

**ITS63304 Object-Oriented Programming**

· Complete this cover page and attach it to your assignment – this should be your first page.

|  |  |  |
| --- | --- | --- |
| **Student declaration:** | | |
| **I declare that:**   * **I understand what is meant by plagiarism.** * **The implication of plagiarism has been explained to us by our lecturer This project is all our work, and I have acknowledged any use of the published or unpublished works of other /people.** | | |
| **Names of Group Member** | | |
| **No** | **Student ID** | **Student Name** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |

1

**MARKING RUBRIC**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria** | **Excellent (8 – 10)** | **Good (6 – 7)** | **Average**  **(4-5)** | **Poor**  **(0-3)** | **Your Score** |
| **Description and rationale** | A detailed description and outstanding support for the  SDG13 aim | Sufficient explanation and backing SDG13 target | Average description and  average  support SDG13  target | Poor description and poor  support SDG13  target |  |
| **User**  **Interface** | Extremely attractive and user-friendly | Moderate in terms of both aesthetics and ease of use | Average visual appeal and user friendliness | Not appealing or user- friendly |  |
| **Source code** | Extremely rational, organized, and satisfying every criterion | Acceptable in that it is logically sound, wellstructured, and generally satisfying. | The average  logical organization that meets given criteria | Not logical, poor organization  and meet few  criteria |  |
| **Report** | Extensive and thorough coverage | Detailed and well-written. | There is little detail and the material is average. | Not detailed and not complete |  |
| **Lesson learned** | The acquired knowledge is extensive and exhaustive in every respect. | The lesson is good and covers most of what you need to know. | The lesson learned is adequate and covers some ground. | The lesson learned is poor and incomplete |  |
| **Overall** | Comprehensive and complete in  all aspects | Good and comprehensive. | Average and cover some ground | Poor and incomplete |  |
| **Reference** | 10 and more  recent references | 6-7 recent references | 4-5 recent references | Less than 4 recent references |  |

# **TOTAL** /70

NOTE: Total marks will be adjusted to a maximum of **50%** allocated for this assignment.

COMMENT:





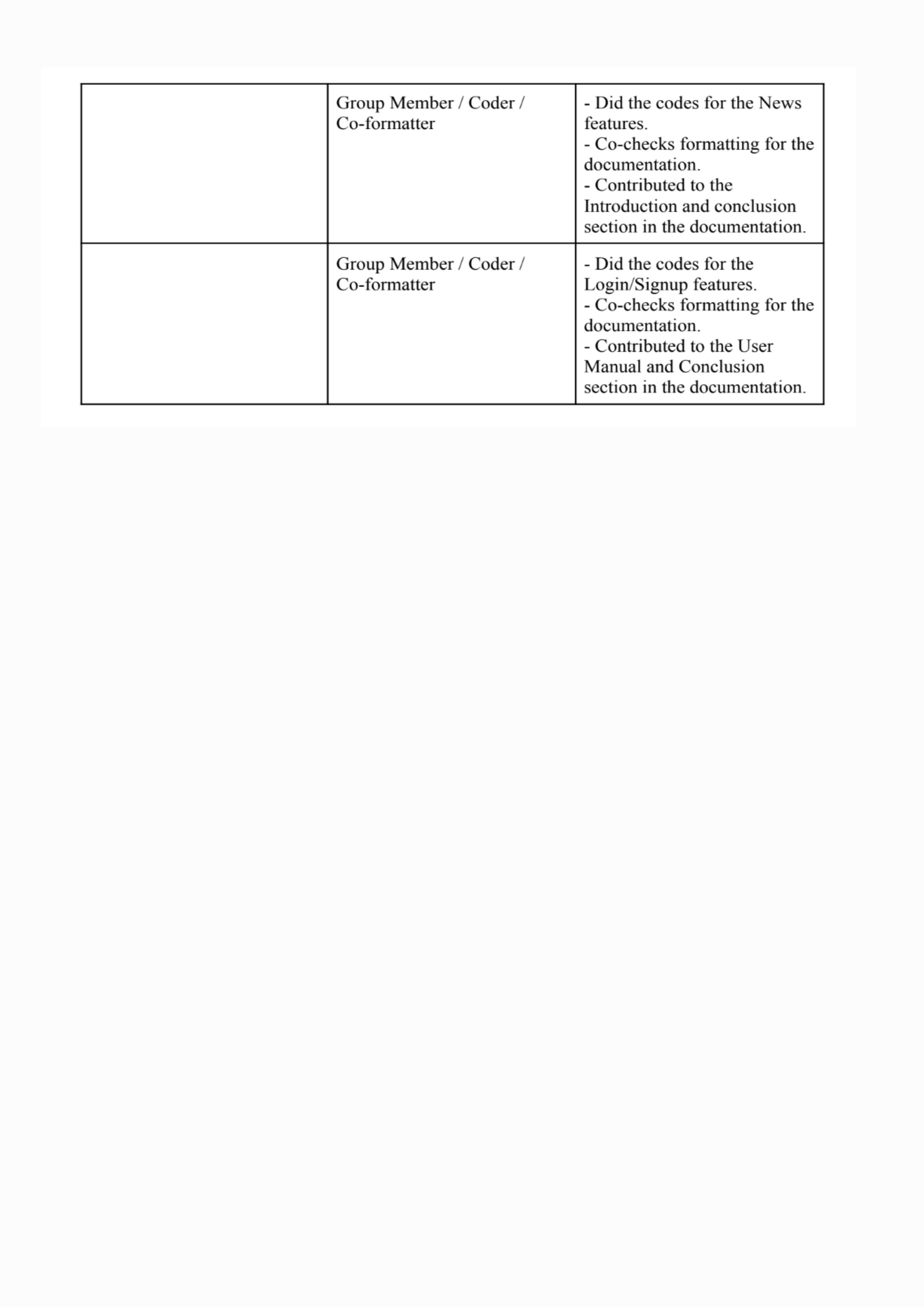


4

**ROLES AND RESPONSIBILITY**

|  |  |  |
| --- | --- | --- |
| Name | Role | Responsibility |
|  | Group Leader / Coder | * Plans and arranges meetings to distribute work among other group members. * Did the codes for the admin-specific features. * Contributed to the   User friendliness and Source code explanation in the documentation. |
|  | Group Member / Coder | * Did the codes for the Shop features. * Contributed to the Source code explanation in the documentation. |
|  | Group Member / Coder | * Did the codes for the Profile features. * Contributed to the User friendliness section in the documentation. |
|  | Group Member / Coder | * Did the codes for the Games features. * Did the presentation slides. |

5



# 1.0 INTRODUCTION

## **1.1 Rationale**

### Using Interactive Education and Technology to Address Climate Change

Climate change is one of the most urgent and complex problems facing humanity today. Its wide-ranging impacts affect the environment, public health, global economies, and societal stability. Extreme weather, melting glaciers, biodiversity loss, and rising global temperatures are just a few of the indicators that show how urgently we must all work together. However, the general lack of understanding and awareness about the causes, consequences, and solutions of climate change is a significant barrier to effectively addressing it.

Our project offers a method to address this problem by raising awareness of climate change and promoting preventative actions. The objective is to create an interactive application that bridges the knowledge-action gap by educating users and enabling them to take action.

### Suggested Resolution

Fighting climate change requires collective action, which can only be accomplished if people are informed and inspired to take part. By creating an interactive application that raises awareness of climate change and provides users with useful tools to help overcome this knowledge gap, our initiative aims to close this gap. As a result, we hope to be in line with Sustainable Development Goal (SDG) 13: Climate Action, which emphasises how crucial it is to incorporate climate measures into strategies, policies, and education.

The goal of this project is to inform people about important facets of climate change, such as carbon footprint reduction, renewable energy, water conservation, and deforestation. By offering users easily accessible and captivating instructional resources, the project enables people to make significant progress in the direction of sustainability.

### Using Technology

Object-Oriented Programming (OOP) is the foundation of the application's development, guaranteeing that its codebase is well-structured, scalable, and manageable. Important OOP ideas like modularity, inheritance, and encapsulation let the application to grow with future improvements without interruption. A complex graphical user interface (GUI), real-time data integration, or gamification elements are a few examples of possible improvements.

The efficiency of the application is further enhanced by the usage of gamification in schooling. It has been demonstrated that interactive learning strategies, such point-based systems and trivia games, improve user engagement and knowledge retention. The application guarantees that users actively engage in their learning process instead of passively absorbing knowledge by integrating these components. A blog management system is also included to promote conversations on climate-related subjects and make it easier for useful information to be shared.

### Characteristics and Inclusion

By accommodating both ‘public users’ and ‘admin users’, the application's dual-user model guarantees inclusivity:

1. Admin Users: With the power to create and oversee content, these users allow the application to be tailored to different audiences, including schools, colleges, and community organisations.
2. Public Users: The public can engage with eco-friendly concepts, access instructional content, and take part in entertaining activities like quiz games thanks to an easy-to-use interface.

### Conformity to SDG 13

The project's goals are in perfect harmony with SDG 13, which highlights the urgency of addressing climate change and its effects. Through raising awareness of climate change, educating people, and offering practical answers, the application supports a community-based approach to sustainability.

### The Impact of Technology and Its Prospects

This initiative demonstrates the revolutionary role that technology plays in promoting social transformation. The application encourages people to make educated decisions and support global climate action by using technology to enlighten and inspire. It is a strong and adaptable tool in the battle against climate change because of its modular construction, which guarantees flexibility in response to changing customer needs and technology breakthroughs.

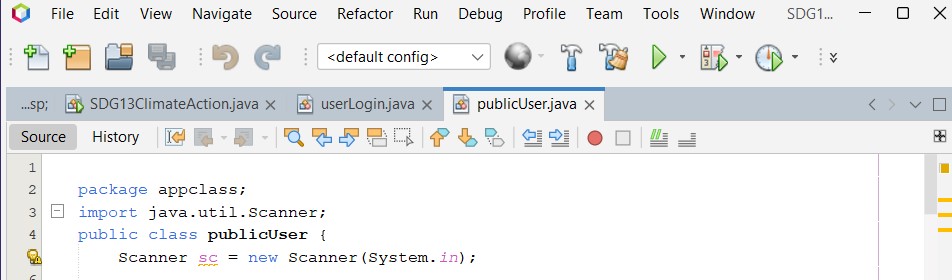
## **Goals**

The primary objectives of this initiative are as follows:

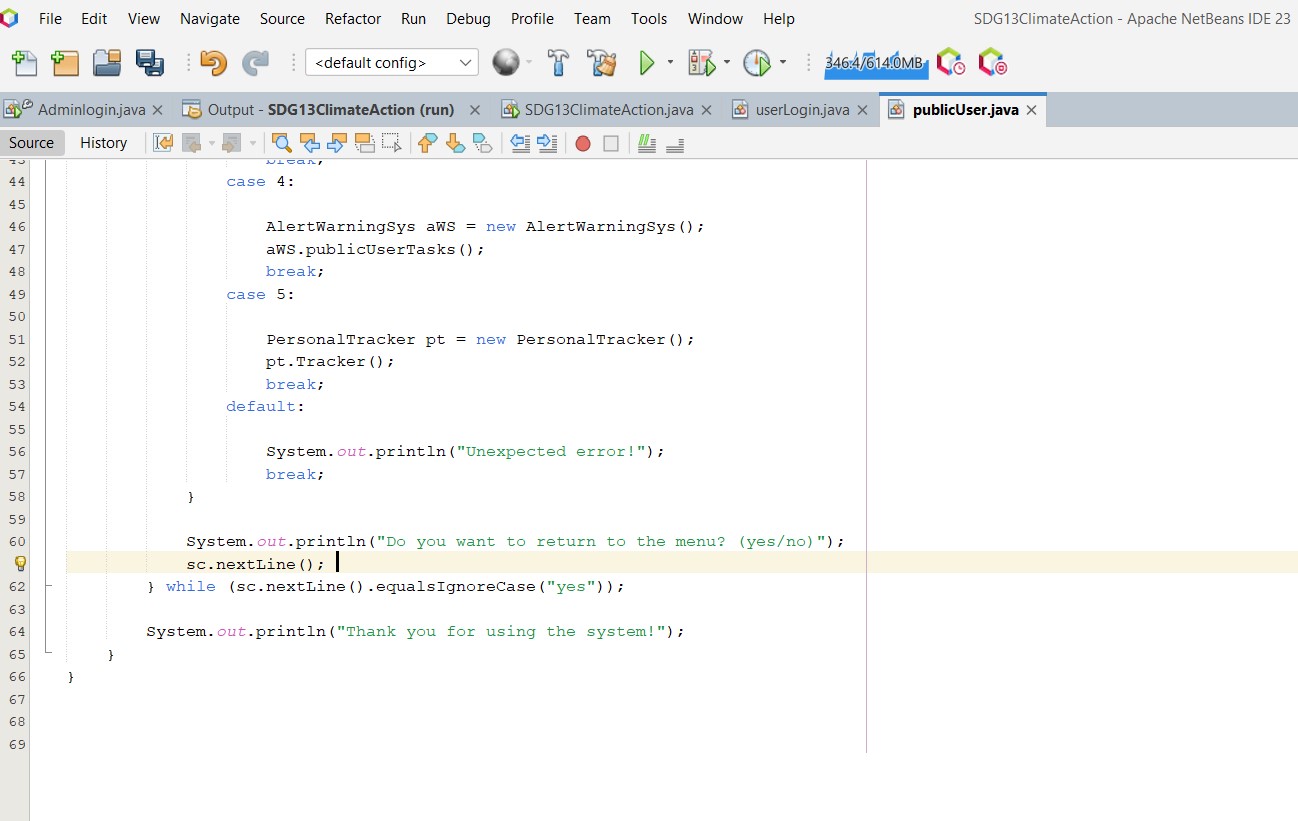
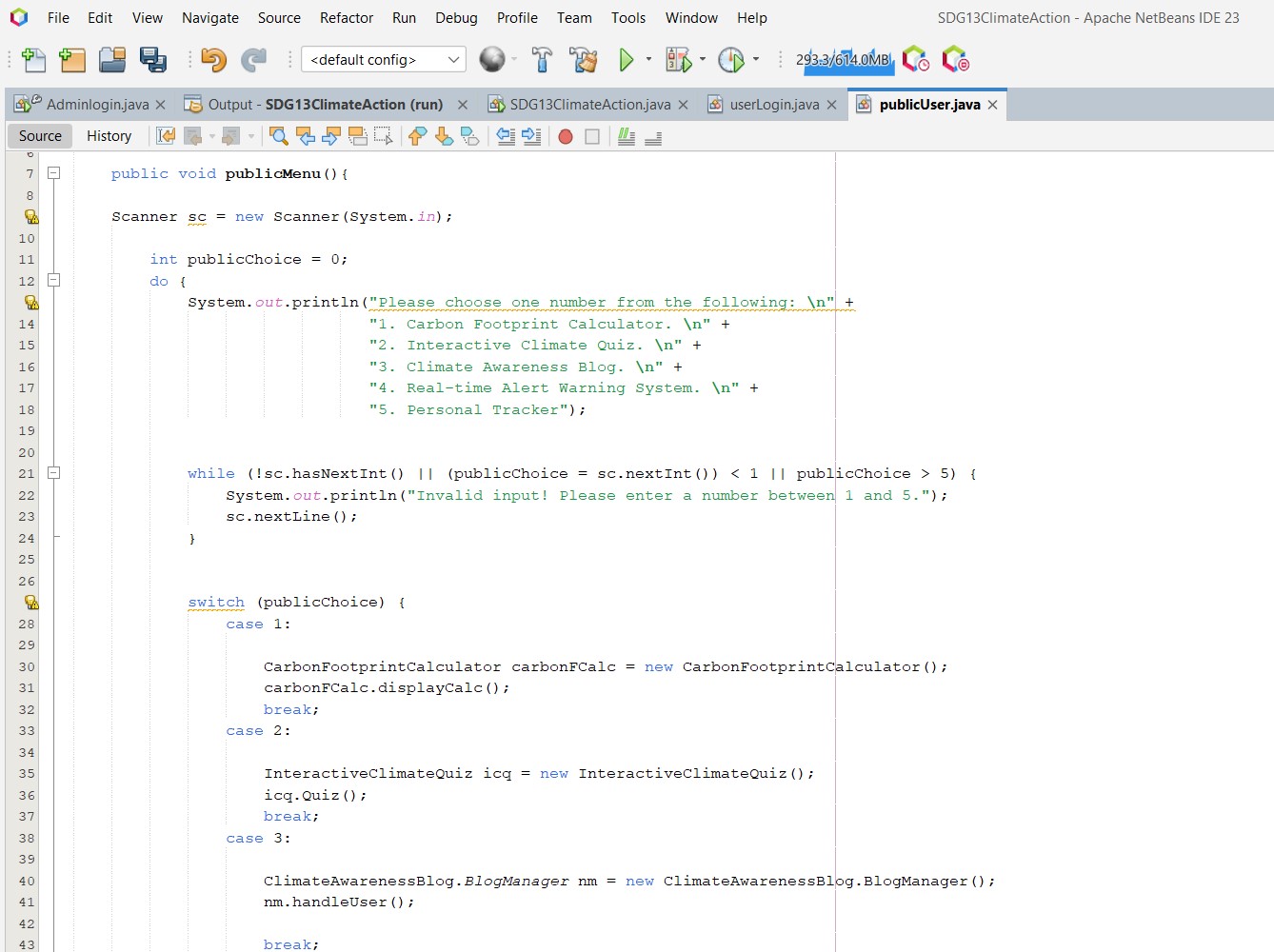
1. Teaching users about the effects of climate change.
2. Making use of OOP concepts to guarantee a codebase that is organised, scalable, and maintainable.
3. Including interactive elements like blog posts & climate awareness quizz.
4. Coordinating the initiative with SDG 13 to advance sustainability and meaningful climate action.

