PFS Assignment 3

Task 1:

1. Develop a fully object-oriented solution by providing source code of implementation for the algorithm written in mini project 2.

To develop a fully object-oriented solution for NSJ (North Sussex Judo) application, we can use these object-oriented concepts such as encapsulation, inheritance, polymorphism, abstraction, and interfaces. For example, these are how we can implement the NSJ application using these concepts:

* **Encapsulation**

We can encapsulate the properties and behaviors of various entities within the classes. So that, we can ensure that the data is kept private and can be accessed only through well-defined interfaces or classes.

To explain more, encapsulation is the process of combining code and data into a single unit. So, in the context of North Sussex Judo Java Program, the Athlete class highlights encapsulation by encapsulating its internal data and giving restricted access via getters. This assures that the class's private instance variables, including name, trainingPlanDetails, currentWeight, competitionWeightCategory, competitionsEnter, and privateCoachingHours, are not directly available from other sources.

For example, let’s look to our Athlete Class.





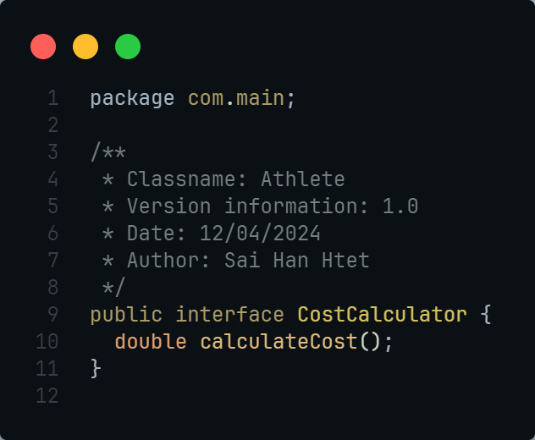
* **Inheritance**: Also, we can use inheritance to construct specialized classes that inherit the attributes and behaviors from other classes. Moreover, not only the child class has the ability to override the parent class methods, but also it can provide the additional functionality to the parent class.

So, in the context of North Sussex Judo Program, we can start by creating a base class for athletes and then construct specialized classes for other categories of athletes, such as competitive athletes.





* **Polymorphism**: Plus, polymorphism can help us by allowing objects to be considered like instances of their parent class. So, this allows us to develop code that can interact with objects from many classes via a common interface.
* **Abstraction**: Moreover, abstraction can also enable us to focus on an object's core qualities and behaviors while obscuring superfluous information. With this, abstract classes and interfaces can be used to describe common behaviors that are implemented by different classes.
* **Interfaces**: Lastly, interfaces can create a contract for classes that implement them, stating which methods they must provide. And, this enables to the many classes to implement the same interface while providing distinct implementations for its methods.



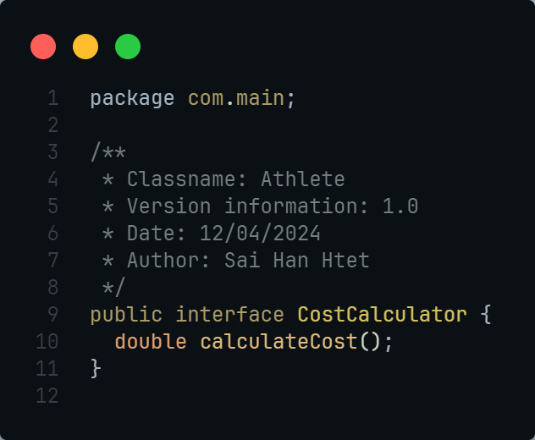
This is the interface that I implemented in the Athlete class.

* North Sussex Judo Project Structure:



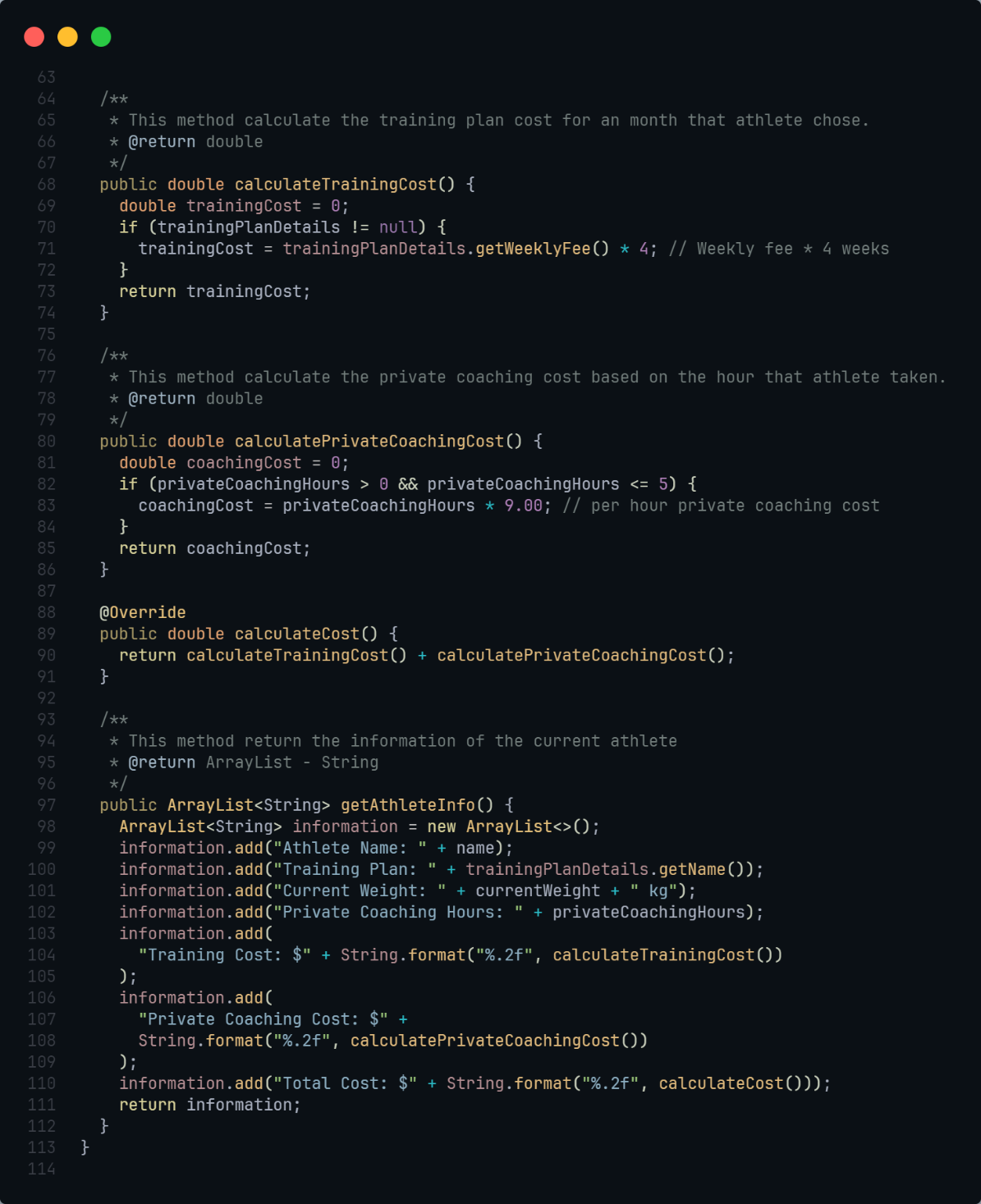
Source code:

