<RegularUser>
- ADMIN_FILE: String
- REGULAR_USER_FILE: String
- MEAL_FILE: String
+ DatabaseManager()
+ getAdmins(): ArrayList<Admin>
+ getRegularUsers(): ArrayList<RegularUser>
+ addAdmin(admin: Admin): void
+ addRegularUser(user: RegularUser): void
+ deleteRegularUser(username: String): boolean
+ saveMeals(): void
+ loadAdmins(): void
+ loadRegularUsers(): void
+ loadMeals(): void
+ isNumeric(str: String): boolean
+ saveRegularUsers(): void
addMeal(meal)
NutritionTracker
- meals: ArrayList<Meal>
+ NutritionTracker()
+ addMeal(meal: Meal): void
+ editMeal(date: String, mealType: String, newMeal: Meal): void
+ getMealsByDate(date: String): ArrayList<Meal>
+ getMeals(): ArrayList<Meal>
+ viewMeals(user: RegularUser): void
Main
+ static main (args: String): void

PDF File:
https://drive.google.com/file/d/1p6gn3fgb4KgbxAUSTbbQZom97mcHeHEQ/view?usp=sharing

The work in this page has been done by: **LIM SI NI** *Date:* **21/06**

Class User:

Attributes	Description
username	The username to login
password	The password to login
Methods	**Description**
getUsername	To get the username value
getPassword	To get the password value
displayOptions	A abstract function

Class Admin:

Attributes	Description
Methods	**Description**
viewFoodItems	To display enum food item list
deleteRegularUser	To delete specific regular user
displayUserList	To display regular user list
displayOptions	To display the options menu for admin

Class RegularUser

Attributes	Description
age	The age of user
gender	The gender of user
weight	The weight of user
height	The height of user
nutritionTracker	The nutrition tracker for the user
Methods	**Description**
getAge	To get the age of user
getGender	To get the gender of user
getWeight	To get the weight of user
getHeight	To get the height of user
calculateBMI	Calculate the BMI by using formulate weight / ((height / 100) * (height / 100))
addMeal	To add new meal to the nutrition tracker of user
viewMeals	To view all the meal records in the nutrition tracker
editMeal	To edit the specific meal
getMealByDate	To search and display the meal by key in the date
getMeals	To return the object meal array list
displayOptions	To display the options menu for user
calculateRecommendedCalories	To calculate the calories that recommended to each user based on their age, gender, weight and height

The work in this page has been done by: **LIM SI NI** *Date:* **21/06**

Class NutritionTracker:

Attributes	Description
meals	Array that store meal objects
Methods	**Description**
addMeal	To add a new meal object into the meals array
editMeal	To edit a specific meal object in the meals array
getMealsByDate	To return the meals array by searching the date
getMeals	To return a specific meal object from the meals array list
viewMeals	To display the meals of list of the user

Class FoodItemEnum:

Attributes	Description
name	The name of food
calories	The calorie of food
Methods	**Description**
getName	To get the name of food
getCalories	To get the calorie of food
getCaloriesByName	To get the calorie value of the food by entering the food name

Class FoodItem:

Attributes	Description
name	The name of food
calories	The calorie of food
Methods	**Description**
getName	To get the name of food
getCalories	To get the calorie of food

The work in this page has been done by: **LIM SI NI** *Date:* **21/06**

Class Meal:

Attributes	**Description**
type	The type of meals: Breakfast, lunch, dinner or snack
date	The date of the meal
foodItems	Thea array list of foodItem object
Methods	**Description**
addFoodItem	To add new food item object into foodItems array
getFoodItem	To get the foodItem array
getType	To get the type of meal
getDate	To get the date of meal
getCalories	To get the total calorie of food item in the array
displayMealDetails	To display the meal details and all the food in the list