# 2.0 SOURCE CODE AND RESULTS

* publicUser Class:

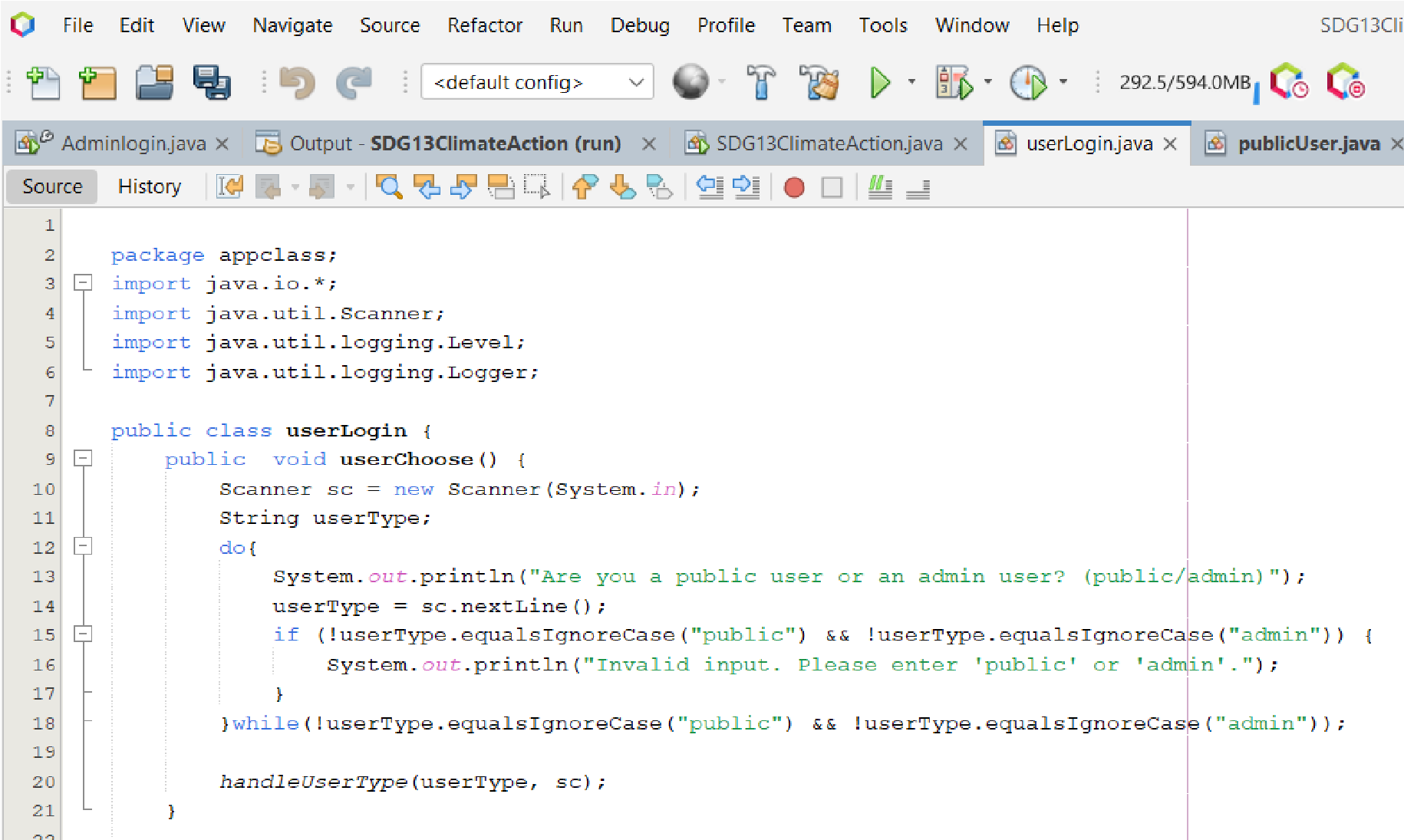


publicUser class manages a menu system for all public user features related to climate action. It uses a scanner function to read user input and navigate between different options such as a Carbon Footprint Calculator, an Interactive Climate Quiz, a Climate Awareness Blog, a Real-time Alert Warning System, and a Personal Tracker.



publicMenu method displays a menu with five main features. It uses a scanner function to read valid user input (1 to 5) and employs a switch statement to execute different methods based on the user’s choice. When the chosen option is fully executed, the method asks the user if they want to return to the menu (yes). The loop will repeat until the user chooses to exit by typing (no).

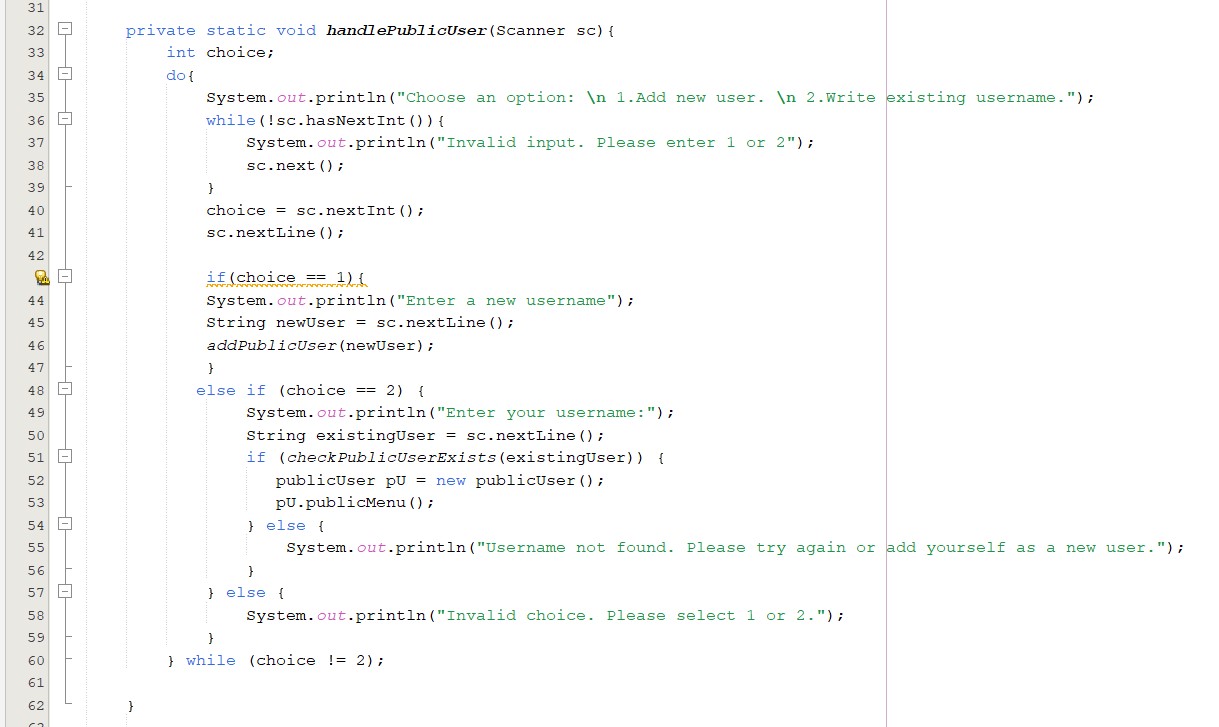
* userLogin Class



userChoose method prompts the user to identify themselves as either a public or admin user and ensures valid input which is based on the user type.



handleUserType method handles the user's type by calling either handlePublicUser or handleAdminUser based on the input received.



handlePublicUser private static method provides options for public users to add new users or use an existing username. It uses a do-while loop to keep asking for user input until a valid choice is made. Inside the loop, it uses an if-else statement to handle the choice.



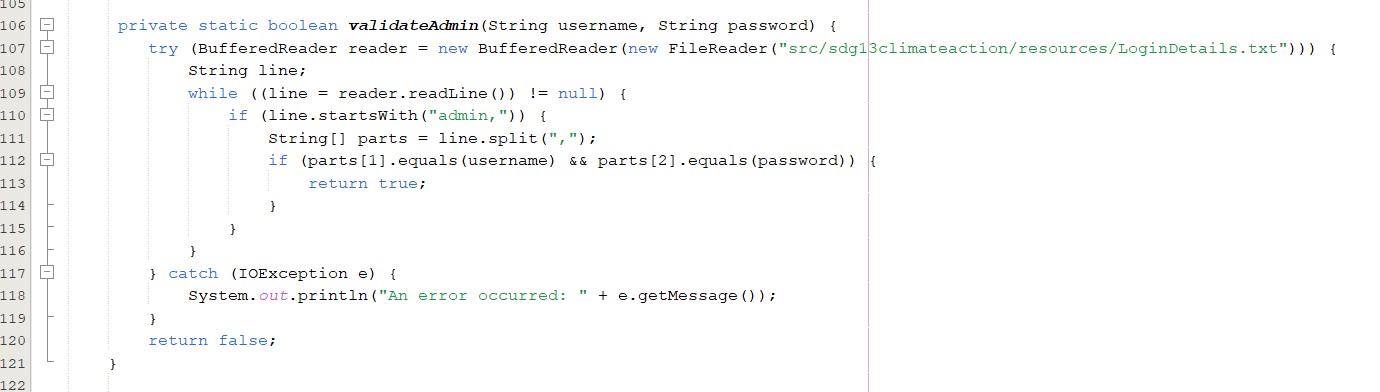
checkPublicUserExists method checks if a public username exists in the login details. It uses a while loop to read each line of the file and an if statement to check if the username matches.



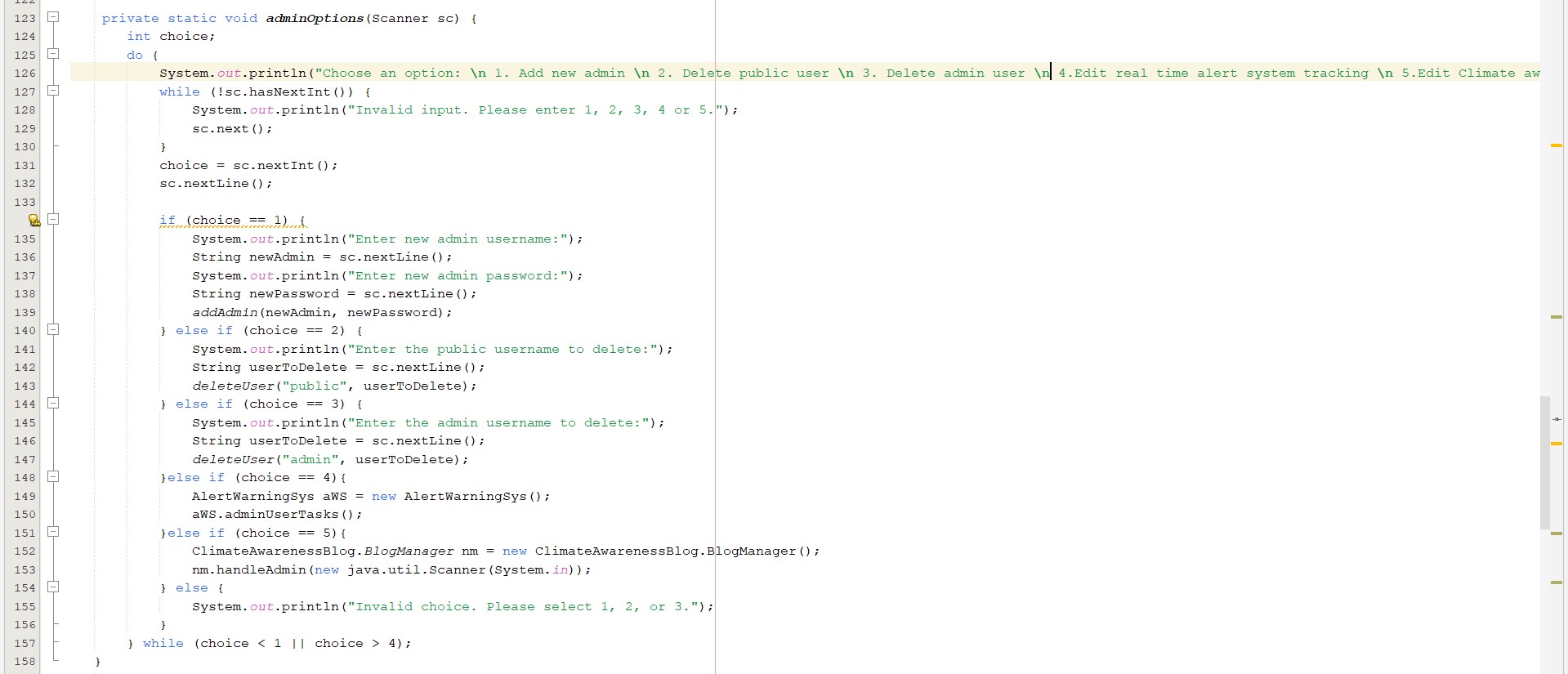
handleAdminUser method prompts the admin user for their username and password. It uses an if-else statement to validate the credentials and either displays admin options or informs the user of invalid credentials.



