PFS Assignment 2

1. Document the architecture of the System along with Module and its components description.

System Architecture:

For the North Sussex Judo (NSJ) Java program, we can define many various objects to represent the various entities such as athletes, training plans, and weight categories. Also, the system architecture for the North Sussex Judo (NSJ) training cost calculator program involves of two main classes: Athlete and CompetitiveAthlete which classes can be represented for athletes and competitive athletes.

**Athlete Class:**

Fields:

* name (String): Represents the name of the athlete.
* trainingPlan (String): Represents the training plan of the athlete.
* currentWeight (double): Represents the current weight of the athlete.
* privateCoachingHours (int): Represents the number of private coaching hours.

Methods:

* calculateTrainingCost(): Calculates the training cost based on the athlete's training plan.
* calculatePrivateCoachingCost(): Calculates the cost of private coaching based on the number of hours.
* calculateTotalCost(): Calculates and return the total cost of training and private coaching.
* getAthleteInfo(): Retrieves information about the athlete, including their name, training plan, current weight, private coaching hours, training cost, private coaching cost, and total cost.

**CompetitiveAthlete Class:**

Fields:

* It inherits all fields from the Athlete class.

Methods:

* calculateCompetitionCost(): Calculates the cost of competitions entered based on the training plan.
* compareWeightToCategory(): Compares the athlete's weight to their competition weight category.
* calculateTotalCost(): Overrides the method in the Athlete class to include competition costs in the total cost calculation.
* getAthleteInfo(): Overrides the method in the Athlete class to include information about competitions entered and competition cost.

**NorthJudoTrainingCostCalculator Class:**

Main Method:

* + Creates a CompetitiveAthlete object representing an athlete.
  + Sets values for competitions entered, competition weight category, and private coaching hours.
  + Retrieves athlete information and displays it using a for loop.

With this architecture, we can get the many advantages because it offers flexibility for future improvements and modifications, while also allowing the management of athlete data and the computation of training, private coaching, and competition cost.

