Scala Evaluation

Name: Ashley Dunn

Student Number: s2372678

2025

Table of Contents

[Functional Thinking 2](#_Toc196480236)

[Functional Programming Style 2](#_Toc196480237)

[Comparison of Functional and Imperative Styles 2](#_Toc196480238)

[Different Choice of Language 3](#_Toc196480239)

[Conclusion 3](#_Toc196480240)

# Functional Thinking

I found it challenging to switch to the functional style of thinking, especially when trying to work with immutability and higher-order functions. At first, this approach made the code feel more complex. However, as I got more comfortable with the syntax, I began to appreciate the advantages of functional programming. I aimed to keep my code functional, especially when working with collections. I used constructs like map and get to replace traditional loops, which I initially tried to implement using for and foreach. I also used Option and pattern matching to handle errors, which provided more predictable and readable results compared to traditional try/catch blocks

# Functional Programming Style

I made a conscious effort to implement functional programming principles throughout my solution. For example, I focused on immutability and tried to write pure functions, such as ‘getCurrentFoodPrice’, which transformed data without modifying it. This allowed the code to be more predictable and easier understand. However, when dealing with user input, I had to resort to more imperative logic because I found it difficult to make that part fully functional. Even though I used Try for error handling in some places, I often turned to pattern matching, as it felt more in line with functional programming principles. Functions like map allowed me to make the code more modular, though I still struggle with some of these concepts, such as foldLeft and did not implement this in my application.

# Comparison of Functional and Imperative Styles

Switching to functional programming initially felt like a more complicated approach, especially coming from an imperative style where the focus is on step-by-step instructions and changing states. However, as I became more familiar with functional programming, I started seeing the advantages. The use of immutability and higher-order functions helped make the code clearer and more predictable, in contrast to the often-complex state changes and side effects in imperative programming. By enforcing pure functions, side effects were minimized.

One aspect of my solution I’m particularly proud of is the compareFood function. It’s a good example of functional programming because it relies on smaller, modular functions. The compareFood function uses the getFoodItem helper function to safely retrieve the price of food items and applies pattern matching to handle any errors. This approach contrasts with the imperative style, where handling such logic might involve more explicit state manipulation and error handling scattered throughout the code. Functional programming allowed me to break down the task into smaller, reusable functions, keeping the code clean and easy to follow. By focusing on immutability, I was able to maintain a consistent state throughout the code, making it easier to manage.

While adopting a functional style of thinking initially increased the complexity of my code, the more I practiced, the clearer the advantages became. However, despite these benefits, I still find myself more drawn to the imperative style. It feels more natural to me, and I appreciate its direct approach to problem-solving. I can see how functional programming might be more beneficial in specific contexts, especially for creating maintainable and predictable code. Over time, I expect to find a balance between the two approaches, using functional principles where they make sense, but still relying heavily on the imperative style as I am more comfortable.

# Different Choice of Language

If I had the option to use a different programming language for this project, I would likely choose Java, C#, or Python, as I am most comfortable with them. While Java and C# would require more lines of code to achieve the same results, I am more familiar with their imperative style, and that would make development faster for me. Python, on the other hand, offers a concise, flexible syntax that would make it easier to quickly develop the project. However, I believe Scala’s functional approach offers clear benefits for structuring and maintaining the code, especially for this project. While other languages may be faster to implement in, Scala’s functional approach leads to cleaner, more maintainable code in the long run.

# Conclusion

Reflecting on the functional approach in this project, I found it pretty challenging at first, especially with concepts like immutability and higher-order functions. Shifting my thinking to a more functional style wasn’t easy, and I found it difficult to get the syntax right initially. But as I worked more with Scala and got more familiar with the concepts, I started seeing the benefits. The code started feeling more predictable, and using functions like map and pattern matching made the logic clearer and simpler. I still found it challenging to implement more advanced concepts like foldLeft, but I feel like I’m getting closer to understanding them. I really enjoyed diving into functional programming, but I need more time and practice to fully grasp these concepts. With continued practice, I believe I'll be able to apply these concepts more effectively in other projects.