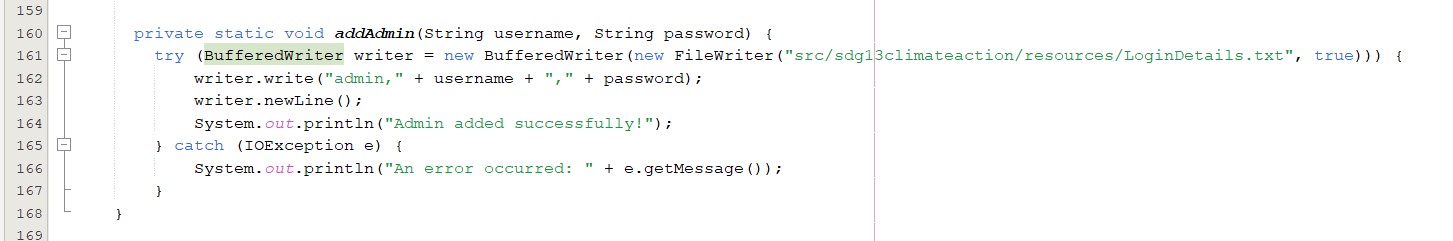
addPublicUser method adds a new public user by writing their username to the login details file. It uses a try-with-resources statement to handle file operations.



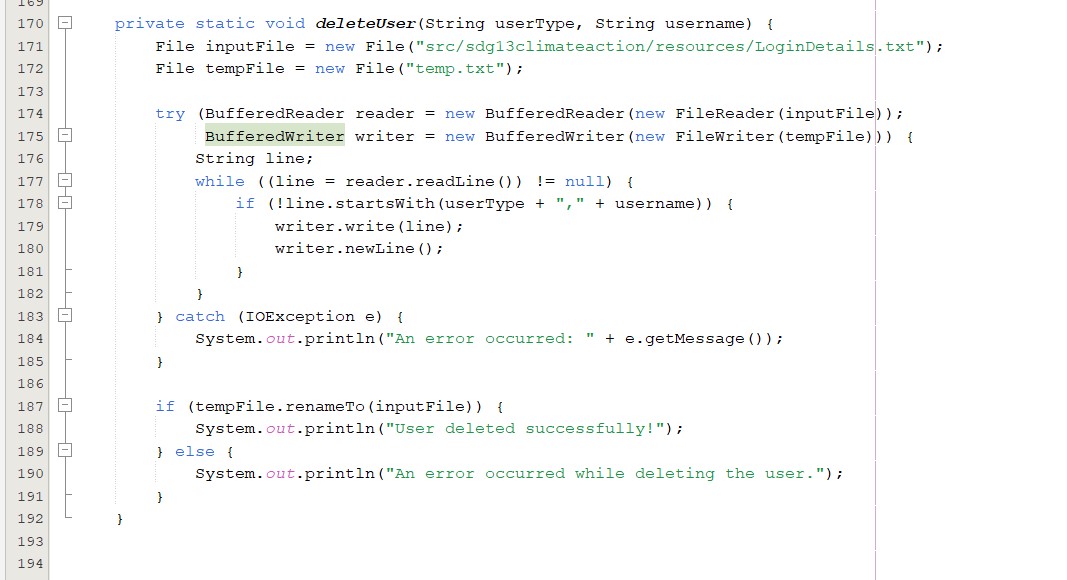
validateAdmin method checks if the admin credentials are valid by reading the login details file. It uses a while loop to read each line and an if statement to check the credentials.



adminOptions method provides options for admin users to manage the system. It uses a do-while loop to keep displaying the options until a valid choice is made and a switch statement to execute actions based on the user's choice.

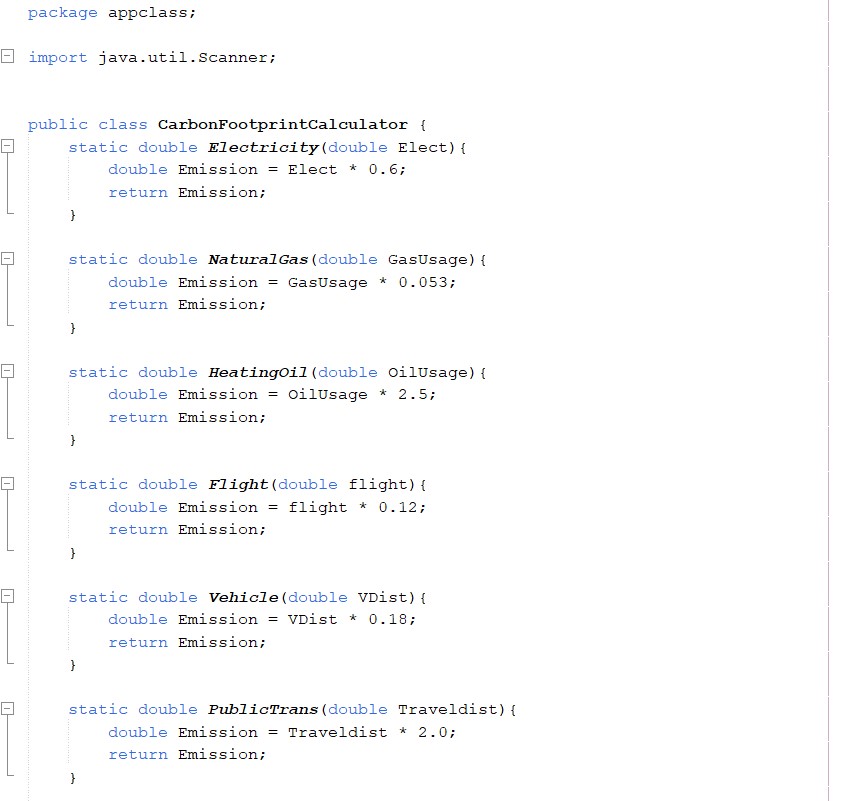


addAdmin method adds a new admin user by writing their username and password to the login details file. It uses a try-with-resources statement to handle file operations.

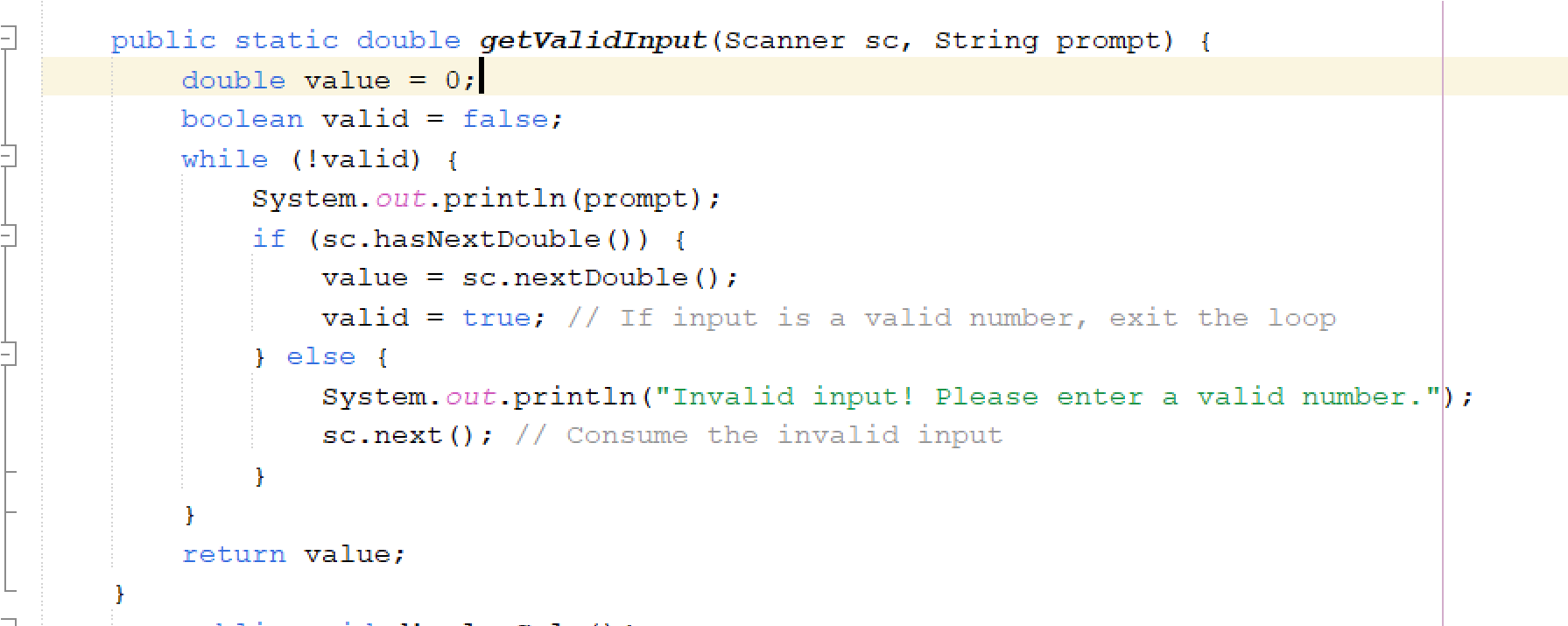


deleteUser method deletes a user (either public or admin) by removing their details from the login details file. It uses a try-with-resources statement for file operations and a while loop to read each line of the files until the end, writing only the lines that do not match the user to be deleted.

## ● CarbonFootprintCalculatorClass



CarbonFootprintCalculator class calculates CO2 emissions based on various activities. It contains methods for calculating emissions from electricity, natural gas, heating oil, flights, vehicle travel, and public transport.



getValidInput method ensures the user enters a valid numeric input by repeatedly prompting until a valid number is entered.

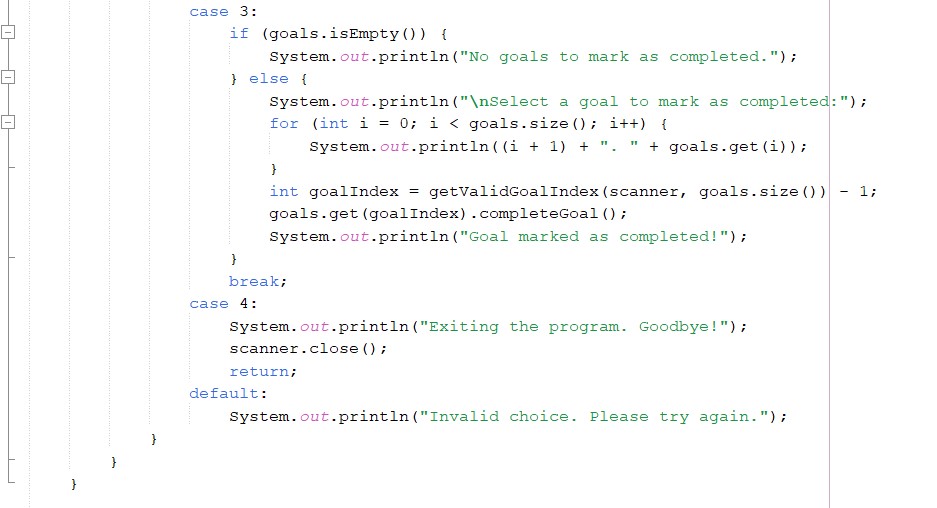
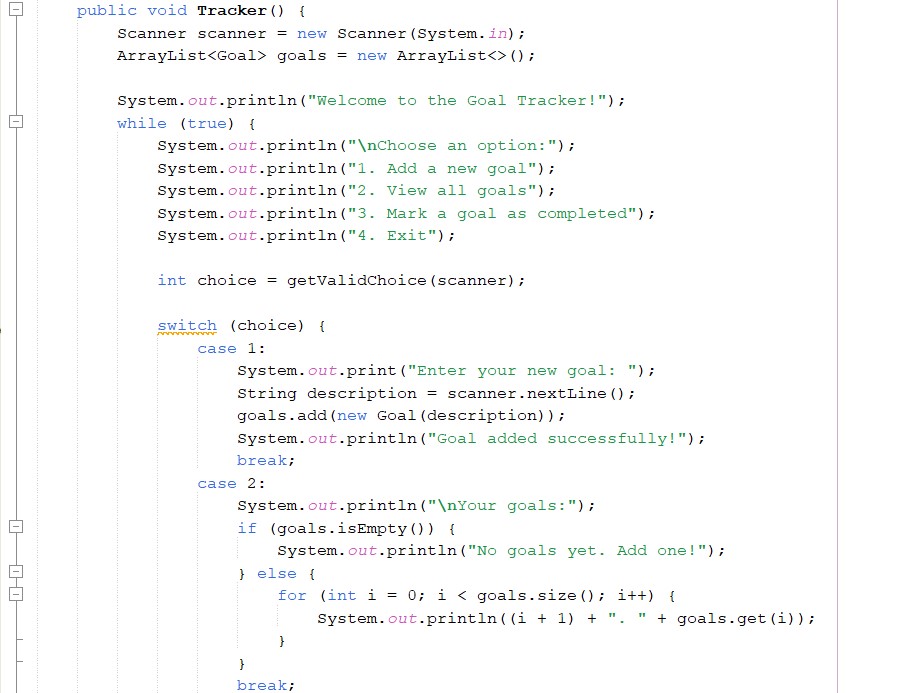


displayCalc method interacts with the user, prompting for inputs related to various activities, calculates the total CO2 emissions, and displays the result rounded to two decimal places.