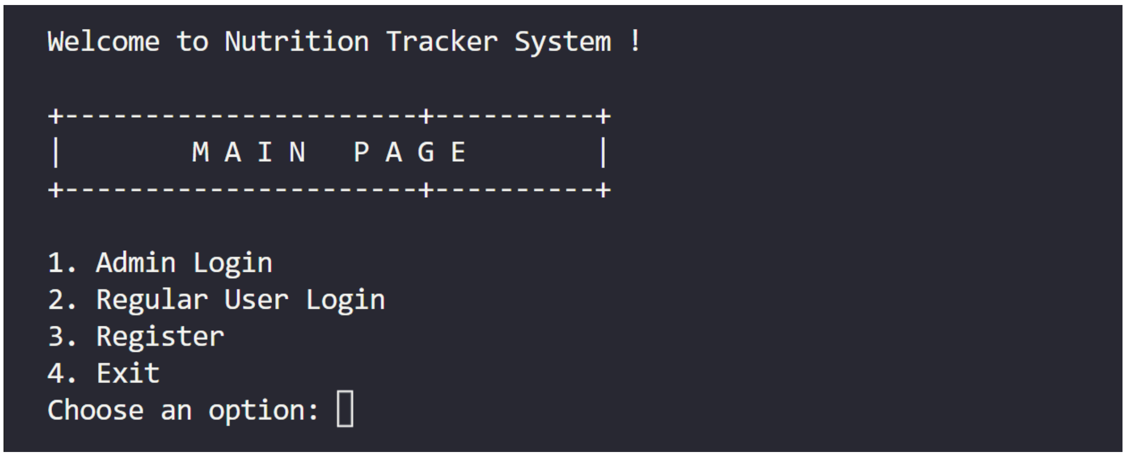
Class DatabaseManager

Attributes	**Description**
admins	The array list of admins
regularUsers	The arraylist of regular users
ADMIN_FILE	The constant admins list file name
REGULAR_USER_FILE	The constant regular users list file name
MEAL_FILE	The constant meal list file name
Methods	**Description**
getAdmins	To get the array of admins
getRegularUsers	To get the array of regular users
addAdmin	To add new admin into the array
addRegularUser	To add new regular user into the array
deleteRegularUser	To get the result of deleting a specific regular user
saveMeals	To load and save the nutrition tracker data of users into the MEAL_FILE
loadAdmins	To load the data of admins from the ADMIN_FILE and store them in the array
loadRegularUsers	To load the data of regular users from the REGULAR_USER_FILE and store them in the array
loadMeals	To load the data of meals from the MEAL_FILE and store them in the array
isNumeric	To cast a string into an integer and return the result of casting either successful or not
saveRegularUsers	To load and save the data of regular user into the REGULAR_USER_FILE

The work in this page has been done by: **ONG KAI XUEN** *Date:* **21/06**

SECTION C: USER MANUAL

At user's first sight is the main page, the system will prompt user to choose the type of login, whether is Admin Login, Regular User Login, Register, or exit.

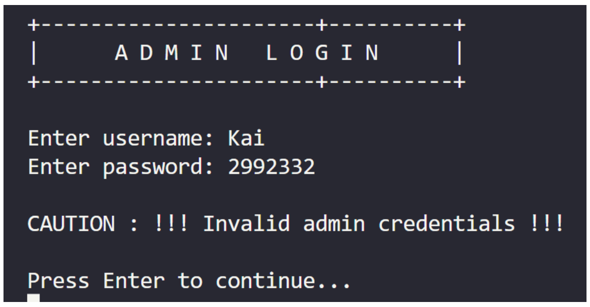


```
Welcome to Nutrition Tracker System !

+---------------------+----------+
|       M A I N   P A G E        |
+---------------------+----------+

1. Admin Login
2. Regular User Login
3. Register
4. Exit
Choose an option: 
```

1. Admin Login

If user choose option 1- admin login, the user required to enter username and password to verify the admin.

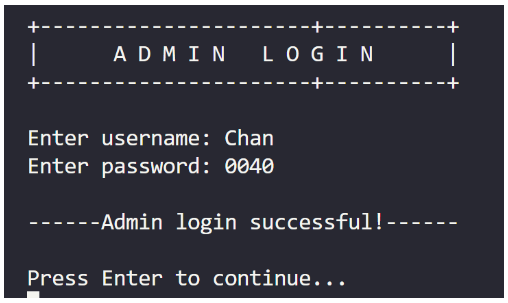
a) Below is the example of entering wrong username and password
User can press enter to return to the main page



```
+---------------------+----------+
|     A D M I N   L O G I N      |
+---------------------+----------+

Enter username: Kai
Enter password: 2992332

CAUTION : !!! Invalid admin credentials !!!

Press Enter to continue...
```