CostCalculator.java

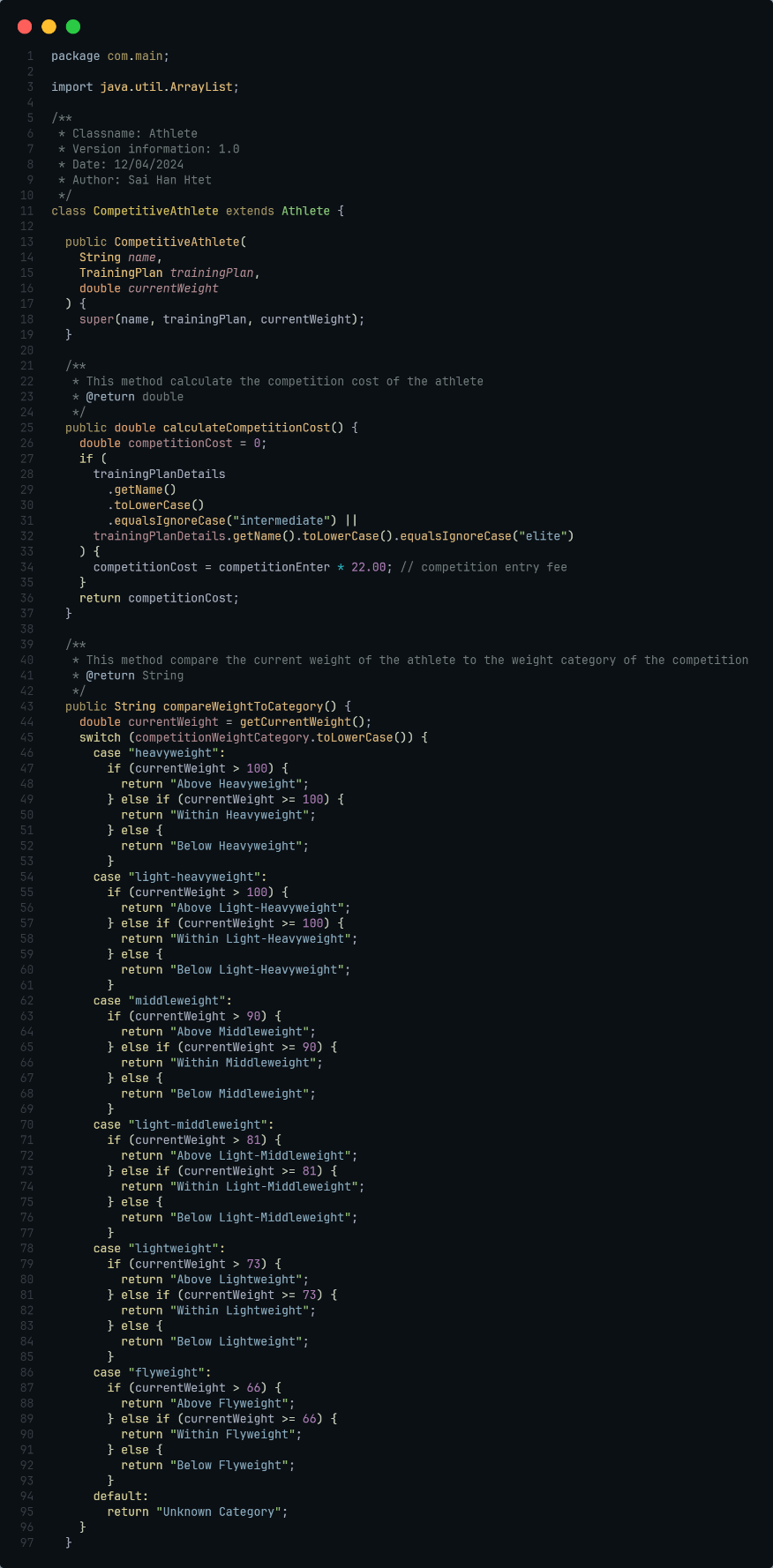


Athlete.java





CompetitiveAthlete.java



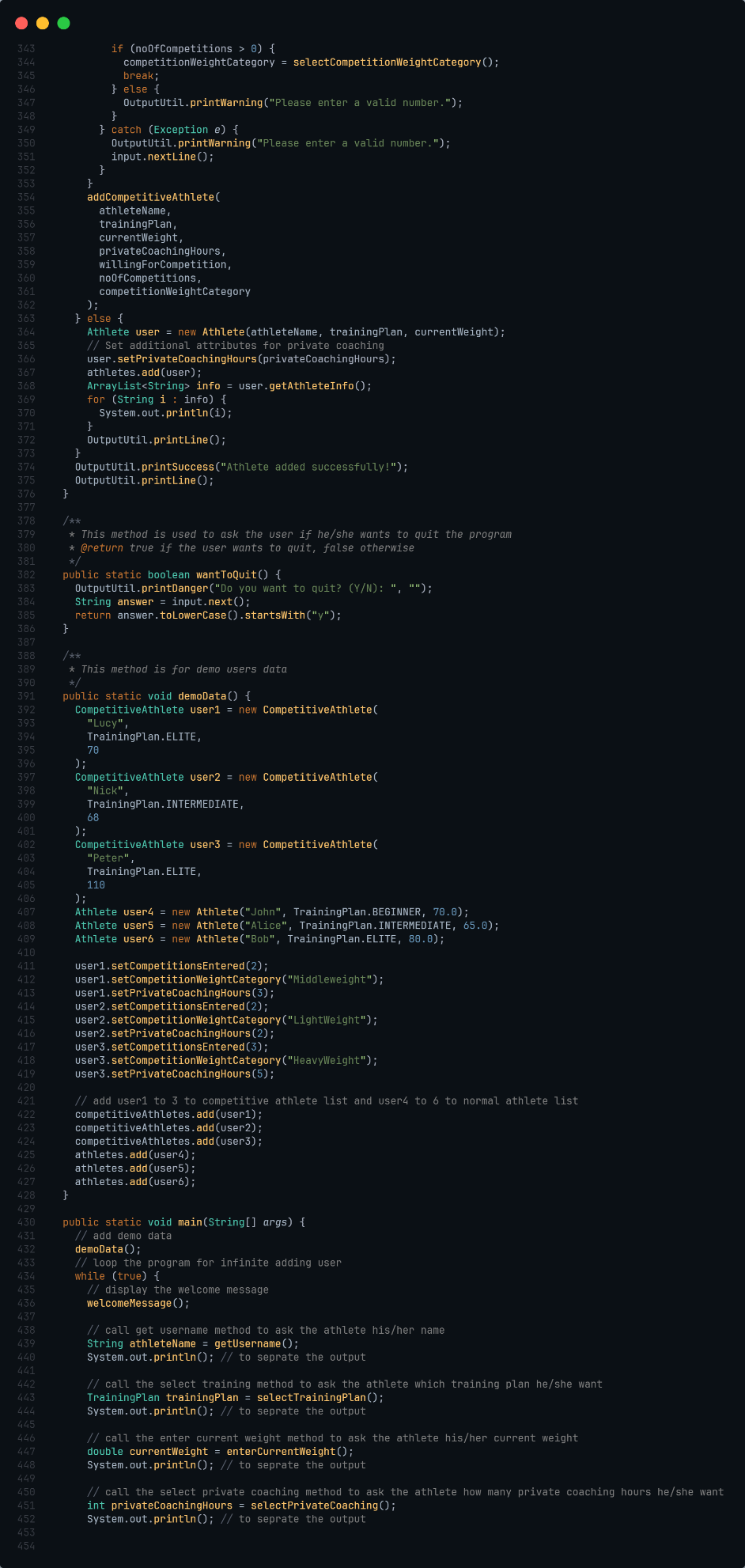


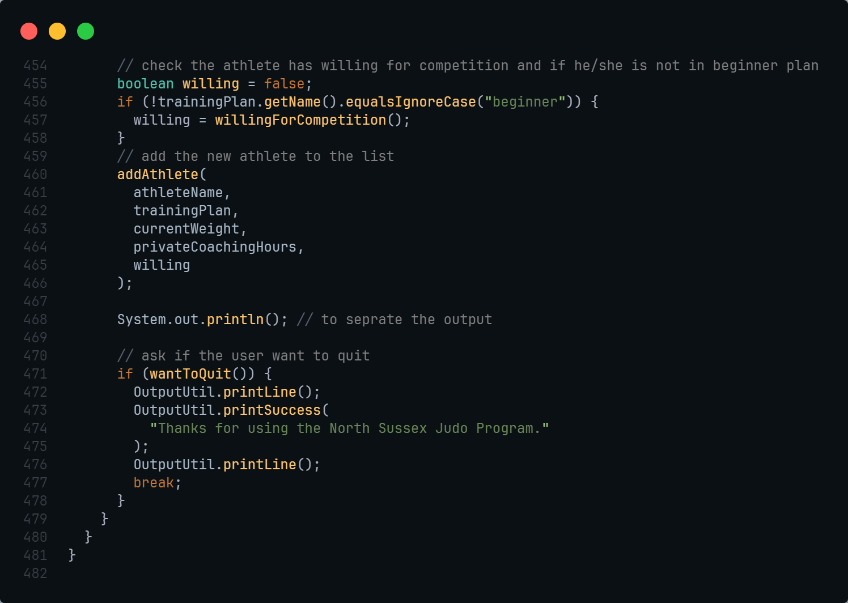
TrainingPlan.java



Main.java

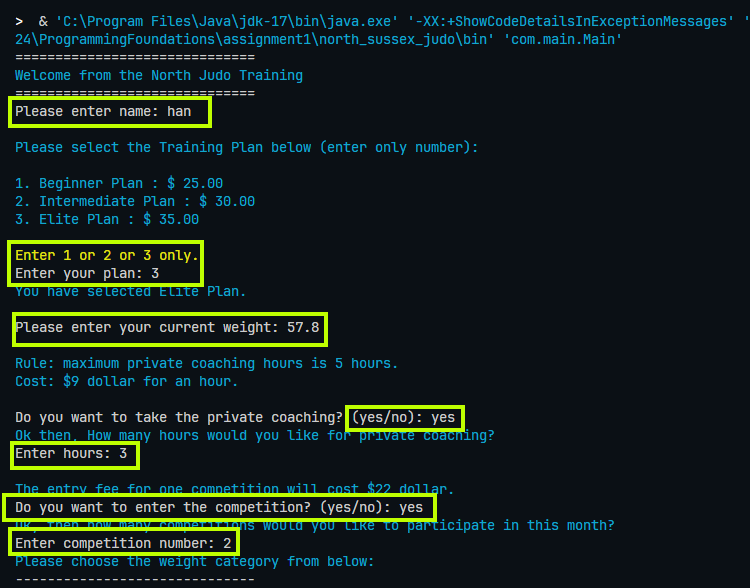


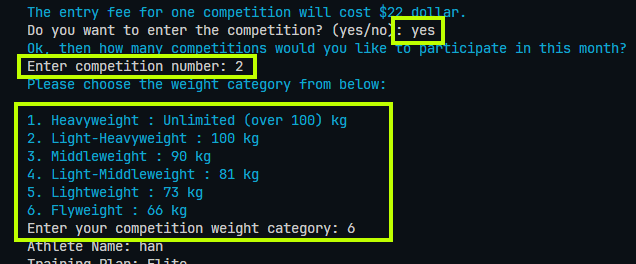




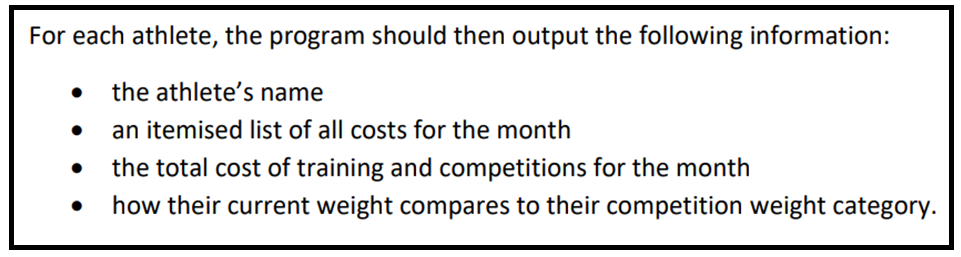
OutputUtil.java ColorIO.java

Output for Requirement 1





Requirement 2



Output of Requirement 2



Source Code for Requirement 2

These are the source code for requirement 2.

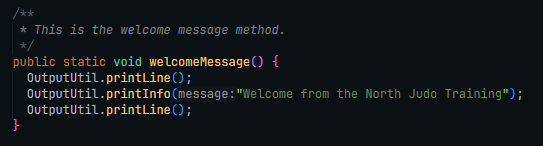
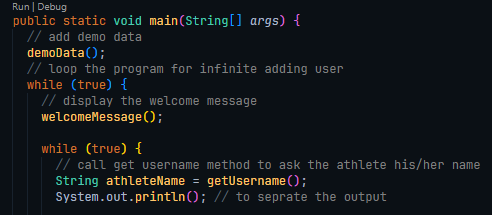
Source code online:

<https://github.com/saihanhtet/SchoolProjectLithan/tree/main/ECMM_HNDC_PFS_0324/ProgrammingFoundations/assignment1/north_sussex_judo>