1. Define an algorithm and outline the process in building an application.

**Algorithm for the North Sussex Judo**

Start

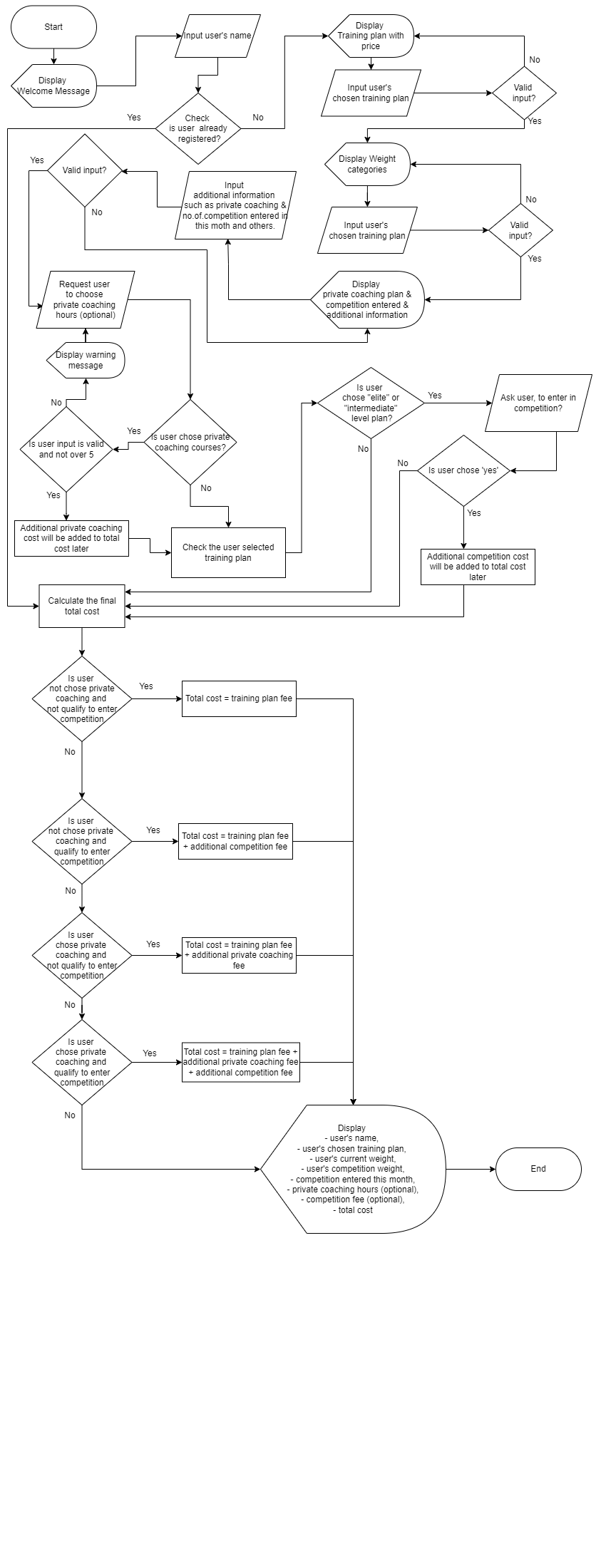
1. Display the welcome message.
2. Display the prompt to user to enter the athlete’s name.
3. Check if the athlete’s name is valid or not.
4. If not, display appropriate message and go back to the step 2. If valid, proceed to step 5.
5. Display the training plan of judo that includes 3 levels with fees. (E.g., Beginner - $25.00, Intermediate - $30.00, Elite - $35.00)
6. Prompt the input for user to enter the training plan that they would like to choose.
7. Check the training plan is valid or not.
8. If not valid, display the appropriate message and go back to **Step 6**. If valid, proceed to **Step 9**.
9. Display the prompt to user to enter their current weight in Kg unit.
10. Check if the weight is valid or not.
11. If valid proceed to **Step 12**, if not display the appropriate message and go back to **Step 9**.
12. Display the private coaching rules and fee.
13. Prompt the user to user to take the private coaching or not.
14. If user take the private coaching, proceed to Step 15. If not proceed to Step 17
15. Prompt the user to enter the number of hours for private coaching.
16. Check if the input valid or not. If valid go to step 17. If not, go back to Step 15.
17. The program will check the current athlete plan is eligible for competition or not.
18. If not eligible, the program will proceed to Step 24. If eligible, it will proceed to Step 19.
19. Display the fees of the competition and prompt the user to enter the competition or not.
20. If user entered competition, the program proceeds to Step 21. If not go to Step 24.
21. Display Competitive Weight Categories based on Kg.
22. Prompt the user to input their competitive weight categories.
23. If input is valid, proceed to Step 24. If not go back to Step 21.
24. Calculate the training costs for the user's preferred training plan (Beginner, Intermediate, Elite, private tuition, competition registration fee).
25. Add together all of the calculated training expenditures to determine the athlete's total monthly cost.
26. Based on their present weight, determine how much they need to obtain their ideal competition weight.
27. Prompt the user to quit or not.
28. If user input “yes”, the program will display farewell and exit. If not, the program will proceed to **Step 2.**

End

**Flowchart for the North Sussex Judo**

**View in browser:**

[**https://drive.google.com/file/d/1k2FnigFESifBkUUuLGEiUnV-DPlqLz6f/view?usp=sharing**](https://drive.google.com/file/d/1k2FnigFESifBkUUuLGEiUnV-DPlqLz6f/view?usp=sharing)



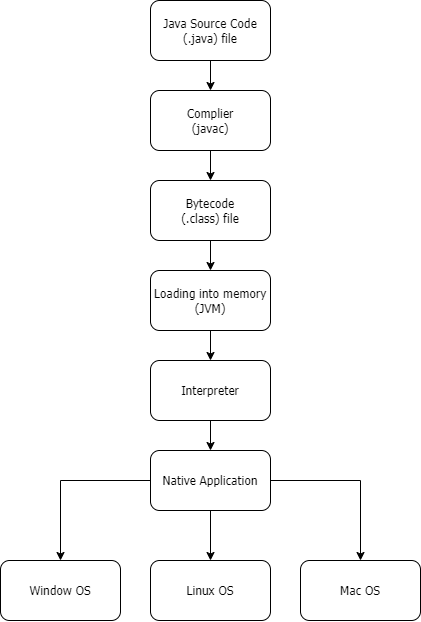
1. Determine the steps taken from writing code to execution

Creating and running Java applications is a systematic procedure that includes writing, compiling, and executing code.