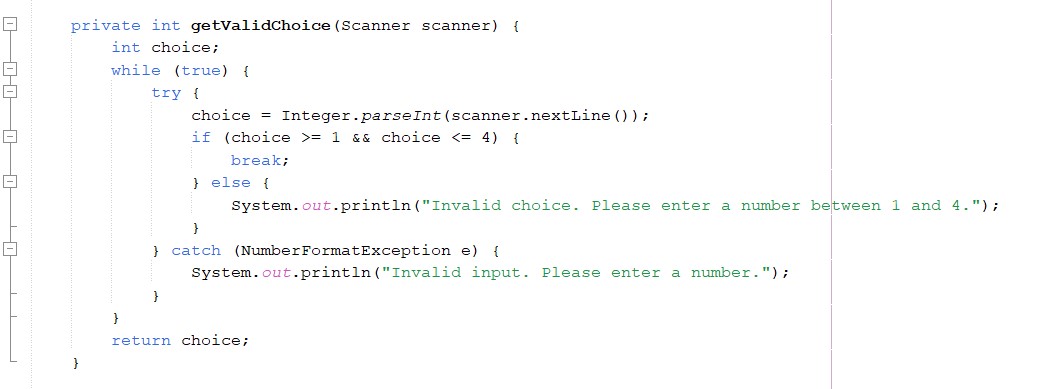
## ● PersonalTracker Class



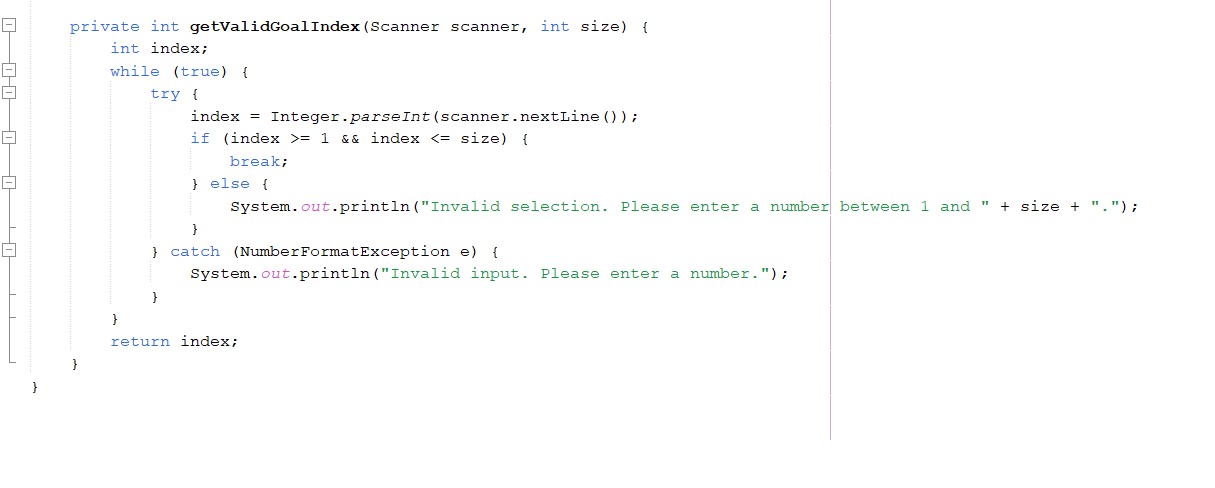
Goal class is used to represent individual goals. It has a constructor that initializes the goal with a description and sets the completion status to incomplete. The getDescription method returns the description of the goal, while isCompleted checks if the goal has been completed. The completeGoal method marks the goal as completed, and the toString method returns a string representation of the goal, indicating its completion status.



Tracker method is the main interface for managing goals. It uses a Scanner to read user input and an ArrayList to store goals. The method runs an infinite loop, presenting the user with options to add new goals, view all goals, mark goals as completed, or exit the program. User choices are handled using a switch statement. When the user adds a new goal, they enter a description, which is then added to the list. Viewing goals displays all current goals, and marking a goal as completed updates its status.

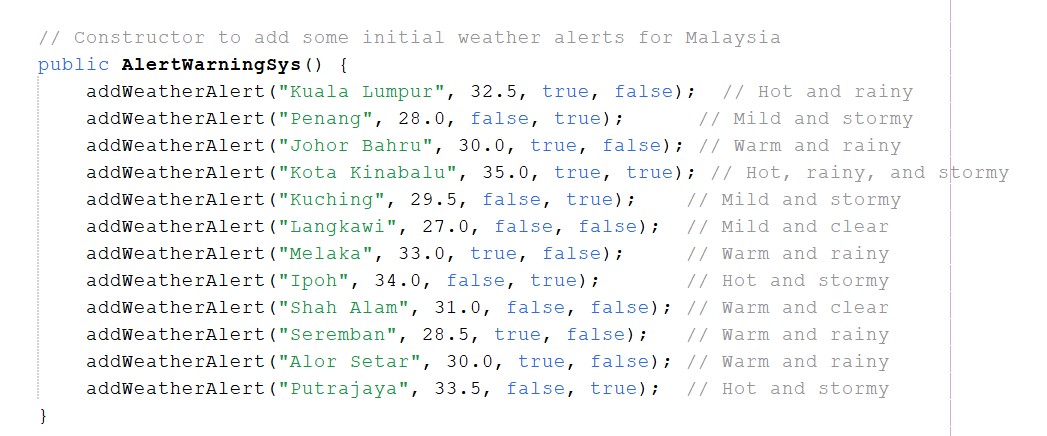


The getValidChoice method ensures the user enters a valid menu choice by repeatedly prompting until a valid input (between 1 and 4) is entered.



The getValidGoalIndex method ensures the user selects a valid goal index from the list by repeatedly prompting until a valid index is provided.

● AlertWarningSystem class



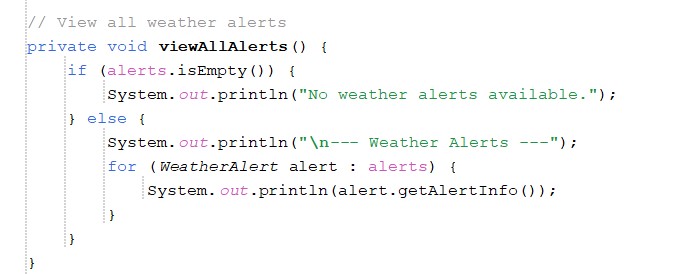
1. Constructor： Use the **public AlertWarningSys()** method to initialize the weather alert system and add some default weather alert data



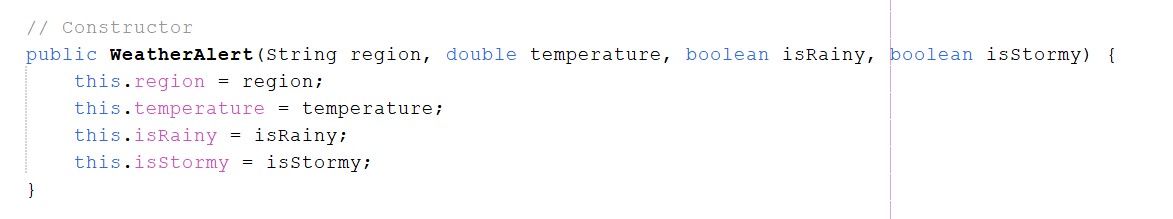
2.Public User Methods：Use the **public void publicUserTasks()** method to provide a menu for public users, where they can choose to view all weather alerts or search for weather alerts by region.



3.Admin User Methods：Use the **public void adminUserTasks()** method to provide a menu for administrators to view, update, or delete weather alerts.

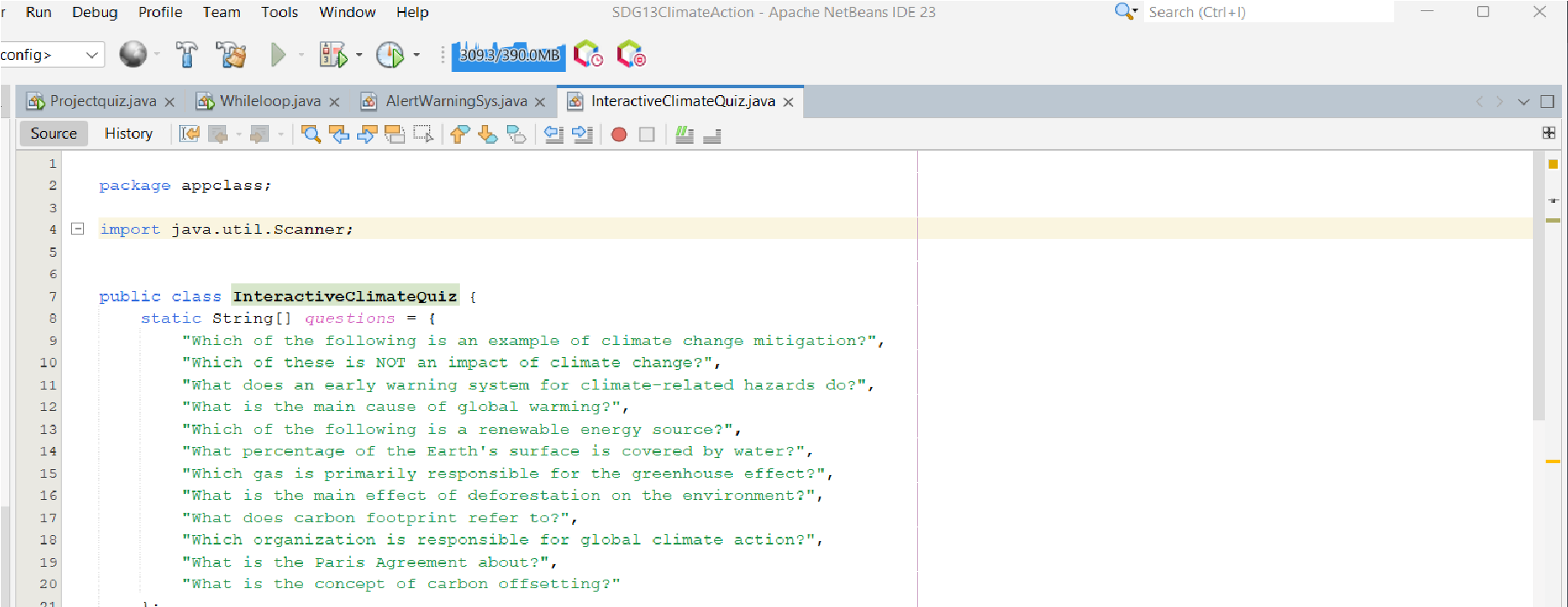


4.Private Methods：Use the **private void viewAllAlerts() private void searchAlertByRegion() private void updateAlert() private void deleteAlert() private void addWeatherAlert(String region, double temperature, boolean isRainy, boolean isStormy)** methods to view all current weather alerts, search for weather alerts by region name, update weather alert information for a specified region, delete weather alerts for a specified region, and add new weather alerts to the alert list.

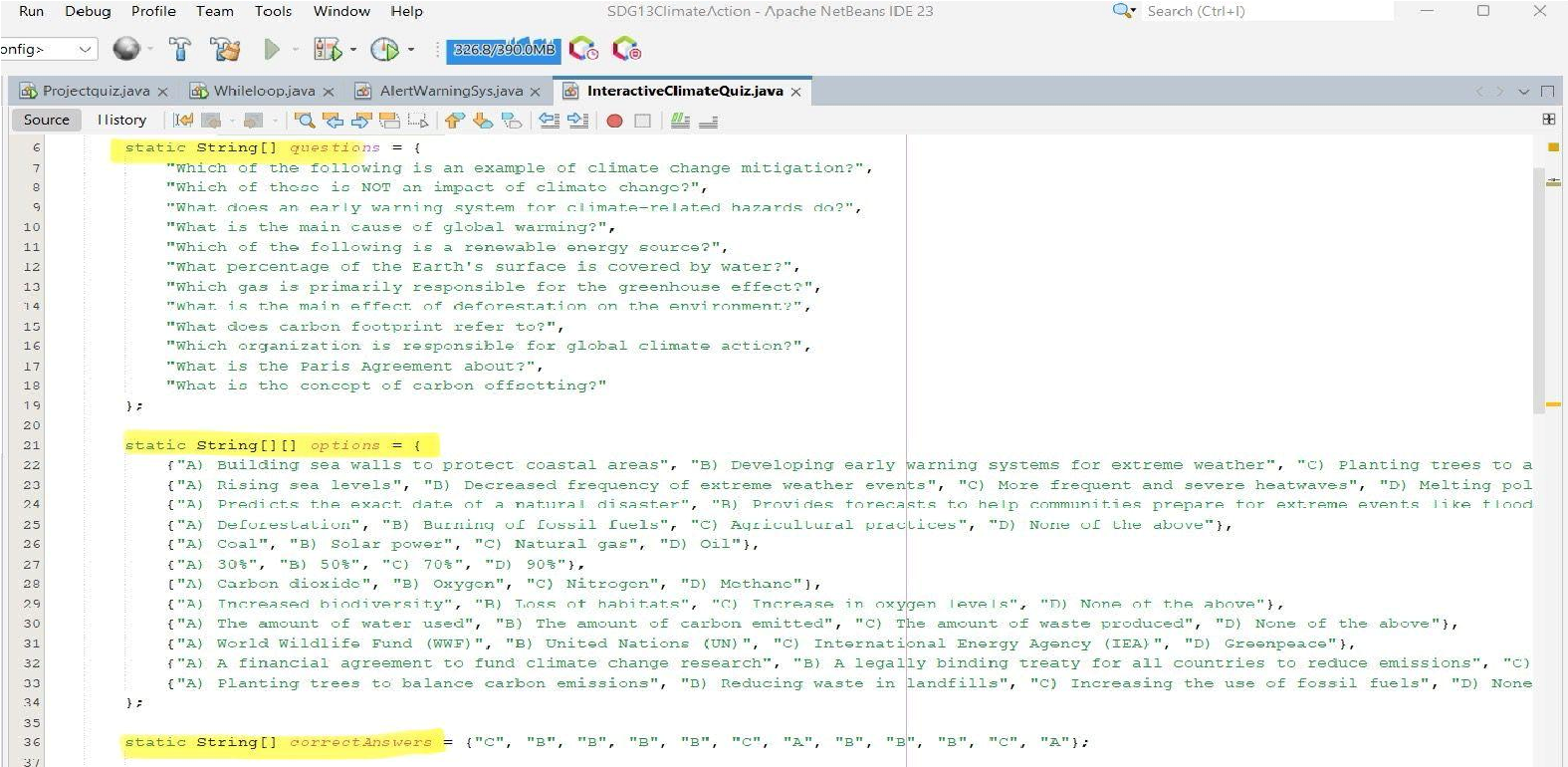


5.Methods in the Nested Class WeatherAlert:Use the **public WeatherAlert(String region, double temperature, boolean isRainy, boolean isStormy)** method to initialize the WeatherAlert object.