1. Provide a brief note on how the algorithm is translated to equivalent code.

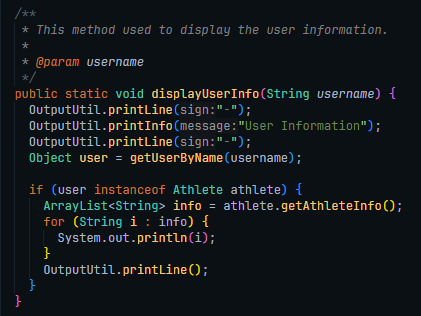
To convert the algorithm into code for the North Sussex Judo application, I used an organized approach that ensure the clarity, correctness, and efficiency in execution. So, to achieve that, I wrote the code along the side of algorithm.

Algorithm part 1

1. Display the welcome message.
2. Display the prompt to user to enter the athlete’s name.
3. Check if the athlete’s name is valid or not.
4. If not, display appropriate message and go back to the step 2. If valid, proceed to step 5.

This code display the welcome message with the welcomeMessage() method. (Algorithm step 1) After that, the program will ask about the athlete’s name because we called the getUsername() method in program and then the athlete’s name that user has been input will be checked based on the regex to see the user input is valid or not. After that, if the user input is valid then we will check the athlete’s name is already registered or not by using isUserRegistered() method and if user is register we will show the information of user with displayUserInfo() method. If user is newcomer, we will continue or program. (Algorithm step 2 to 4)

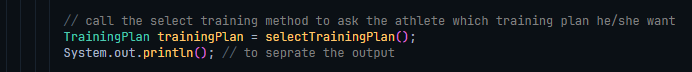
Notice: displayUserInfo method screenshot in below.



This method displays the user information line by line.