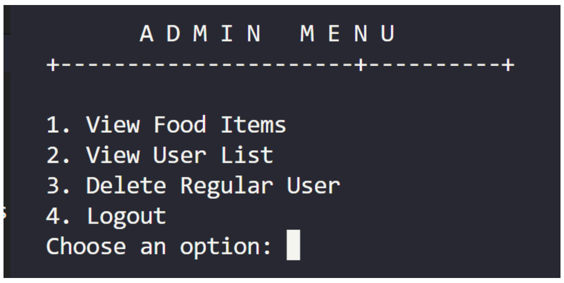
The work in this page has been done by: **ONG KAI XUEN** *Date:* **21/06**

b) Below is the example of admin successfully login.
User can press enter to enter next page



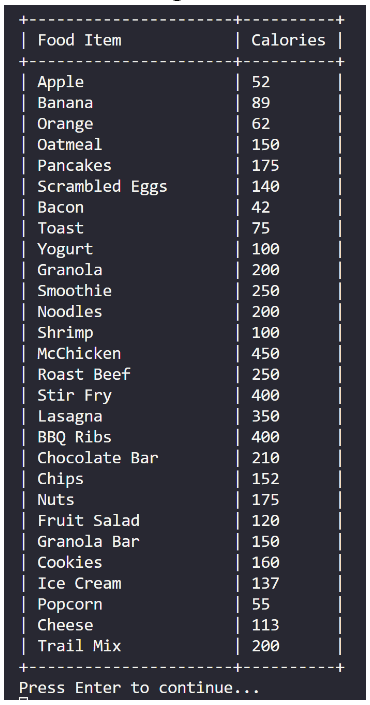
```
+---------------------+----------+
|     A D M I N   L O G I N     |
+---------------------+----------+

Enter username: Chan
Enter password: 0040

------Admin login successful!------

Press Enter to continue...
```

After successfully login, system display the admin menu. The admin can choose to view food item, view user list, delete regular user, and logout.



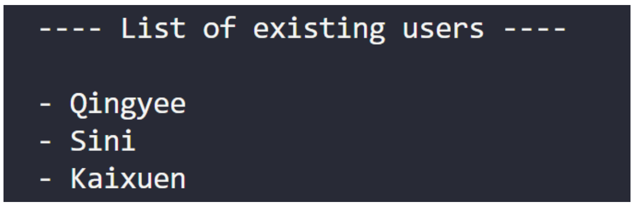
```
        A D M I N   M E N U
  +---------------------+----------+

  1. View Food Items
  2. View User List
  3. Delete Regular User
  4. Logout
  Choose an option:
```

i) if admin choose option 1, system will display the list of food items and their calories
Admin can press enter to return to the admin menu.



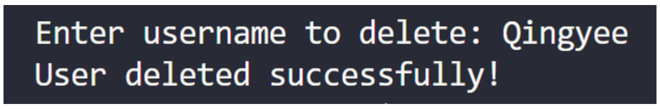
```
+--------------------+---------+
| Food Item          | Calories |
+--------------------+---------+
| Apple              | 52       |
| Banana             | 89       |
| Orange             | 62       |
| Oatmeal            | 150      |
| Pancakes           | 175      |
| Scrambled Eggs     | 140      |
| Bacon              | 42       |
| Toast              | 75       |
| Yogurt             | 100      |
| Granola            | 200      |
| Smoothie           | 250      |
| Noodles            | 200      |
| Shrimp             | 100      |
| McChicken          | 450      |
| Roast Beef         | 250      |
| Stir Fry           | 400      |
| Lasagna            | 350      |
| BBQ Ribs           | 400      |
| Chocolate Bar      | 210      |
| Chips              | 152      |
| Nuts               | 175      |
| Fruit Salad        | 120      |
| Granola Bar        | 150      |
| Cookies            | 160      |
| Ice Cream          | 137      |
| Popcorn            | 55       |
| Cheese             | 113      |
| Trail Mix          | 200      |
+--------------------+---------+
Press Enter to continue...
```

ii) if admin choose option 2, list of system's existing users is being displayed
Admin can press enter to return to the admin menu.



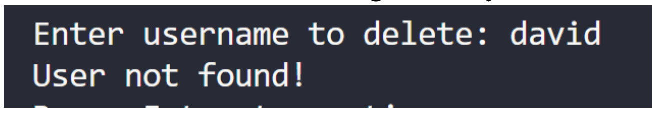
```
---- List of existing users ----

- Qingyee
- Sini
- Kaixuen
```

iii) if admin choose option 3, admin is required to enter the username of user to delete from the system
Admin can press enter to return to the admin menu.