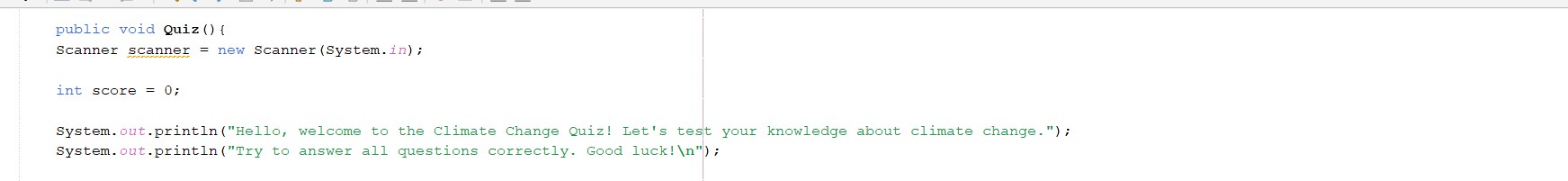
## **● Interactive Climate Quiz class**



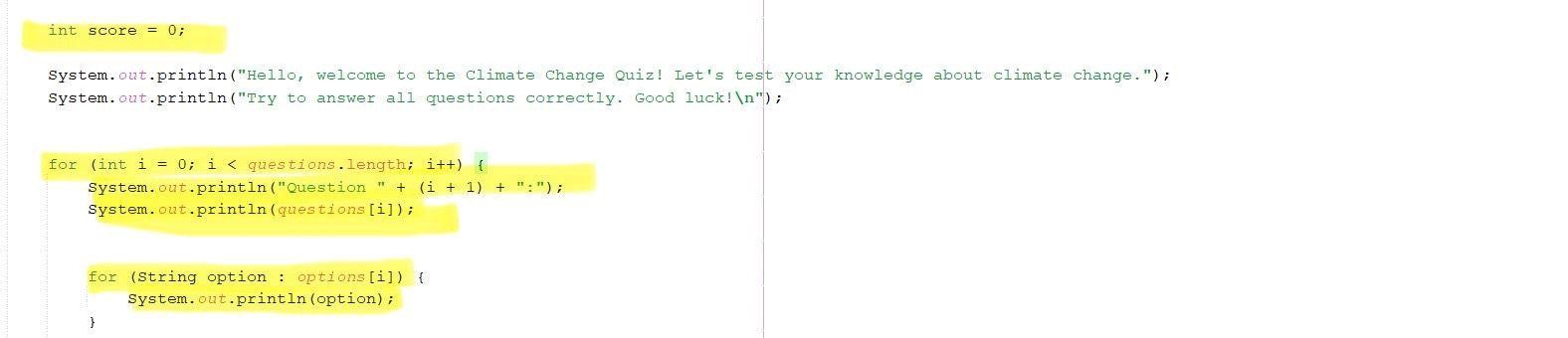
The InteractiveClimateQuiz class has a **Scanner** for getting input from the users and captures the users answers during the quiz , also enables user interaction and allows us to check whether the users answers correct or not .



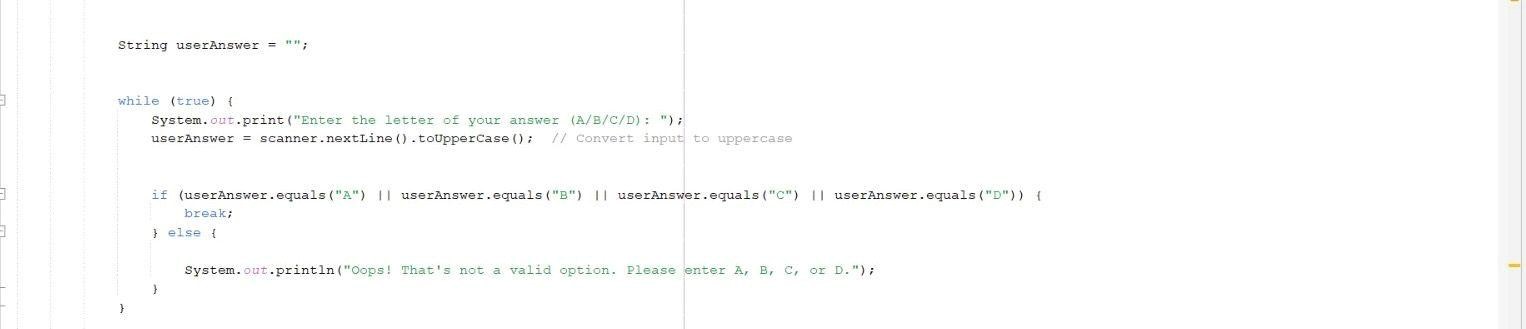
Here we used three **Arrays** for the same data type **String []** to make the code more organized , the first one for stores the quiz questions and the second one for stores the multiple choice options (A,B,C,D) for each question, the last array it is for stores the correct answers for each question .



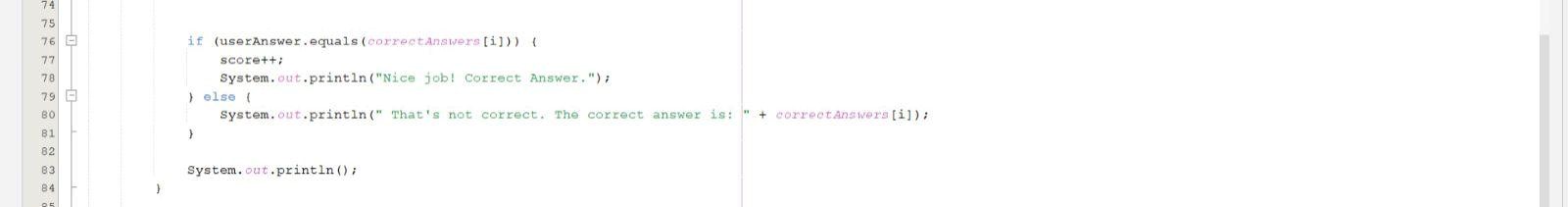
The **Quiz() method this is the main method** that drives the quiz , and thee first displays a welcome message to the user .



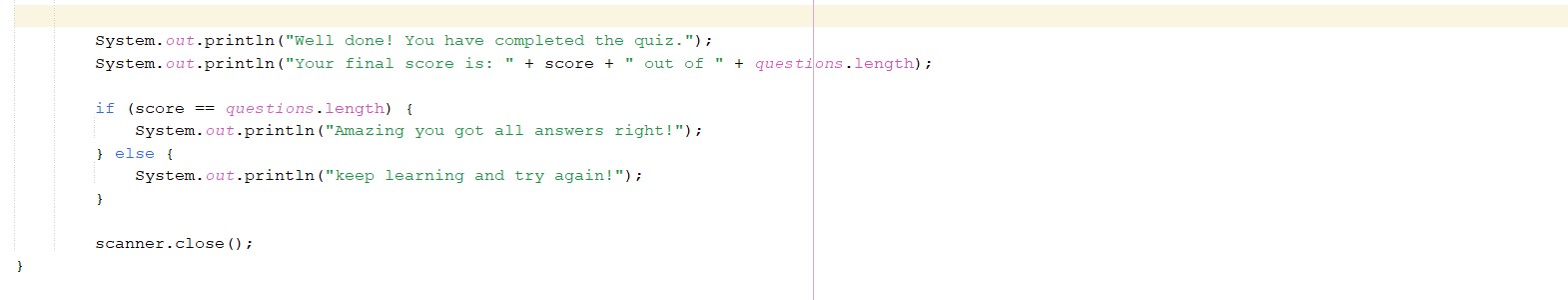
We used two for loop , the first one **(int i = 0; i < questions.length; i++)** for iterates through the questions array , the second loop enhanced for loop which is **(String option : options[i])** used to iterate through the answer options for each question .



A **while loop** for make sure a valid answer which can only be A or B or C or D add to that It also changes the input to uppercase to avoid problems with case sensitivity If the user gives an invalid answer, the program will ask them to enter a valid option again and it will keep doing this until the user provides a correct choice



In this part of the code, the program compares the answer the user gave with the correct answer from the” correctanswers array”. If the user answer is correct, their score increases by 1, and they get a message saying "nice job! correct answer " and If the answer is wrong the program shows the correct answer so the user knows .



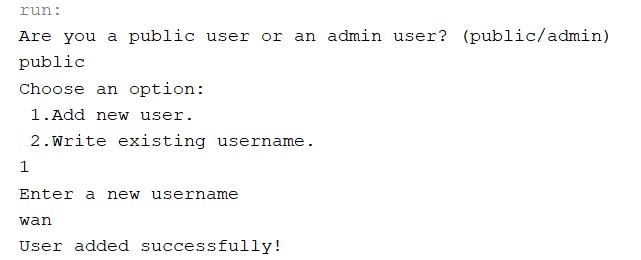
After the user has answered all the questions the program displays their final score. If the user answers all the questions correctly, they will get a message saying "amazing you got all answers right!" If the user doesn't answer all the questions correctly, they are encouraged to keep learning and try again.

# 3.0 USER FRIENDLY

Our program embodies user-friendliness through the following components:

## ❖ 3.1 Effectiveness

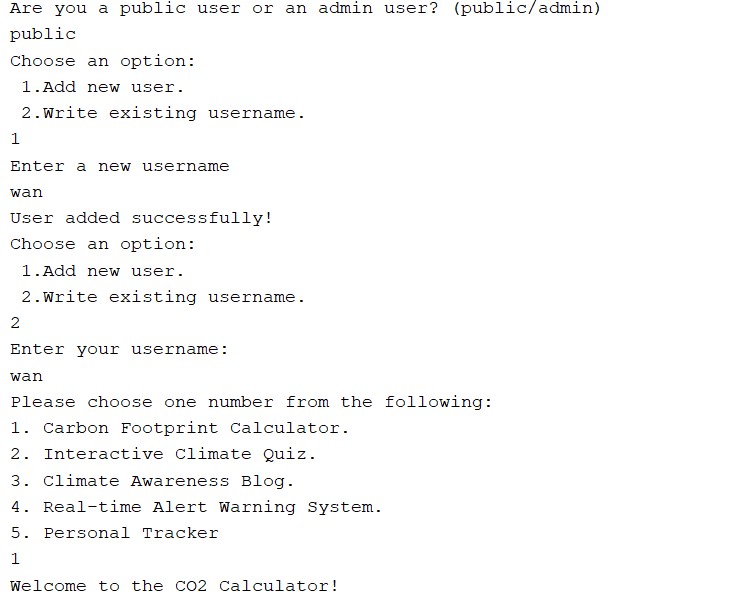
**Effectiveness**: Our program guarantees effectiveness by providing simple and easy-to-use interfaces. For example, each menu and prompt is designed to be straightforward. Space and use of text formatting have been effectively used to ensure that information is easy to read and navigate.



***Figure: The use of space and text formatting***

## ❖ 3.2 Efficiency

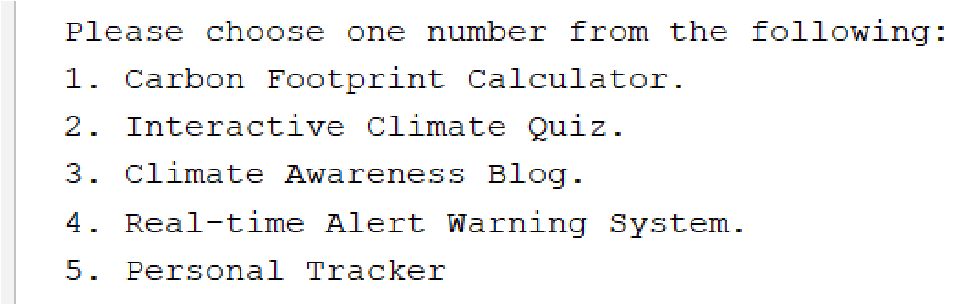
**Efficiency**: The use of intuitive commands, including numbers to enter when scrolling through menus has made the use of the program more efficient. By using the single digit, users can get familiar with the operation of the system and learn to navigate all the functions quickly.



***Figure: Usage of single-digit command***

## ❖ 3.3 Engagement

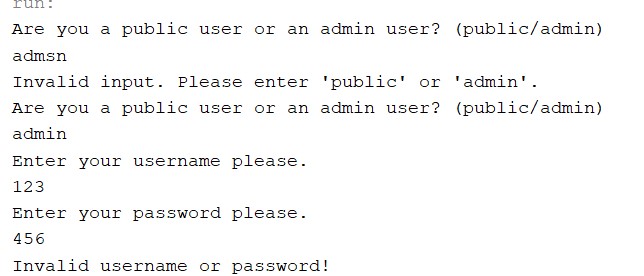
**Engagement**: The program includes features that engage users interactively, such as the Climate Quiz and the Personal Tracker. These features not only educate users about climate change but also keep them motivated and interested through interactive elements and progress tracking.



***Figure: Example of engagement features***

## ❖ 3.4 Error Tolerance

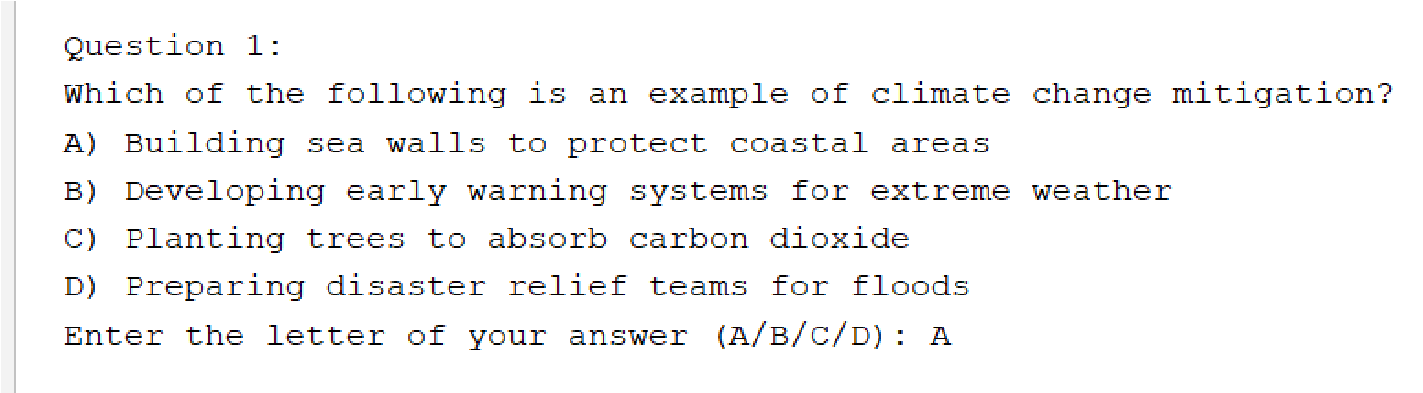
**Error Tolerance**: Our program is designed to handle user input errors gracefully. By validating inputs and providing clear error messages, we ensure that users can correct mistakes without frustration. Furthermore, techniques such as while loops and conditional statements will also prevent invalid inputs from causing crashes or unwanted behavior.