1. Display the training plan of judo that includes 3 levels with fees. (E.g., Beginner - $25.00, Intermediate - $30.00, Elite - $35.00)
2. Prompt the input for user to enter the training plan that they would like to choose.
3. Check the training plan is valid or not.
4. If not valid, display the appropriate message and go back to **Step 6**. If valid, proceed to **Step 9**.
5. Display the prompt to user to enter their current weight in Kg unit.
6. Check if the weight is valid or not.
7. If valid proceed to **Step 12**, if not display the appropriate message and go back to **Step 9**.
8. Display the private coaching rules and fee.
9. Prompt the user to user to take the private coaching or not.
10. If user take the private coaching, proceed to Step 15. If not proceed to Step 17
11. Prompt the user to enter the number of hours for private coaching.
12. Check if the input valid or not. If valid go to step 17. If not, go back to Step 15.

(Algorithm Step 5 to Step 8)



This code calls the selectTrainingPlan() method to able the user to select the training plan.

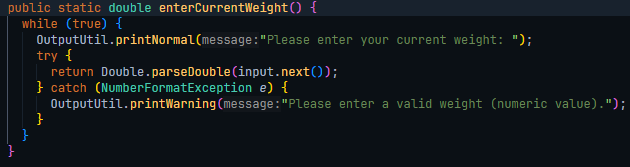
This is the selectTrainigPlan() method. Just like on algorithm step 5 to 8, it displays the training plans with its fee. After that, it checks the user input is valid or not with the try catch statement. And choose the training plan based on the user input such as 1,2, or 3.

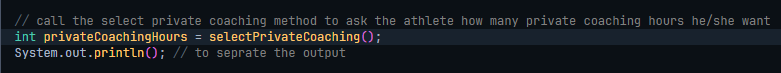
**Notice: OutputUtils is the class defined by myself. It exactly acts like System.out.println()**

**Source:** [**https://github.com/saihanhtet/SchoolProjectLithan/blob/main/ECMM\_HNDC\_PFS\_0324/ProgrammingFoundations/assignment1/north\_sussex\_judo/src/com/main/OutputUtil.java**](https://github.com/saihanhtet/SchoolProjectLithan/blob/main/ECMM_HNDC_PFS_0324/ProgrammingFoundations/assignment1/north_sussex_judo/src/com/main/OutputUtil.java)

And, these two following code screenshots are the asking for the user weight input and checking weather it is valid or not. (Algorithm step 8 to step 11)





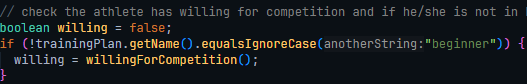


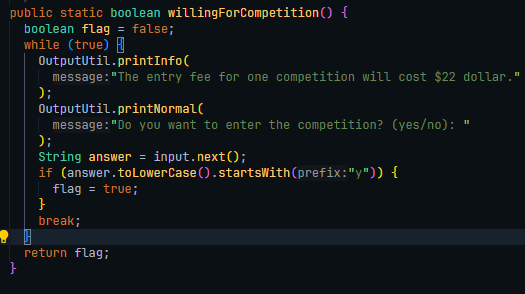


And, these two-code screenshots above display the private coaching hours rules and price and ask the user to enter the if they would like to take the private coaching or not. And the code displays the appropriate messages based on the user input. **(Algorithm Step 12 to Step 16)**

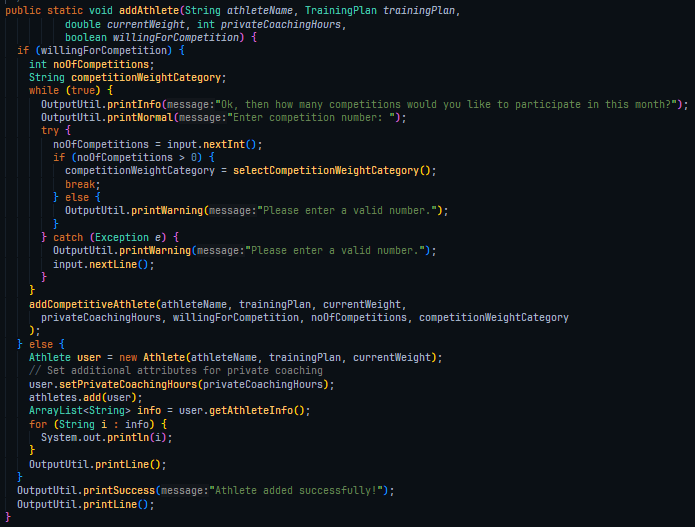
1. The program will check the current athlete plan is eligible for competition or not.
2. If not eligible, the program will proceed to Step 24. If eligible, it will proceed to Step 19.
3. Display the fees of the competition and prompt the user to enter the competition or not.
4. If user entered competition, the program proceeds to Step 21. If not go to Step 24.
5. Display Competitive Weight Categories based on Kg.
6. Prompt the user to input their competitive weight categories.
7. If input is valid, proceed to Step 24. If not go back to Step 21.
8. Calculate the training costs for the user's preferred training plan (Beginner, Intermediate, Elite, private tuition, competition registration fee).
9. Add together all of the calculated training expenditures to determine the athlete's total monthly cost.
10. Based on their present weight, determine how much they need to obtain their ideal competition weight.

These two following codes check the athlete is eligible for competition or not based on their training plan. If the plan is not beginner plan, they are eligible for competition. Plus, the program will ask the eligible users that they want to enter the competition sure or not.