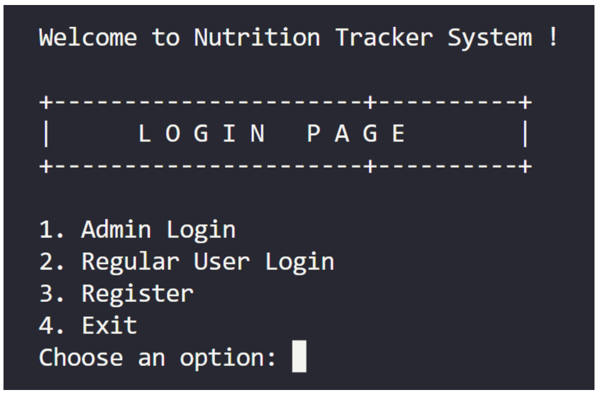
```
Enter username to delete: Qingyee
User deleted successfully!
```

if the username is not existing to the system, it will unable to delete it.



```
Enter username to delete: david
User not found!
```

iv) System will return to the main page if admin choose to logout.



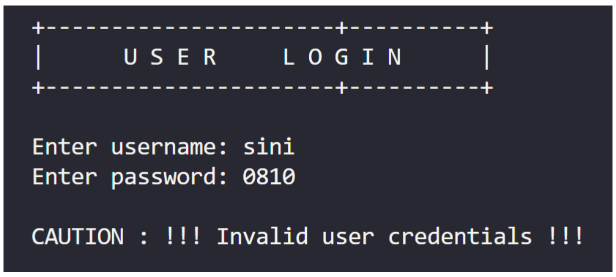
```
Welcome to Nutrition Tracker System !

+---------------------+----------+
|     L O G I N   P A G E        |
+---------------------+----------+

1. Admin Login
2. Regular User Login
3. Register
4. Exit
Choose an option: 
```

The work in this page has been done by: **ONG KAI XUEN** *Date:* **21/06**

2. Regular User Login

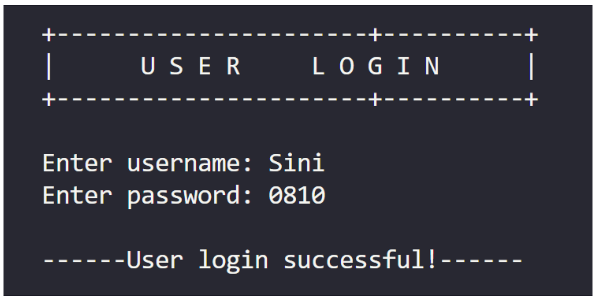
If user choose option 2- regular user login, the user required to enter username and password to verify him/her.

a) Below is the example of entering wrong username and password
User can press enter to return to the main page



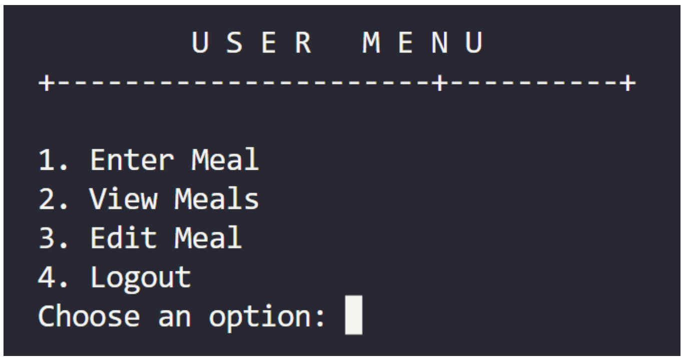
```
+---------------------+----------+
|     U S E R    L O G I N     |
+---------------------+----------+

Enter username: sini
Enter password: 0810

CAUTION : !!! Invalid user credentials !!!
```

b) Below is the example of user successfully login.
User can press enter to proceed enter the next page



```
+---------------------+----------+
|     U S E R    L O G I N     |
+---------------------+----------+

Enter username: Sini
Enter password: 0810

------User login successful!------
```

After successfully login, system display the user menu. The user can choose to enter meal, view meals, edit meal, and logout.