***Figure: Example of error***

## ❖ 3.5 Ease of Learning

**Ease of Learning**: Instructions and prompts are clearly displayed throughout the program, guiding users step by step. This minimizes the learning curve and allows users to become proficient quickly. For example, prompts like "Enter the letter of your answer (A/B/C/D)" in the quiz are easy to understand and follow.

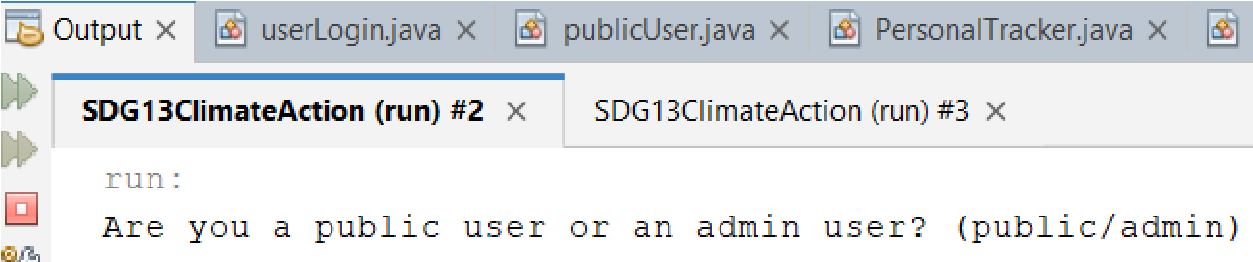


***Figure: Example of easy-to-understand instruction***

# 4.0 USER MANUAL

**● 4.1 Feature 1: Login & Sign Up**

## ❖ 4.1.1 User Identification



### Figure: Login example

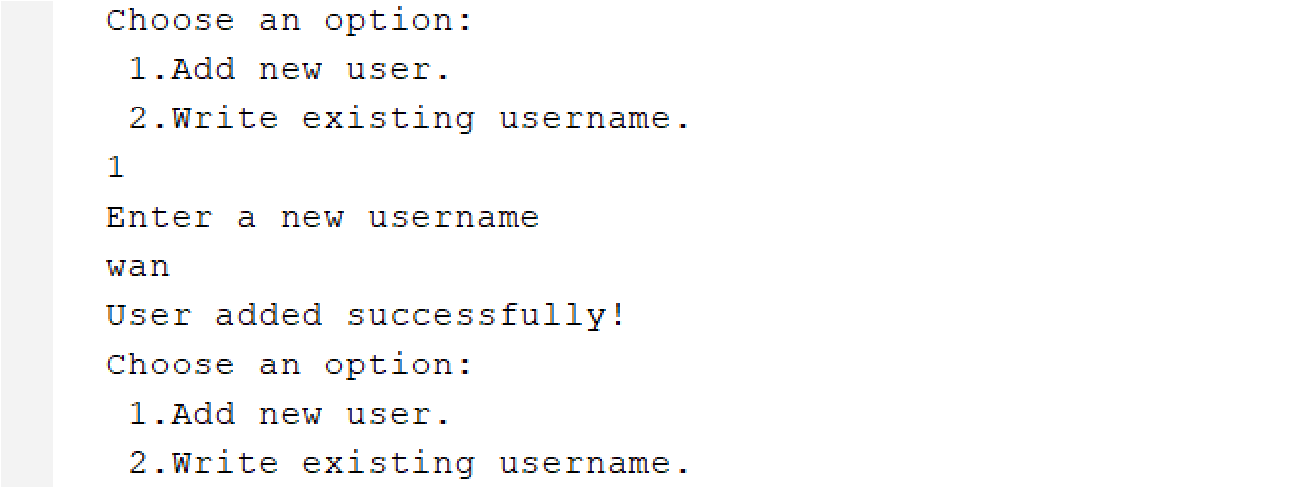
When you start the application, you will be prompted to identify yourself as a public user or an admin user.

## ❖ 4.1.2 Public User Login

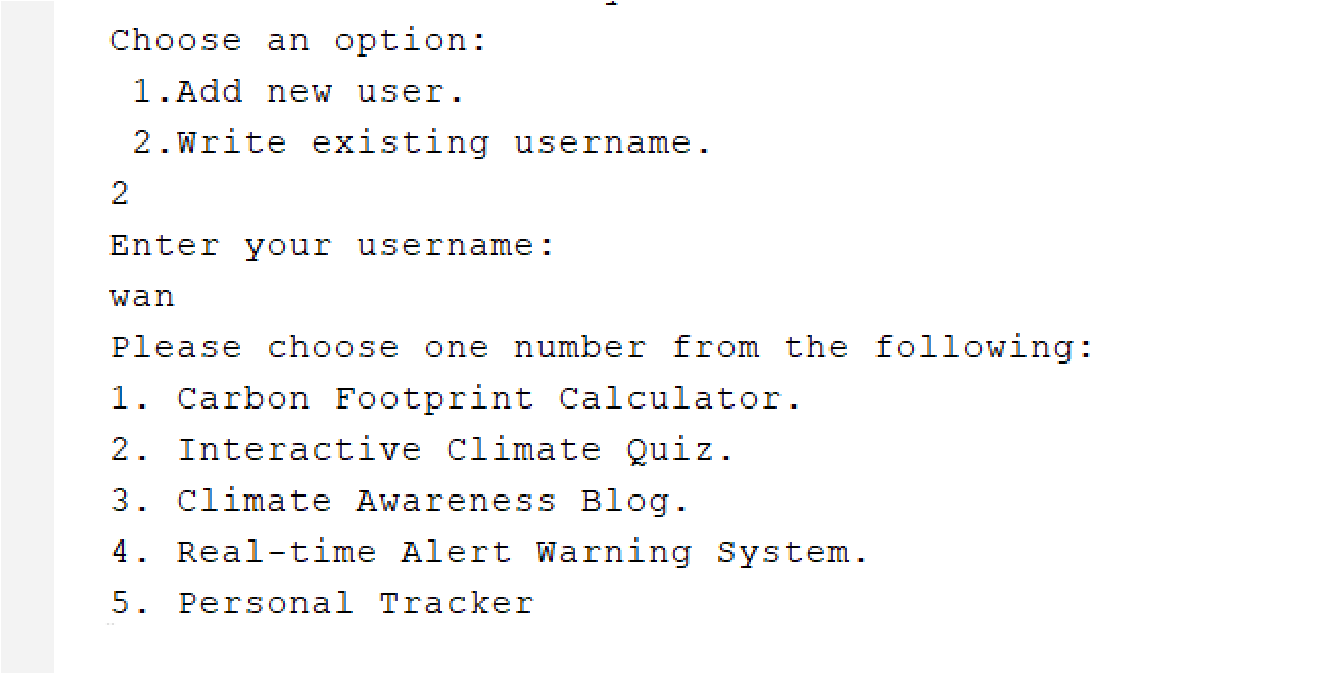


### Figure: Admin login

If you choose to be a public user, you will be given options to either add a new user or login with an existing username. The add new user function is to create a new public user account while writing existing username function is to log in by using the existing account username.

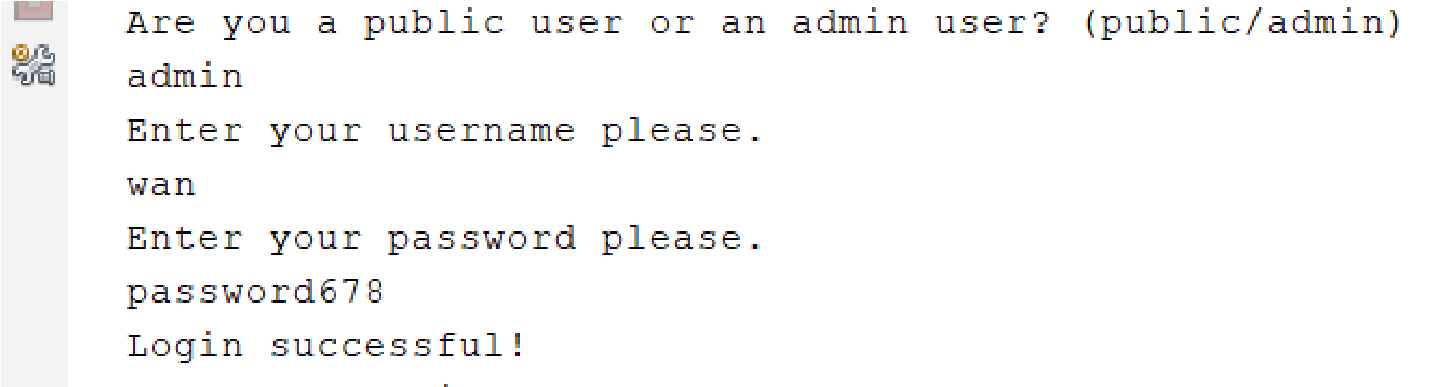


### Figure: Add new user

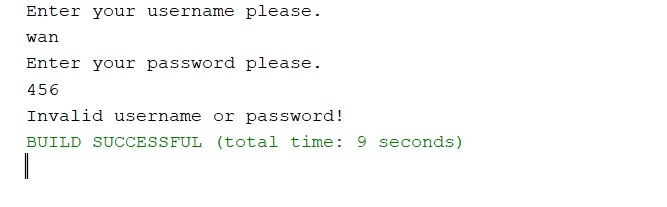


***Figure: Login with an existing username***

❖ 4.1.3 Admin Login



### Figure: Successful login

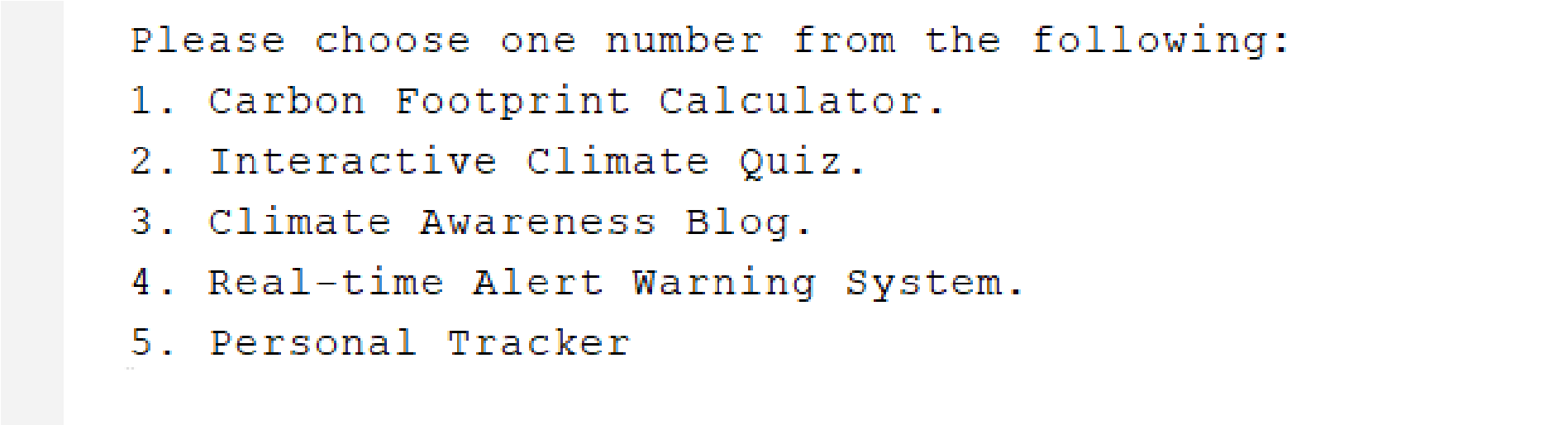


### Figure: Unsuccessful login

If you choose to be an admin user, you will need to enter your admin username and password to log in. If the credentials are correct, you will see a login success message and proceed to the admin menu. Otherwise, you will be prompted invalid username or password and the system break.

## **● 4.2 Feature 2: Public User Menu**

After logging in as a public user, you will access the public user menu with several options



### Figure: Public user menu

**Carbon Footprint Calculator**: Calculate your carbon footprint based on your activities.

**Interactive Climate Quiz**: Test your knowledge about climate change through a quiz.

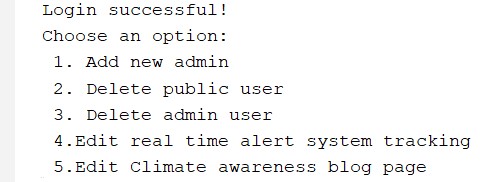
**Climate Awareness Blog**: Read articles and posts related to climate awareness.

**Real-time Alert Warning System**: View weather alerts and warnings.

**Personal Tracker**: Set and track personal goals related to climate action.

## **● 4.3 Feature 3: Admin Menu**

After logging in as an admin, you will access the admin menu with various management options.



### Figure: Admin menu

**Add New Admin**: Add a new admin user by entering a username and password.

**Delete Public User**: Remove an existing public user by entering the username.

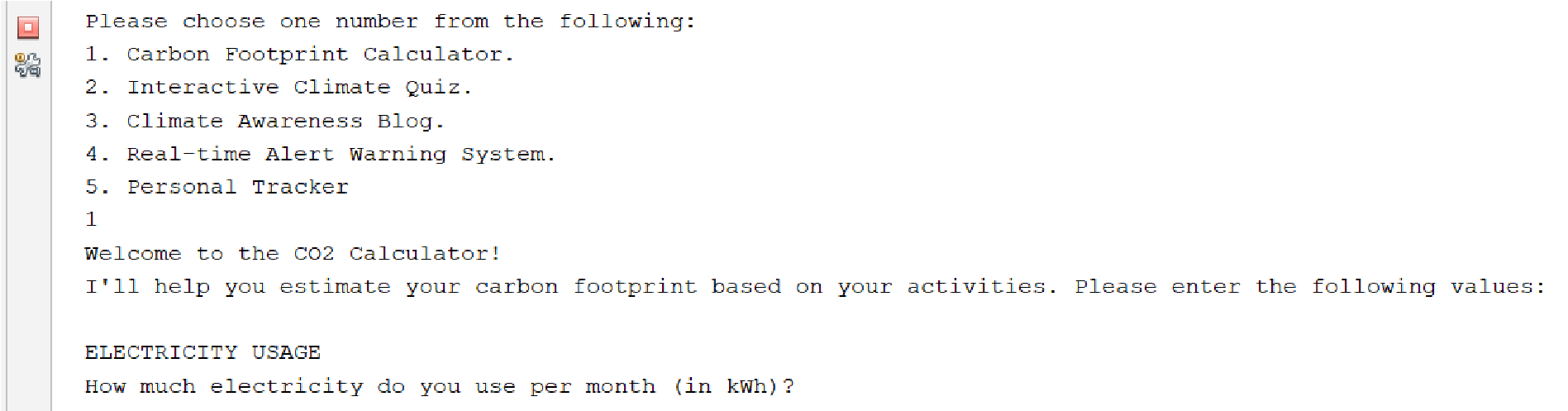
**Delete Admin User**: Remove an existing admin user by entering the username.

