Blue Minev

228089922

CI505 Functional programming

Reflective report on the development of a software system

# INTRODUCTION

I have developed a student software system in Java using functional programming principles. The system can add and remove students as well as perform a multitude of queries.

# DEVELOPMENT

After looking through the requirements for this project I decided to start by tackling the problem using an object-oriented approach so that I could understand the scope and requirements better. After learning about records and the stream API in lectures I was then able to confidently start this project using functional programming. By using the knowledge I already had to try and tackle the problem I was able to understand what is needed and issues that I may come across better than if I had started using functional programming. For example, before learning about records I was planning on using a list or a dictionary but was encountering a lot of issues since each student would need a lot of information and I would have to be using 3D arrays to get around this issue as I knew that objects would not be allowed. When we were taught about records this solved this issue.

A record is the functional programming equivalent to an object its aim is to remove unnecessary boilerplate code such as getters and setters. Records are an easier to implement data carrier class than objects requiring little to no extra methods to set them up.

A computer screen shot of a program code

Description automatically generated*The original data values in my student record.*

When considering what data, I wanted to have assigned to each student I knew that there must be a unique value for each student to make it easier for searching students with same names or finding details of the student when you only have this value. I chose to use a unique student ID starting from one and incrementing with each additional student. I also decided to split the students name into a first and last name that way if a potential user only knew the first name of the student, they were they would be able to still search for that student. I also decided to include year of study and the course that they were on as data values. I Also included modules and marked for those modules as paired data values for the student.

I decided to use an Enum for the course after beginning with a string value to prevent errors when a user would add a student for example misspelling the course name or capitalization issues. I chose to include a default NULL option for if what the user inputted was not an option.

A screen shot of a computer program

Description automatically generated*The list of Courses I chose to include, which is very easy to add more options if needed.*

I encountered many issues in working out how to implement the marks and modules section for each student. This is since I did not fully understand the map data value. Originally, I fought the map.of() created a single map and you had to have a list of the maps this meant that when I tried to access the values in this list of maps, I struggled a lot. After some more investigation I discovered that is singular map.of() would suffice as you could set create multiple values inside the map. I still then encountered issues when I try to use this stream API to only get the marks out of the map especially since my original aim was to get an average of these marks. Since I was unable to do this using the stream API, I decided to rethink the way I was going to get the value for the grades. Since each student had three modules and therefore 3 marks instead of adding up all the Marks and dividing it by three, I decided to triple the grade percentages and use that instead . For example, instead of 70 being a first I tripled that and therefore the requirement to get a first was 210 marks. This removed the need to do division altogether and I still got the same results as I would have if I divided the values.

A screenshot of a computer program

Description automatically generated

Above shows the function stuByGrades(). This function organises all students by their grade. During testing for this function, I realised that only students with a first and students who are failing were being represented correctly. After some investigation I discovered that I had not set any upper limits for the students with any other grades. This meant that where I wanted to see students with a 2:2, it was also showing all students that had a 2:1 and a first. I fixed this by adding another filter to the stream and the issue was fixed.

A computer screen shot of white text

Description automatically generated*My final version of the student record.*

After creating the base of my student record, I then began to explore the stream API and its abilities in terms of queries for this student record. I began to think about what queries would be most common when using a student record as well as being able to show my knowledge of stream API. I decided on showing all students ordered by last name, being able to find a student by their student ID, being able to find a student with just the 1st letter of their first name, being able to show all students on a course ordered by grade, showing all students grouped by their grade, and showing all students grouped by their year of study. I also created the option to add and remove students.

When removing a student, I decided that the added security you would have to confirm that you wanted to remove that specific student this involved. I decided that the process to remove a student would be for the user to enter the student ID of the student they wanted to delete then the programme would ask if they wanted to remove this student and show the student record for to start student if the user replied yes then the record would be deleted if not the record will not be deleted and I've away a confirmation of what happened would be shown.