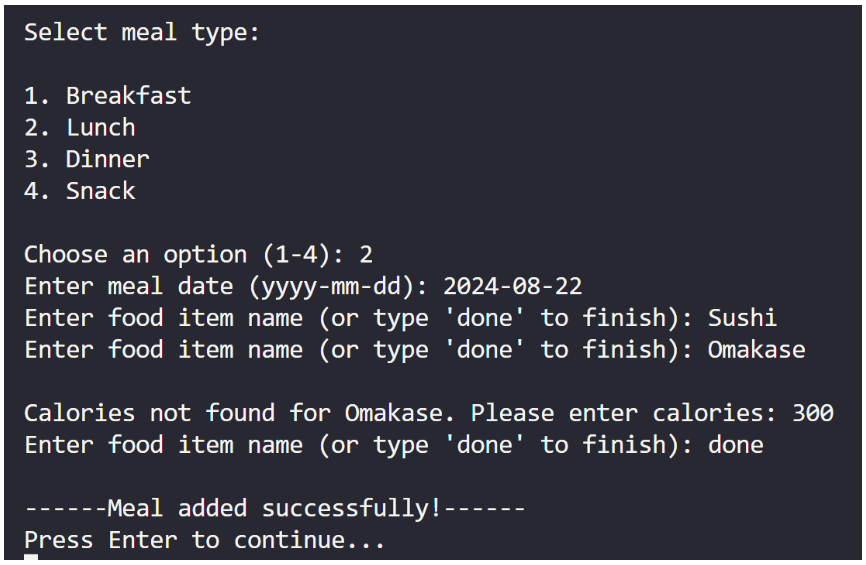
```
         U S E R   M E N U
+---------------------+----------+

1. Enter Meal
2. View Meals
3. Edit Meal
4. Logout
Choose an option: 
```

The work in this page has been done by: **ONG KAI XUEN** *Date:* **21/06**

i) For option 1, user can choose to record their meal.

User can record their meal by entering the meal type, date and also the food item.

If the calories of food item that user entered is not recorded in the system, such as the "Omakase" , user will required to enter the calories of that speific food item.

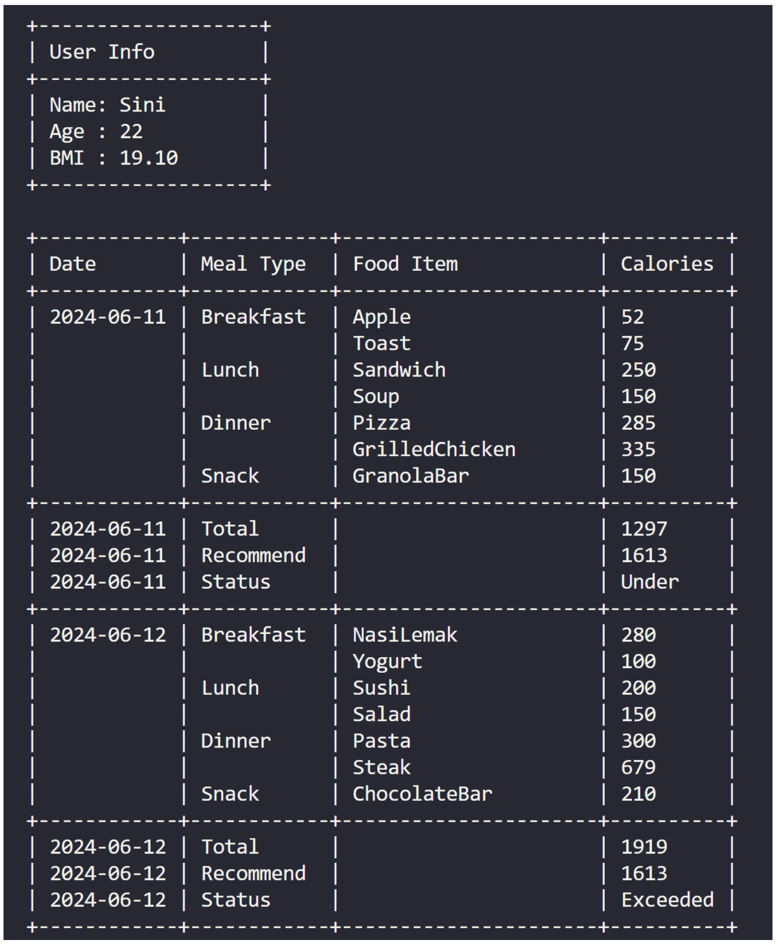


```
Select meal type:

1. Breakfast
2. Lunch
3. Dinner
4. Snack

Choose an option (1-4): 2
Enter meal date (yyyy-mm-dd): 2024-08-22
Enter food item name (or type 'done' to finish): Sushi
Enter food item name (or type 'done' to finish): Omakase

Calories not found for Omakase. Please enter calories: 300
Enter food item name (or type 'done' to finish): done

------Meal added successfully!------
Press Enter to continue...
```