**Edit Real-time Alert System Tracking**: Update or delete weather alerts.

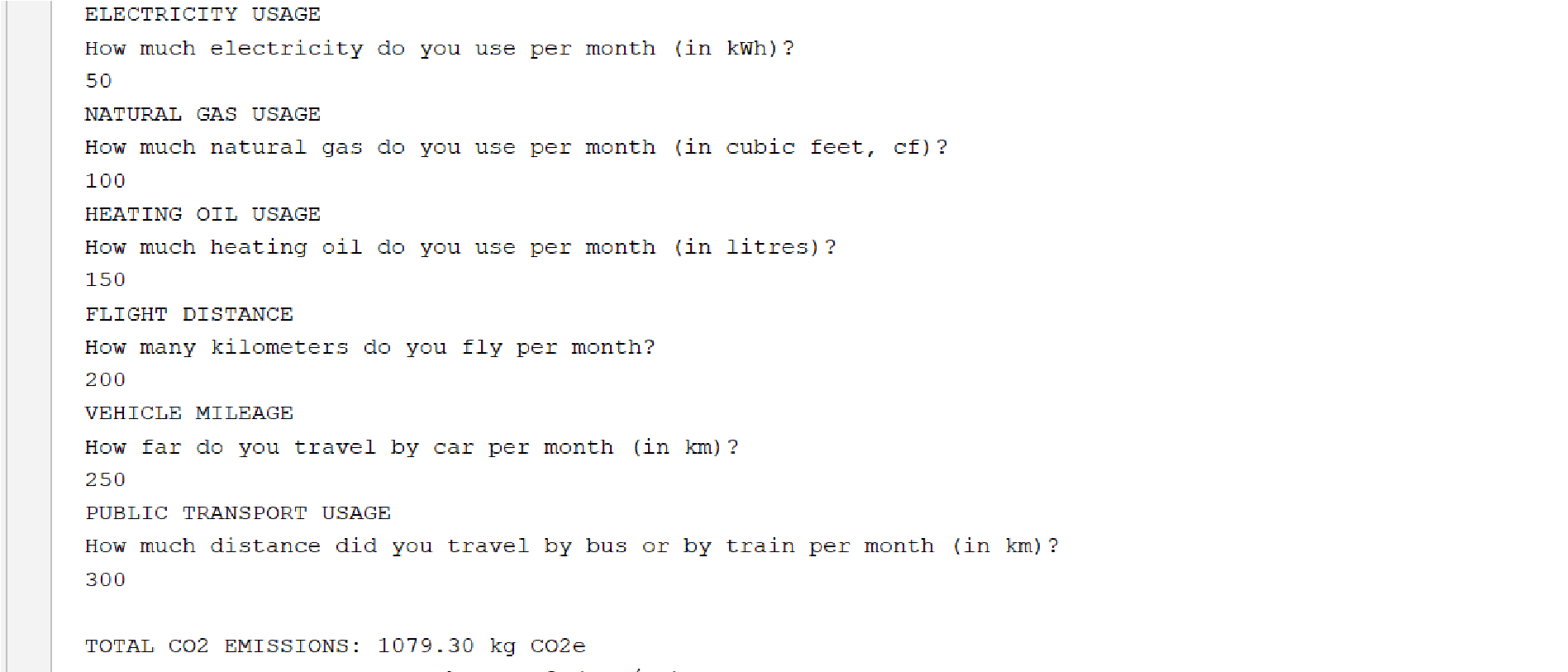
**Edit Climate Awareness Blog Page**: Manage blog posts, including adding, editing, or deleting posts.

## **● 4.4 Feature 4:Carbon Footprint Calculator**

This section explains how to use the Carbon Footprint Calculator. When you select the Carbon Footprint Calculator option from the public user menu, you will be prompted to enter several values related to your activities.



***Figure: Prompt for input***



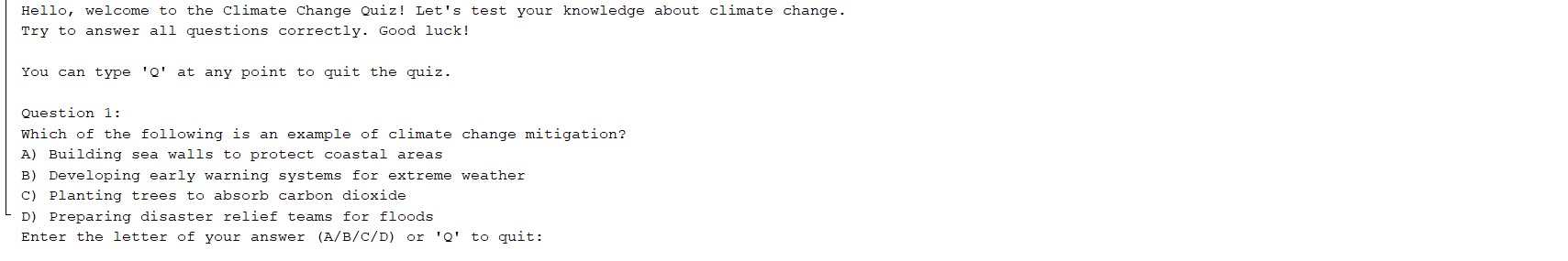
***Figure: Sample question and output***

Once all inputs are provided, the calculator will display the total CO2 emissions based on your usage.

## **● 4.5 Feature 5: Interactive Climate Quiz**

The purpose of this is to create an interactive quiz about climate change. The program allows the user to answer multiple choice questions and provides feedback based on their responses.

Once the user executed this program the welcome message will show and the quiz will start display the question with the options for each question and the Q letter purpose In case the user wants to finish the quiz and does not want to complete it



If the user answer its correct Nice job message will display and if the answer incorrect not correct message will show with the correct choice ز



At the end of the quiz when the user completes the last question , the final score will display .

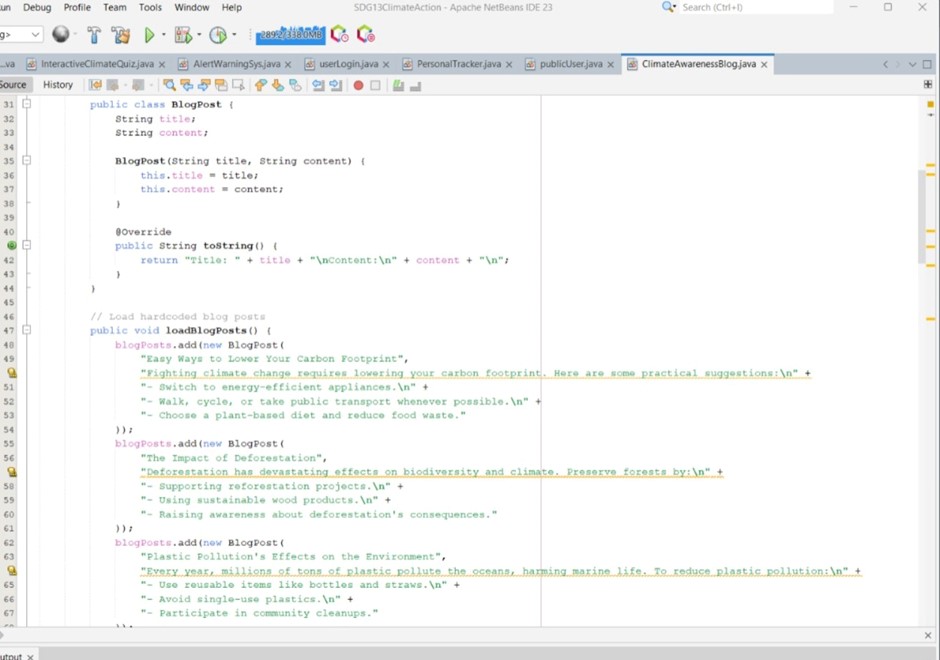


## **● 4.6 Feature 6: Climate Awareness Blog**

**BlogPost Class:**

* Individual blog articles are defined via the BlogPost class using properties like title and content.
* Private fields (title, content) and public getter and setter methods (getTitle, setTitle, getContent, setContent) are used to illustrate encapsulation and guarantee controlled access to the data.
* Each blog post may be represented as a formatted string using the toString method, which is helpful for showing blog details.
* Additionally, the screenshot displays a loadBlogPosts method that simulates beginning data for testing or demonstration by adding hardcoded blog entries.

**Code Snippet:**



**Explanation:**

An essential component of object-oriented programming (OOP) is encapsulation, which makes sure that an object's internal state is shielded from unwanted or unauthorised access. Encapsulation protects data integrity and security by limiting direct access to a class's characteristics and permitting only meaningful and controlled changes through expressly specified methods.

Encapsulation is accomplished in the BlogPost class by designating the attributes title and content as private. This guarantees that these characteristics cannot be directly accessed or changed from outside the class. Rather, the class offers public getter and setter methods that enable restricted access and adjustments to the attributes, including getTitle, setTitle, getContent, and setContent.

For instance:

Controlled changes to a blog post's title attribute are made possible with the setTitle method. This guarantees that any title modifications are deliberate and follow the guidelines specified in the procedure.

Similar to this, the setContent technique makes sure that changes are made to a blog post's content in a controlled and predictable way, avoiding unintentional or malevolent alterations.

The BlogPost class gains the following advantages by encapsulating these characteristics:

Data Integrity: By limiting direct access to important attributes, the possibility of

unintentional data corruption is decreased.

Controlled Access: By offering a transparent interface for working with the object's contents, the getter and setter methods make sure that alterations are significant and, if required, validated.

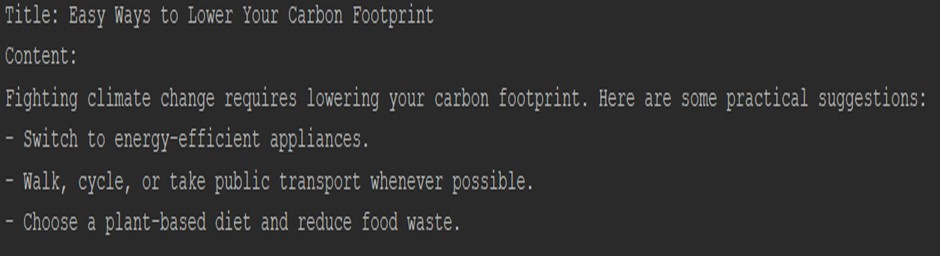
Flexibility: Encapsulation permits modifications to the class's internal implementation

without impacting the code that utilises it externally. In the future, for instance, setTitle or setContent can have more validation criteria added without affecting the functionality that is already in place.

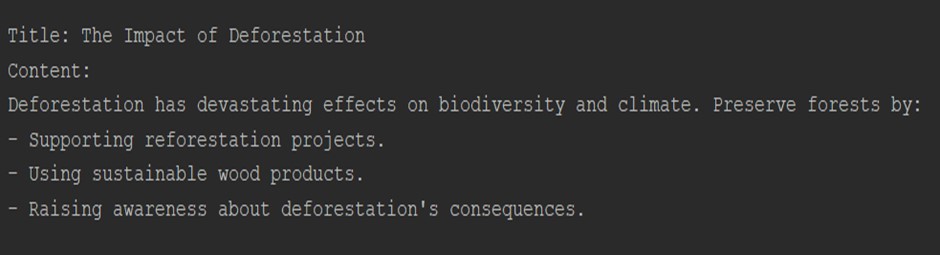
Security: Encapsulation reduces vulnerabilities and stops unwanted external code intervention by restricting access to the object's internal state.

**Some Blog Outputs:**

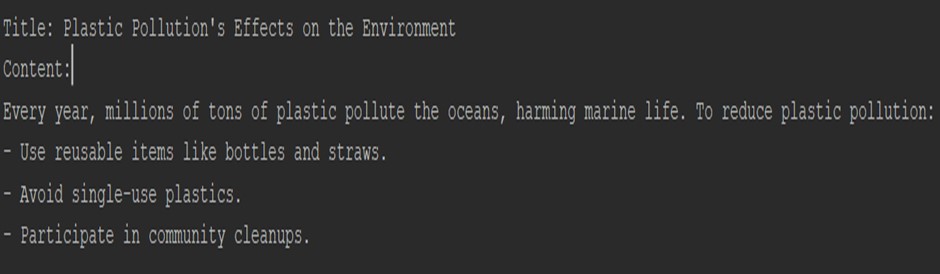
**Blog 1:**



**Blog 2:**



**Blog 3:**



**Features:**

**Blog Management**

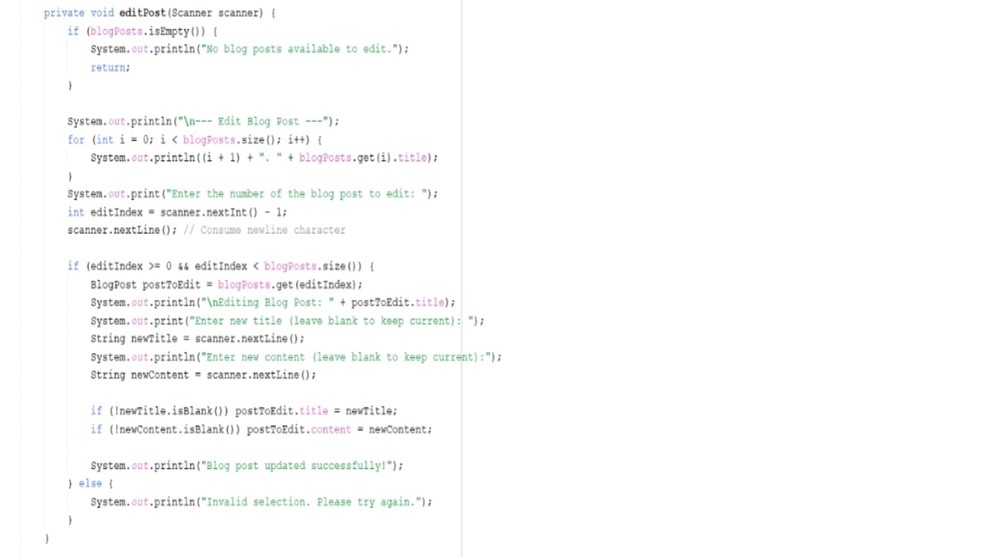
1. Blog posts can be created, edited, deleted, and searched by admin users.
2. The blogs are accessible to the general public, who can also leave comments.