A computer screen shot of a program

Description automatically generated

Adding the student, the choirs a lot more code done to delete the student as the user has to input a lot of information. To start I used the scanner tool to get input from the user. I collected the first name and the last name and set them to temporary variables I then collected the year and then used a switch case Convert the string into its integer value if it was the years one through 5 if the user gave any other input it would default to the year being one. I did a similar process with collecting the course name, by fast converting everything to the same case type and then using a switch case and adding module and mark information based on the course provided and gave each student a base mark of 50. I then had all the information to be able to successfully add a student. Once a student is added I success message is printed to the terminal so then the user is aware. When creating the module information, I take the year of study provided earlier by the user and add free to it as that is the typical way that university module names are used for example CI 505 is the second year of university study.

A computer screen shot of a program

Description automatically generatedA screen shot of a computer screen

Description automatically generatedA computer screen shot of a program code

Description automatically generated

When adding the student, I call the GetLastStudentID() function. This function uses the reduce function on a stream to get only the last item in a stream. I then collected the student ID. I had to get the student ID this way instead of just finding the length of the list of students as if a student was deleted, the length plus one would be an already used student ID. A screen shot of a computer program

Description automatically generated

I use the function CourseToEnum() to clean up my code and make it easier to read, this also means that if this code is needed elsewhere, I can call this function, reducing the number of lines of code used.

A screen shot of a computer program

Description automatically generated

The code takes the user input and converts it to lowercase and then uses a switch case to return the Enum equivalent.

A vital aspect of the development of this project was the use of functional programming. The functional programming paradigm prioritises the use of functions instead of objects. This creates code that is easy to maintain as well as easier to understand at first.

One part of functional programming is pure functions. Pure functions are functions that have no effect on any values outside of the function and will always return the same output when given the same input. This aspect is shown in my code by functions like stuByYear().

A screen shot of a computer program

Description automatically generated

Another aspect of functional programming is immutability. Immutability is the concept of not modifying data outside of a function. One way I have achieved this is by using Stream API. Stream API creates a copy of the data and makes changes to that instead of the original data. I used the Stream API for many parts of my code, specifically for queries and searches. Within the stream API I also used predicates. When using Stream Api to filter items, you use a lambda function to specify what values you want in the stream. Predicates are a way of storing a specific lambda expression to lessen the amount of code if it will be needed multiple times in the code. In my project I felt I did not really have a use for predicates as I was not using a specific lambda expression multiple times as I needed multiple values changed however, I chose to include some in Queries.java to show my understanding of them.

A black background with white text

Description automatically generated*An example of a predicate I used in the stuByYear() function.*

In my main Java file, I create an array list and then add 15 different students to it on varying courses with varying years of study and varying marks. This is used for testing and so that my functions can be used. I chose to use a switch case with nine options on the terminal as to be able to focus on the functional programming aspects instead of focusing on UI design. The switch case begins with text output it to the terminal of a list of all options the user can do user then inputs which number option they want the number is then used to go through the switch case and either cause the appropriate function to be called or runs through the code. I chose to use a while loop so that the user can do many different tasks before quitting the program.

A computer screen shot of a program

Description automatically generated

A computer screen shot of a program

Description automatically generated

Case 4 shows all students on a course and sorts them by grade and then provides a count of the number of students on the course. To sort the students by grade, I had to reduce each value to the module marks added together and then compare it to another reduced item in the stream. To do this I had to create a stream of a stream. To create the count, I had to use the .count() feature of a stream.

A screenshot of a computer

Description automatically generated

To end the program, I set cont to false and the while loop ends, finishing the program. Fi the user inputs something that is not one of the predetermined answers it sends them back to the beginning and makes them input it again. A screen shot of a computer program

Description automatically generated

Throughout my development I was testing my code. I ensured to use edge cases as well as incorrect values to test all possible scenarios and compare them to the expected result. By using a terminal UI testing was made a lot easier as I could see all the main options I had. When the result was not the one expected I was able to easily find where in the code the issue was. I made use of print to terminal statements to figure out what went wrong and fixed it. I then repeated the tests to ensure that the code was behaving in the way expected.