The work in this page has been done by: **ONG KAI XUEN** *Date:* **21/06**

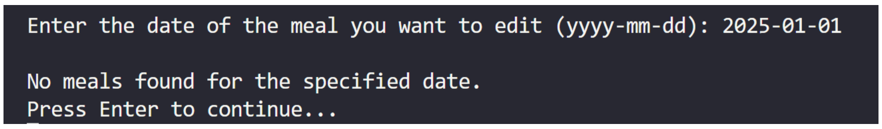
ii) if user chooses option 2, the system will display the user's personal details and the list of meals they have recorded. System will also automatically calculate and display their calories consumed status.

User can press enter to return to the user menu.



```
+-------------------+
| User Info         |
+-------------------+
| Name: Sini        |
| Age : 22          |
| BMI : 19.10       |
+-------------------+

+------------+------------+----------------------+----------+
| Date       | Meal Type  | Food Item            | Calories |
+------------+------------+----------------------+----------+
| 2024-06-11 | Breakfast  | Apple                | 52       |
|            |            | Toast                | 75       |
|            | Lunch      | Sandwich             | 250      |
|            |            | Soup                 | 150      |
|            | Dinner     | Pizza                | 285      |
|            |            | GrilledChicken       | 335      |
|            | Snack      | GranolaBar           | 150      |
+------------+------------+----------------------+----------+
| 2024-06-11 | Total      |                      | 1297     |
| 2024-06-11 | Recommend  |                      | 1613     |
| 2024-06-11 | Status     |                      | Under    |
+------------+------------+----------------------+----------+
| 2024-06-12 | Breakfast  | NasiLemak            | 280      |
|            |            | Yogurt               | 100      |
|            | Lunch      | Sushi                | 200      |
|            |            | Salad                | 150      |
|            | Dinner     | Pasta                | 300      |
|            |            | Steak                | 679      |
|            | Snack      | ChocolateBar         | 210      |
+------------+------------+----------------------+----------+
| 2024-06-12 | Total      |                      | 1919     |
| 2024-06-12 | Recommend  |                      | 1613     |
| 2024-06-12 | Status     |                      | Exceeded |
+------------+------------+----------------------+----------+
```

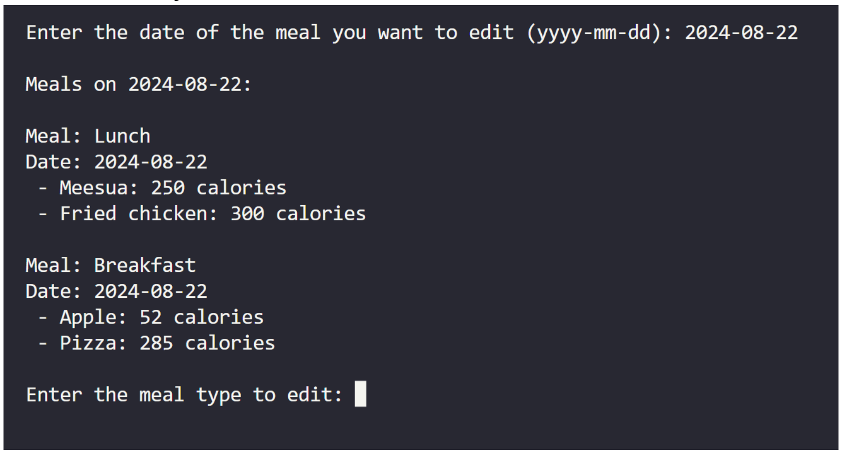
iii) if user choose option 3, user is required to enter the date to edit their meal
If the date entered is not found, system will display error message.



```
Enter the date of the meal you want to edit (yyyy-mm-dd): 2025-01-01

No meals found for the specified date.
Press Enter to continue...
```