**Admin Functionality Snippets**

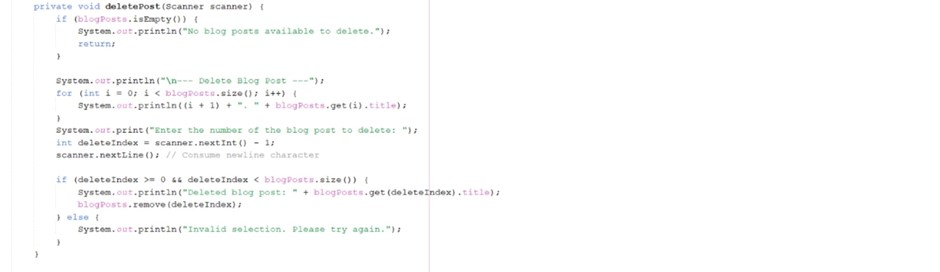
1. **Create Blog Post**



1. **Edit Blog Post**



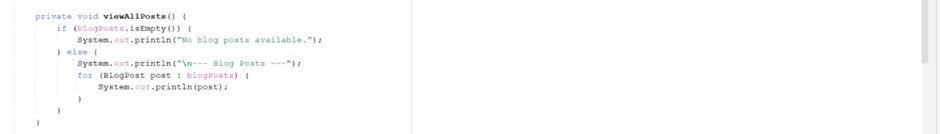
1. **Delete Blog Post**



1. **Search Blog Posts**



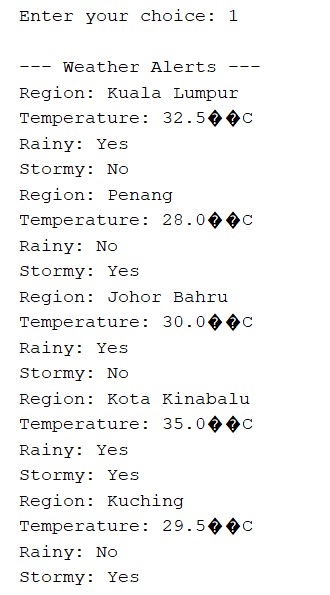
**Public Functionality Snippets**



## **● 4.7 Feature 7: Real-time Alert Warning System**

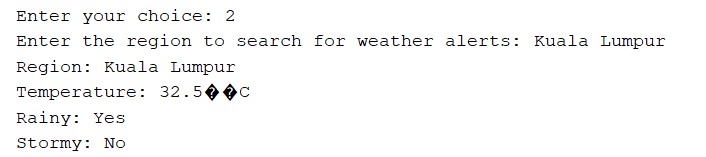
When you select the Real-time Alert warning system from the public user main menu, you will be redirected to another menu where users can query the temperature and rainfall and wind conditions of all regions that have been entered or query by region

**View all weather alerts:**Query the temperature of all entered regions and whether there is rain or storm

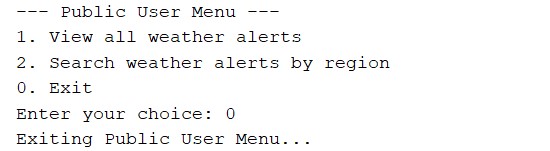


***Figure: view all weather alerts***

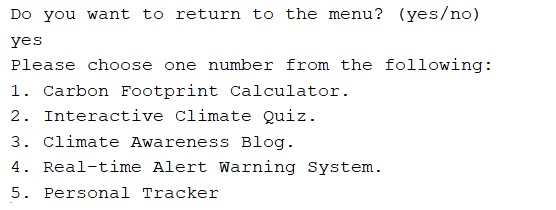
**Search weather alerts by region:**Search by entering the region the user wants to query



***Figure:Search weather alerts by region***



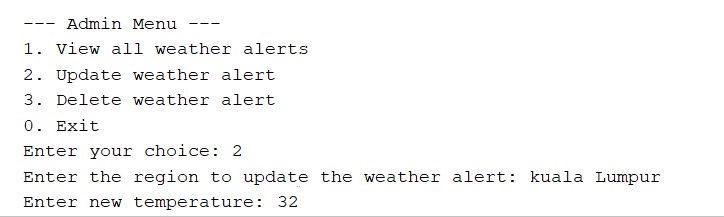
***Figure: Exit***



***Figure: Return to the menu***

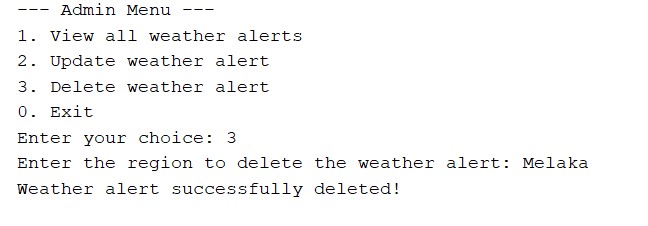
When you select "Real-time Alert Warnings" in the Admin main menu, you will be redirected to another menu where you can modify the temperature, rainfall and wind conditions for all the regions entered, as well as add and delete

**Update weather alert:**Enter the name of the region you want to update, the temperature, and whether it is raining or stormy.



***Figure: Update weather alert***

**Delete weather alert**：Delete the region you want to delete



***Figure: Delete weather alert***

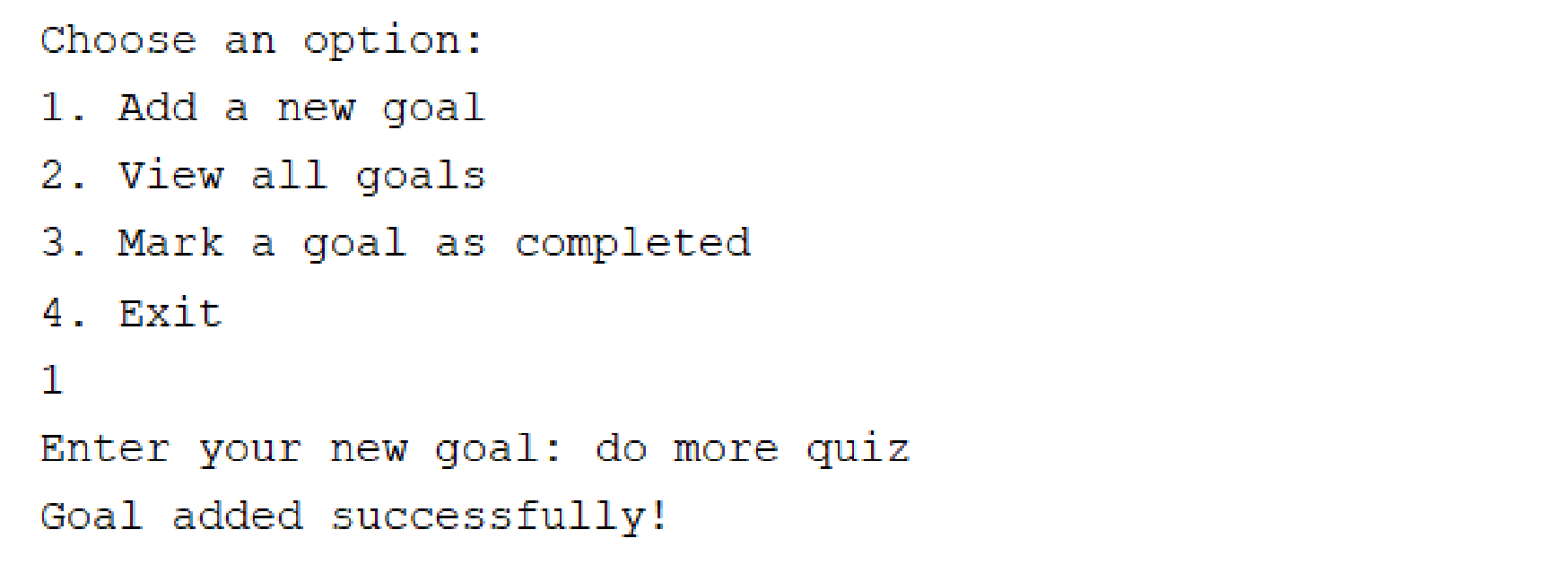
## **● 4.8 Feature 8: Personal Tracker**

When you select the Personal Tracker option from the public user menu, you will be prompted with several options to manage your goals.



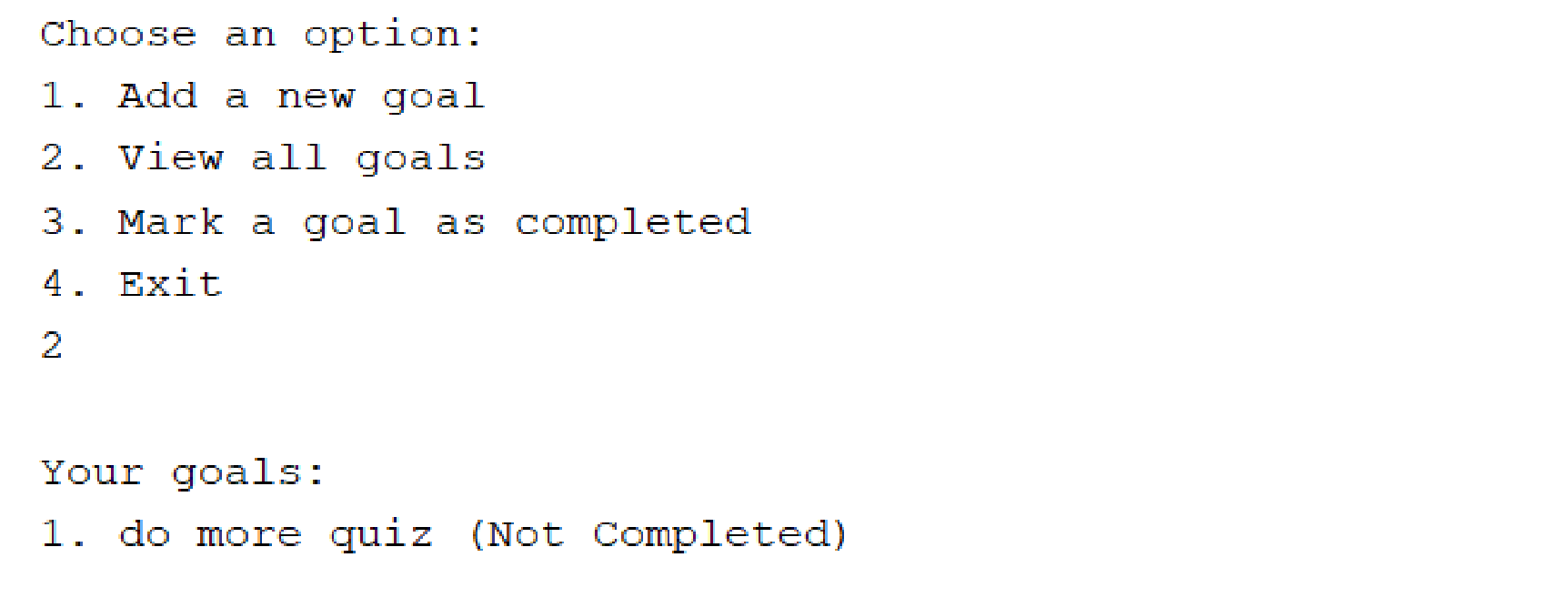
***Figure: Personal Tracker menu***

**Add a New Goal**: Enter a description for your new goal.



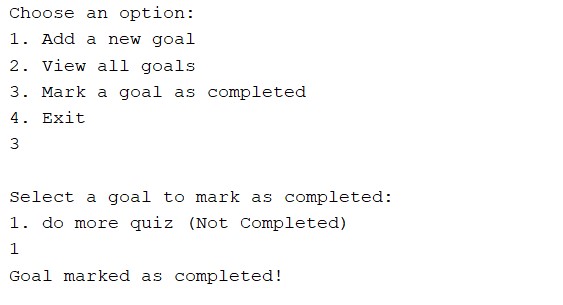
***Figure: Add a new goal***

**View All Goals**: Display all your goals along with their completion status.



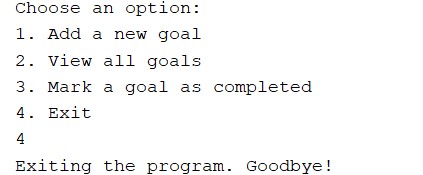
***Figure: View all goals***

**Mark a Goal as Completed**: Select a goal from the list to mark it as completed.



***Figure: Mark a goal as completed***

**Exit**: Exit the Personal Tracker menu and return to the previous menu.



***Figure: Exit program***

# 5.0 LESSON LEARNED

❖ 4.1.1 User Identification

Through this assignment, we have gained a clearer understanding of Sustainable Development Goal 13 (SDG 13), which focuses on climate action. We have come to realize the importance of raising awareness about climate change. Climate issues are everybody’s issues with all people, especially the institutions, having a responsibility. They are crucial in teaching and encouraging the youth to be active and responsive in the fight against climate change. Hence, there is a need to incorporate climate change education in school and university education. This will not only provide them with the relevant knowledge but will also enable them to contribute to making changes in the environment.

Besides, designing this program has been a tough yet rewarding learning experience. Although some of our team members were familiar with the relevant assignments, others were unfamiliar with Java coding, which caused initial challenges in both program writing and understanding the design requirements. We faced difficulties like figuring out which features were unnecessary and hard to link with other program components. Through patience and teamwork, we found the necessity of reducing our approach to match the task's criteria while maintaining functionality. In the end, we came up with a program design that exactly fits our goals and utilizes the simplest techniques to achieve the same level of efficacy as more complicated solutions.

In conclusion, we conducted an in-depth study on SDG 13 as well as developed our coding skills in Java. We hope that our efforts on this assignment will play a meaningful role in increasing public awareness of climate change.

**REFERENCES**

1. United Nations Development Programme (UNDP). (n.d.). *Sustainable Development Goals (SDG 13: Climate Action)*. Retrieved from https://www.undp.org/sustainable-development-goals
2. NASA Earth Science Division. (n.d.). *Global Climate Change: Vital Signs of the Planet*. Retrieved from https://climate.nasa.gov
3. IPCC - Intergovernmental Panel on Climate Change. (2023). *Climate Change 2023: The Physical Science Basis*. Retrieved from https://www.ipcc.ch
4. International Energy Agency (IEA). (2022). *World Energy Outlook 2022*. Retrieved from https://www.iea.org/reports/world-energy-outlook-2022
5. Java Documentation. (n.d.). *Java Platform, Standard Edition Documentation*. Oracle Corporation. Retrieved from https://docs.oracle.com/javase/8/docs/
6. GeeksforGeeks. (n.d.). *Understanding Encapsulation in Object-Oriented Programming*. Retrieved from https://www.geeksforgeeks.org/encapsulation-in-java/
7. Statista. (2023). *Global Greenhouse Gas Emissions by Sector 2022*. Retrieved from https://www.statista.com/statistics/1065755/global-greenhouse-gas-emi ssions-sector/
8. International Renewable Energy Agency (IRENA). (2021). *Renewables: Key to a Sustainable Future*. Retrieved from https://www.irena.org/
9. Kumar, S., & Bansal, M. (2019). *Principles of Object-Oriented Programming Using Java*. Springer Publications.

10.Environmental Protection Agency (EPA). (2022). *Understanding Climate Change Impacts*. Retrieved from https://www.epa.gov/climate-change