1. Writing a Program: The development process start with the writing source codes, creating classes, methods, variables, and authoring using text editors such as IDEs to create a java application. Most developers use the IDEs to make the coding easier and for enhanced coding experience, syntax highlighting and debugging tools.
2. Compiling: After writing the source code, the source code files are compiled into bytecode by using the java virtual machine (JVM). And it is done by using Java complier (javac) which translates the human readable .java files into platform-independent bytecodes files (.class). And in this stage, JVM ensure the code is translated correctly and can be executed on any system with a compatible JVM.
3. Bytecode Verification (JVM): Also, before executing the bytecode, JVM performs the bytecode verification to ensure its integrity and security. And, this verification procedure makes sure the bytecode doesn't break any security restrictions and complies with Java language requirements. So, with this the JVM prevents the any potential security vulnerability and ensure the stability of the program.
4. Execution: Typically, execution begins with the main method declared within a class. And JVM execute the bytecode instructions and carry out the specified tasks defined in the program. After that, program interacts with users, processes data, and generates output in accordance with its logic as it is being executed.
5. Debugging: Plus, during the development and execution stages, debugging tools are crucial for finding and fixing code problems. So, to ensure accuracy and increase the efficiency, developers go through code blocks by using breakpoints, check variable values, and examine program behavior with the debugging tools that are offered in IDEs.

- Java Program architecture

<https://www.geeksforgeeks.org/compilation-execution-java-program/>



View Image:

<https://drive.google.com/file/d/1OcFQhji_Cm-UHfO4Ejw98CKNrg7a3ILJ/view?usp=sharing>

So, in the context of North Sussex Judo application:

Also, creating a solid Java application, such as the one designed for North Sussex Judo, requires a systematic procedure that involves several critical phases. For example, start from writing the initial source code to executing and debugging the program, also each phases play as very important role in ensuring the functionality, security and reliability of the software. These are the several critical phases that is required for North Sussex Judo program:

1. Writing the initial source code for the program

The first step in developing a Java program is to write the source code and this involves setting up classes, methods, and variables to provide the necessary functionality. So, in the context of North Sussex Judo application, we can say that we need to start by writing the java source code which involves creating classes, methods such as athlete class, competitive athlete classes as I mentioned before.

Most of the developers typically use the Integrated Development Environments (IDEs) such as Visual Studio Code to make coding easier, including capabilities like as syntax highlighting, code completion, and version control integration.

1. Compiling the source code (bytecode) and (.class) files

After writing, the Java source code must be converted into a language that the Java Virtual Machine (JVM) can understand. And, the compilation can be achieved only by using the Java Complier (javac). After that, the human-readable Java files which are named (.java) in the extension of the file, are transformed by the compiler into an intermediate representation called bytecode (.class files). Because, with the bytecode, java applications can execute on any kind of computer that has a JVM that is compatible with them.

1. Loading into memory (JVM)

And, once the bytecode has been generated, the next step is to use the JVM to load the resulting bytecode into memory. Also, the JVM is made up of many parts that control how programs run. Class loaders manage the process of loading classes into memory as needed during runtime. Plus, they can be divided into three categories such as Bootstrap Class Loader, Extension Class Loader, and Application Class Loaders. For instance, the compiled class files are loaded into memory by the North Sussex Judo Java software.

1. Bytecode verification (JVM)

Also, to guarantee the security and integrity of the application, the JVM verifies the bytecode before running it. And, this phase is very important and it include comparing the bytecode to pre-defined rules and constraints provided by the Java language standard. For example, it loads all the classes such as Athlete, CompetitiveAthlete, and other methods such as training plan, weight category, and main methods to ensure that they are all meets the pre-defined rules and constraints of Java language standard.

And, the JVM can reduces possible hazards like memory corruption and unauthorized access by validating the bytecode.

1. Execution

And, once the bytecode has been loaded and confirmed, the JVM will start executing the Java application. After that execution begins normally with the main method specified in the Main class, or any other designated entry point. During execution, and next, the software performs its predefined tasks, such as interacting with users, processing data, and providing output. And lastly, the program is executed until it hits a return statement or the end of the main procedure.

1. Debugging

During software execution, errors or unexpected behaviors might occur even with careful development and testing. So, developers can use the debugging tools offered by IDEs like Visual Studio Code to find and fix these kinds of errors. Plus, Plus, developers can walk through code, check variable values, and create breakpoints to interrupt execution for analytical reasons in order to solve problems in a systematic way. So, debugging a Java program continuously can enhances its overall quality and performance by ensuring the program's stability and dependability.

Class Diagram:

<https://drive.google.com/file/d/1Izu1G-HDPg38r1NofcHFqtt1NbU844uA/view?usp=sharing>

# Sign is for protect

- Sign is for private

+ Sign for public