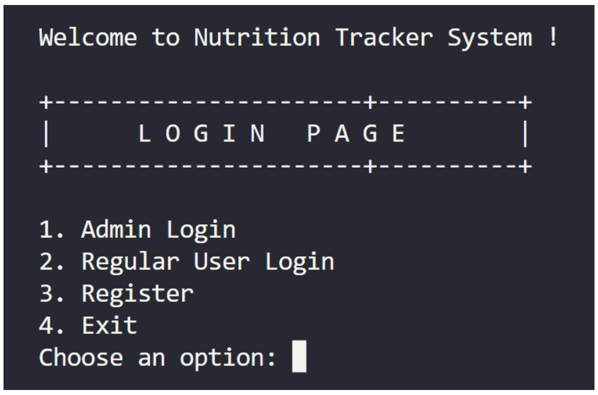
If the date entered is found, system will display the list of meals on that day,
User can modify the meal by entering the information of meal.

For example, if user enter Breakfast, user will required to re-enter the whole breakfast list on that day.



```
Enter the date of the meal you want to edit (yyyy-mm-dd): 2024-08-22

Meals on 2024-08-22:

Meal: Lunch
Date: 2024-08-22
 - Meesua: 250 calories
 - Fried chicken: 300 calories

Meal: Breakfast
Date: 2024-08-22
 - Apple: 52 calories
 - Pizza: 285 calories

Enter the meal type to edit: 
```

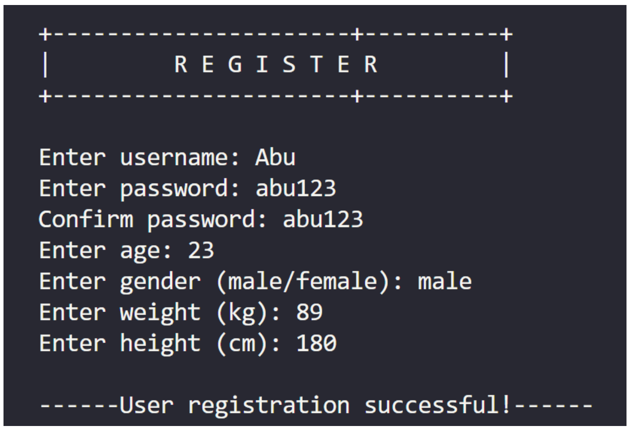
iv) System will return to the main page if user choose to logout.



```
Welcome to Nutrition Tracker System !

+----------------------+----------+
|     L O G I N   P A G E         |
+----------------------+----------+

1. Admin Login
2. Regular User Login
3. Register
4. Exit
Choose an option: 
```

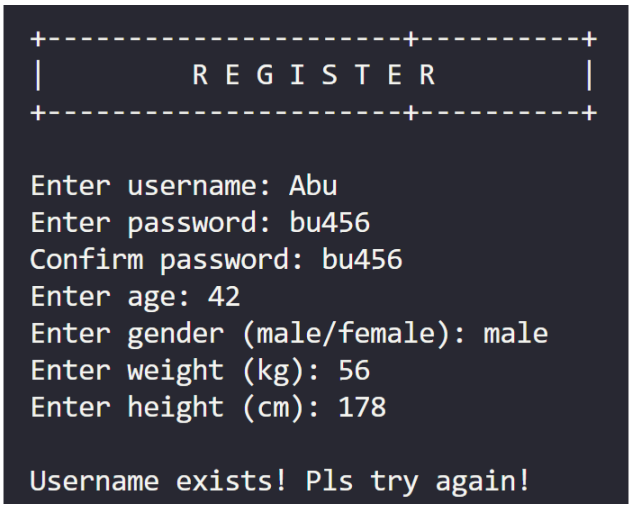
3. Register

If user choose option 3-register , the user is required to enter their information in order to register to the system.



```
+---------------------+----------+
|        R E G I S T E R         |
+---------------------+----------+

Enter username: Abu
Enter password: abu123
Confirm password: abu123
Enter age: 23
Enter gender (male/female): male
Enter weight (kg): 89
Enter height (cm): 180

------User registration successful!------
```

If someone trying to use the same username to register, the registration will not success.
User then press enter to return to main page.



```
+---------------------+----------+
|        R E G I S T E R         |
+---------------------+----------+

Enter username: Abu
Enter password: bu456
Confirm password: bu456
Enter age: 42
Enter gender (male/female): male
Enter weight (kg): 56
Enter height (cm): 178

Username exists! Pls try again!
```