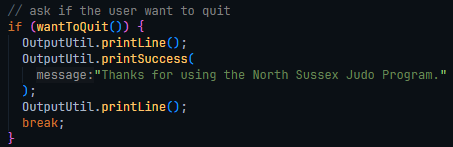
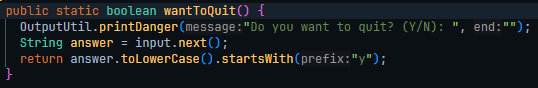
Moreover, after that, the program will be created the new after it done asking the number of competition that the athlete want to participate and after user has been selected the competition weight category. The athlete that they want to enter the competition will be stored in competitionAthletes ArrayList and those who not want to enter the competition will be stored in athletes ArraryList. After they are added, the program will display their information such as athlete’s name, training plan, current weight, and so on. (Algorithm **Step 17 to Step26)**



Step 27: Prompt the user to quit or not.

Step 28: If user input “yes”, the program will display farewell and exit. If not, the program will proceed to **Step 2.**

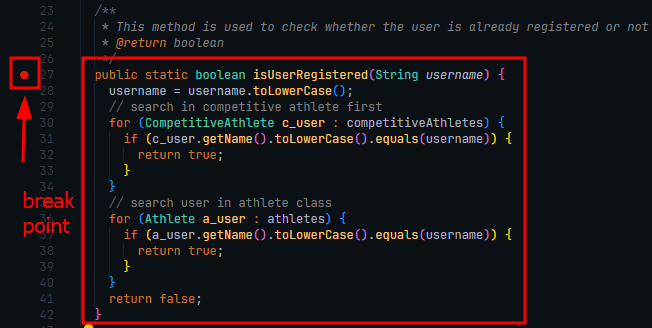
Task 2:

1. An explanation of the debugging features available in your chosen IDE.

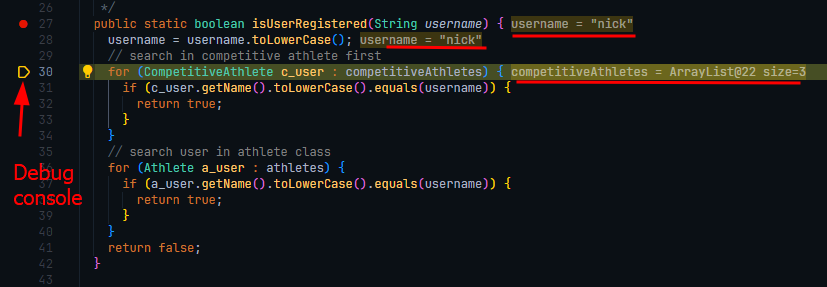
Debugging is a very important component of software development that involves detecting and resolving mistakes, or bugs, inside a program. Because these flaws might create the unexpected behavior or functionality problems in the software. So, one efficient technique to debug code is to use the debugging features provided by Integrated Development Environments (IDEs) such as Visual Studio Code.

Here's an overview of the debugging features in Visual Studio Code:

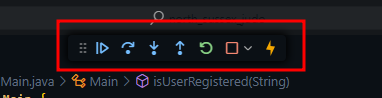
1. **Breakpoints**: Breakpoints are code markers that cause the program to pause when they are reached. This enables developers to examine the program's state, variable values, and execution flow at the specified moment. VS Code supports a variety of breakpoints, including line, conditional, and function breakpoints. For example, in below screen capture, the break points are the red points along the side of line numbers and they can enable us to examine the whole isUserRegistered method entirely.



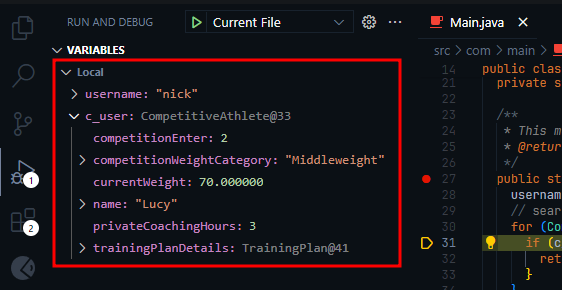
1. **Debug Console**: The debug console is an interactive environment where the developers can execute the code snippets, evaluate expressions, and print out some debug messages while debugging. And, it provides valuable insights into the program's behavior and helps in comprehending the runtime data. This is the debug console shown during the debugging, when user input the name as “nick” as in testing case.



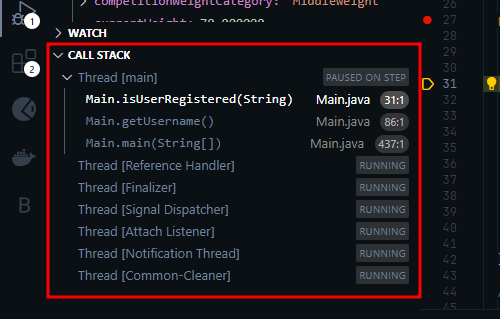
1. **Step-by-Step Execution:** With Visual Studio Code, many developers can now execute the code step by step, because step-by-step execution can allow us to monitor the program's behavior at every point. So, developers can step into functions to investigate their implementation, step over function calls, and navigate the code line by line. For example, Visual Studio Code allow the developers to use these tools.



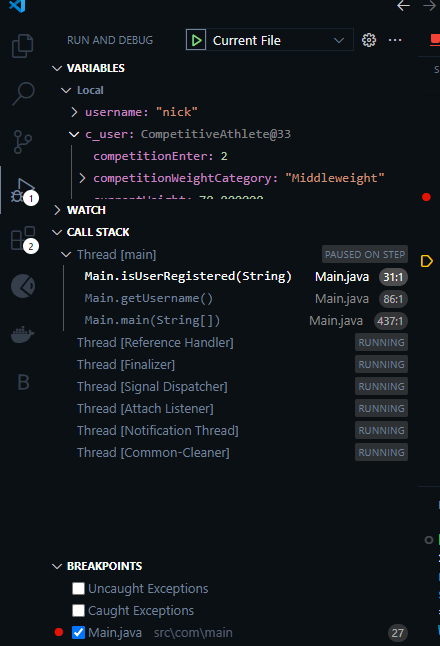
1. Watch Expressions: Plus, watch ­expressions can enable us (developers) to watch the value of variables or expressions in real time as the application runs. So, by adding watch expressions, it can help the developers to follow certain data points and rapidly spot any unusual changes or abnormalities.