# CRITICAL REVIEW

I believe that I have developed a solid understanding of functional programming paradigms and have successfully displayed my knowledge. I created a simple terminal-based user interface so that users would not need ay coding knowledge to use the system. I created multiple queries meaning that the users can collect common information needed quickly.

To improve my system, I would have created an option to edit student records such as module marks. To do this I would have to create a new student record with the old information and update the new information and then delete the old student record as records are immutable and cannot be changed.   
I would also create a function to move all students up a year and add a new Enum value for graduates. This would be time consuming as I would have to delete and create new student records for every student.

Finally, I would have consolidated the CourseToEnum() function so that would not have to have a duplicate code with module marks added on to it. To complete this I would have to do further research to fully understand the scope.

# CONCLUSION

To conclude I believe that I have developed a well thought out student software system using the practises of functional programming stop I have learned how to use the stream API as well as records and I'm confident in my knowledge and application of them. Although they could be improvements, I am happy with my final product.

# ESTIMATED GRADE

Functional principled: B

Report: B

# REFERENCES

Anil, et al. “‘functions Are First Class Values’ What Does This Exactly Mean?” Stack Overflow, stackoverflow.com/questions/10777333/functions-are-first-class-values-what-does-this-exactly-mean. Accessed 22 Jan. 2024.

Baeldung, Written by: “How to Get the Last Element of a Stream in Java?” Baeldung, 8 Jan. 2024, www.baeldung.com/java-stream-last-element.

Dapangma, et al. “Java 8 Nested Streams : Return a Value in Last Stream.” Stack Overflow, stackoverflow.com/questions/43498884/java-8-nested-streams-return-a-value-in-last-stream. Accessed 22 Jan. 2024.

Elliott, Eric. “Master the JavaScript Interview: What Is a Pure Function?” Medium, JavaScript Scene, 24 Aug. 2021, medium.com/javascript-scene/master-the-javascript-interview-what-is-a-pure-function-d1c076bec976.

Elliott, Eric. “Master the JavaScript Interview: What Is a Pure Function?” Medium, JavaScript Scene, 24 Aug. 2021, medium.com/javascript-scene/master-the-javascript-interview-what-is-a-pure-function-d1c076bec976.

Java Docs. “Enum Types.” Enum Types (The JavaTM Tutorials > Learning the Java Language > Classes and Objects), docs.oracle.com/javase/tutorial/java/javaOO/enum.html. Accessed 22 Jan. 2024.

Java Docs. Stream (Java Platform SE 8 ), 8 Jan. 2024, docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html.

Pankaj. “How to Use Remove() Methods for Java List and ListArray.” DigitalOcean, DigitalOcean, 18 Nov. 2022, www.digitalocean.com/community/tutorials/java-list-remove-methods-arraylist-remove.

Pankaj. “Java Stream Collect() Method Examples.” DigitalOcean, DigitalOcean, 3 Aug. 2022, www.digitalocean.com/community/tutorials/java-stream-collect-method-examples.

Study Smarter. First Class Functions, www.studysmarter.co.uk/explanations/computer-science/functional-programming/first-class-functions/. Accessed 22 Jan. 2024.

Ugarte, Written by: Alejandro. “Guide to Stream.Reduce().” Baeldung, 8 Jan. 2024, www.baeldung.com/java-stream-reduce.

“Using Java Streams in Java 8 and Beyond.” JRebel by Perforce, www.jrebel.com/blog/java-streams-in-java-8. Accessed 22 Jan. 2024.

“What Are Java Records and How to Use Them alongside Constructors and Methods?” GeeksforGeeks, GeeksforGeeks, 22 June 2022, www.geeksforgeeks.org/what-are-java-records-and-how-to-use-them-alongside-constructors-and-methods/.