1. **Call Stack:** In the debug panel's call stack view, it displays the function call to hierarchy, which means it includes the sequence of function calls that led to the present execution point. So, this functionality enables us to trace the code's execution route and understand the flow of program control.



1. Run and Debug View: And in this view, it acts like the central location for the management of debugging sessions. Plus, it provides the options to specify the parameters, set environmental variables, and configure other debugging-related settings. So, this view can simplify the debugging process and increases developer productivity.



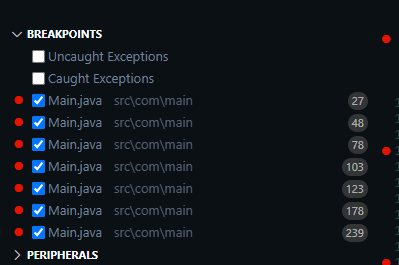
So, to summarize, using the debugging features in VS code allows us to quickly and efficiently identify and address the errors in our code. Not only that, the breakpoint, debug console, call stack, step-by-step execution, watch expressions, and run and debug view helps us to obtain the insights into our North Judo Sussex program’s execution and program flow. So, in other word, by using the debugging tools and features that are offered by the IDE, we can surely increase our code efficiency and effectiveness.

Other Debugging features: <https://code.visualstudio.com/docs/editor/debugging>

1. Explain briefly the steps involved in debugging the developed application, along with screen captures.

These following are the step by step debugging of North Sussex Judo Java Program.

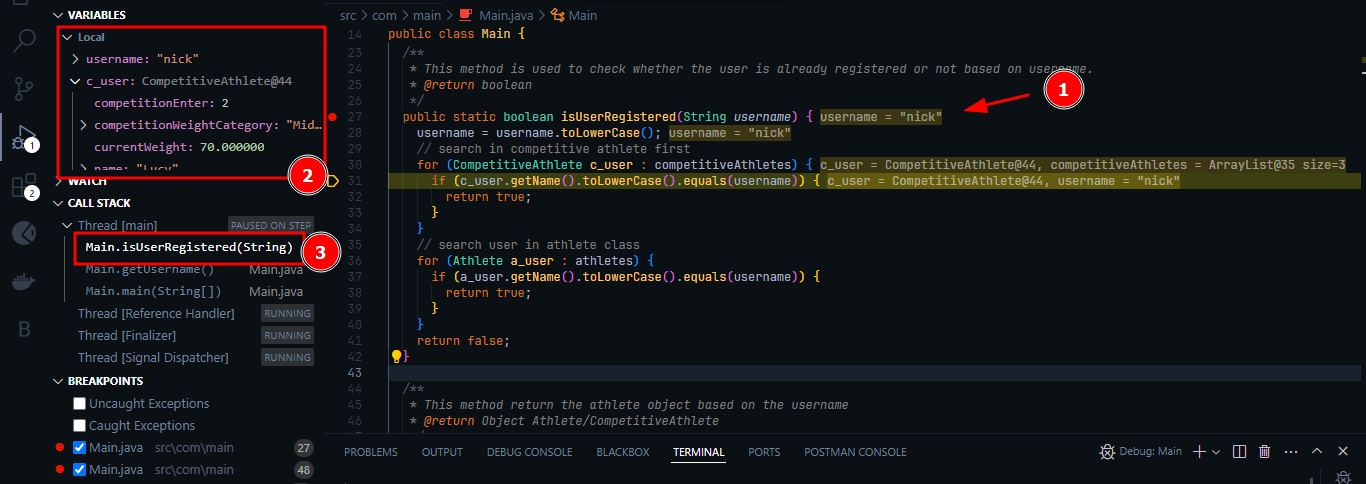
**Step 1. Setting the break points**

I added the break points in each method which are necessary for North Sussex Judo functionality. These are the break points I added in the Main.java file of the North Sussex Judo Program.

As you can see, the break points are on line number 27,48,78,103,123,178, and 239 so on.

**Step 2. Start Debugging**

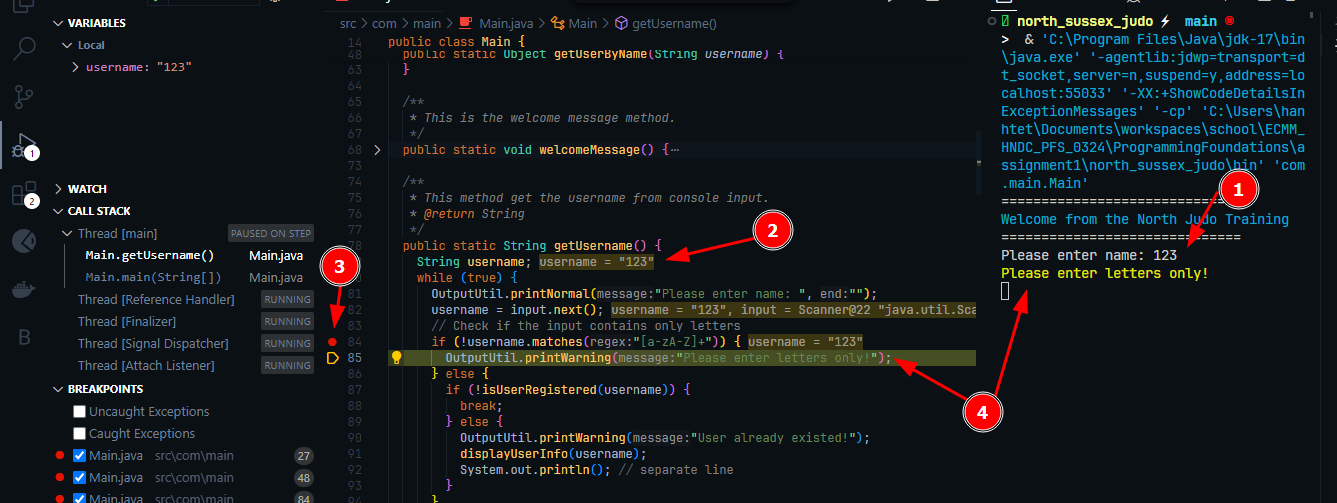
So, on the line number 27, we can see the method that check the user is already register or not. So, we can see the debugging result like this below after the user added his/her name in the input box.



In above screen capture, we can see that user entered his name as “nick” in input and the program passed the “nick” as a parameter of isUserRegistered method (1). After that, the program converts the parameter username to the lowercase and matches with each athletes’ users in competitive athletes or normal athletes from list. Lastly, it will return the true value if the user is found in the program.

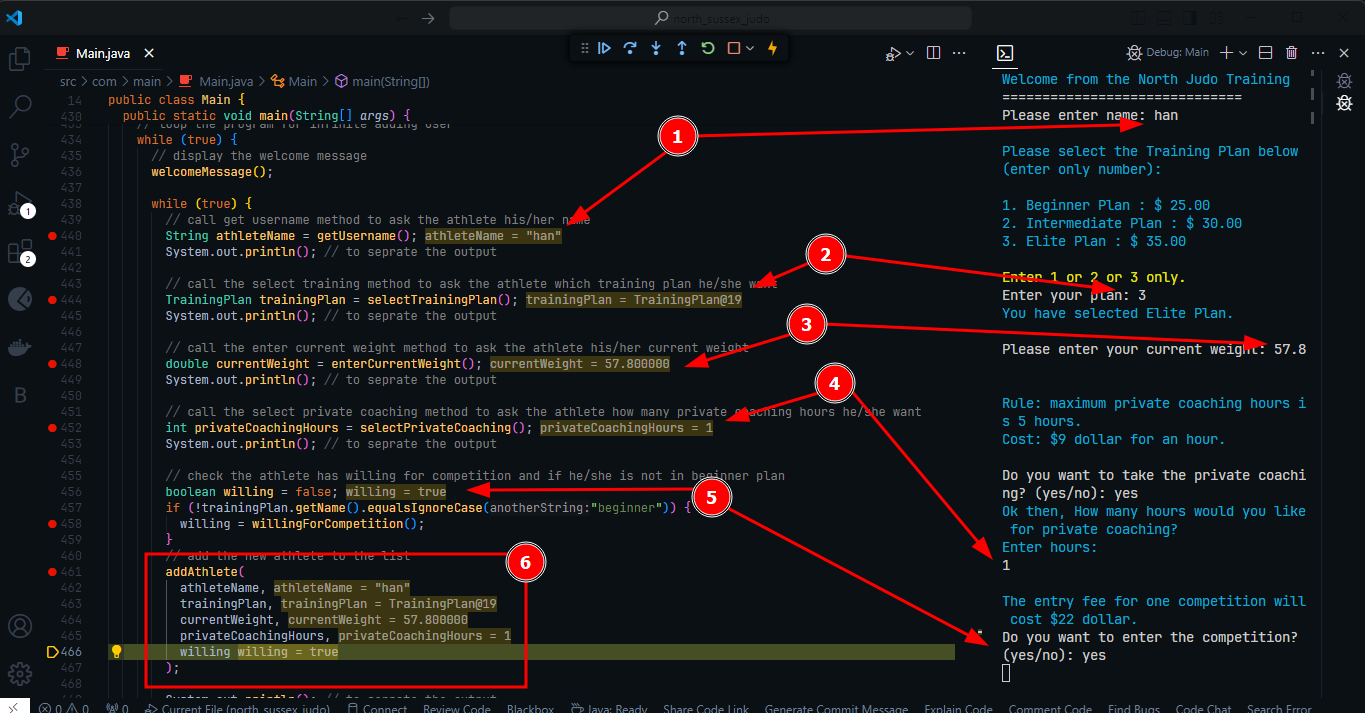
Most importantly, we can see the real time data in left panel upper side (2) and the current call stack in left panel of the screen capture (3).

**Step 3. Check the validation with Debugging.**



As you can see on the above picture, the user added the numeric data in the name request box (1), and the program carries on the numeric data and start check the validation that in line 84, which I put break point at there. And, we can see that the username “123” has been matches by using the regex which is to make sure the user input is only the alphabets only (3). And then the program displays the appropriate warning to user (4).

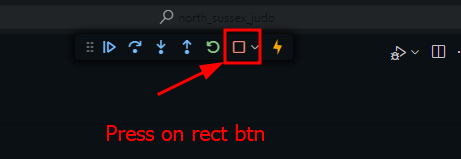
**Step 4. Steps through whole program.**

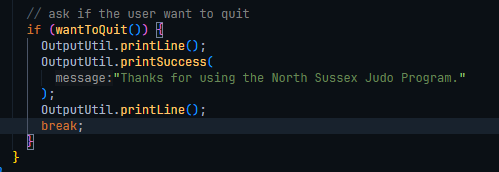


In this screen capture, we can see the entire program has been debugged by using the breakpoints, watch expression, debug console, step-by-step execution, so in each step from number 1 to 6.

**Step 5. End the debugging process.**

We can simply end the debugging process by pressing the rectangle btn on the tools.



And then, our java application will be stopped. But, we can also stop the program also by answering this.

