Improvements of Existing Autograding System for Interactive Learning Functional Programming in Java

Nikola Dimitrijević[1], Nemanja Zdravkovic[1] and Milena Bogdanović[1]

[1] *Faculty of Information Technolgy, Belgrade Metropolitan University, Tadeuša Košćuška 63, 11000 Belgrade, Serbia*

Abstract

In Computer Science, functional programming is a programming paradigm which uses functions to describe the logic of what the program has to achieve. These are mathematical function-like constructs (e.g., lambda functions) that are evaluated in expression contexts. Google Trends currently ranks functional programming as more popular than object-oriented programming. Also, functional programming is very important in existing popular platforms like Hadoop and Apache Spark. Many students may ask what are the benefits and why should they use functional programming in Java. From its beginnings, best practices in Java have encouraged object-oriented programming, which is an extension of procedural programming - type of imperative programming. Since version 8 Java introduced Java developers to functional programming with lambda expressions, method references and predefined functional interfaces. This Java release effectively notified developers that it's no longer sufficient to think about Java programming only from the imperative, object-oriented perspective.

The first purpose of this paper is to present the benefits of adopting a functional paradigm to students which worked with the older versions of Java (6, 7 or earlier) and to the ones working with legacy code or using newer versions but without switching to functional paradigm. Secondly, we present some improvements to existing interactive system for online learning of functional programming in programming language Java, supporting automatic grading of given assessments. These improvements are aimed at students who already have basic knowledge of Java programming languages, and want to switch to functional programming. Finally, we analyzed our solution by comparing its content to multiple commercially available solutions and point out the advantages and disadvantages of each.

Keywords

Functional programming, eLearning, autograding, Big Data

1. Introduction to Functional Programming

Functional programming, a paradigm at the heart of computer science, has seen a resurgence in popularity and practical application in recent years. Unlike imperative programming, which focuses on how a program operates (through statements and instructions), functional programming emphasizes what the program should accomplish, using pure functions and immutable data.

At its core, functional programming is about building software by composing pure functions. A pure function is one that, given the same input, will always return the same output and does not cause any observable side effects. This concept is not only central to functional programming but also to the mathematical notion of a function, making the paradigm highly expressive and predictable [1].

Proceedings for the 14th International Conference on e-Learning 2023, September 28-29, 2023, Belgrade, Serbia
EMAIL: nikola.dimitrijevic@metropolitan.ac.rs (A. 1); nemanja.zdravkovic@metropolitan.ac.rs (A. 2); milena.bogdanovic@metropolitan.ac.rs (A. 3)
ORCID: 0000-0002-8038-0642 (A. 1); 0000-0002-2631-6308 (A. 2); 0000-0003-0316-4484 (A. 3)

One of the key principles of functional programming is immutability. Immutability in this context refers to the idea that once a data structure is created, it cannot be altered. This principle reduces complexity and improves the predictability of code. Side-effect-free functions, another cornerstone of functional programming, contribute to the reliability and maintainability of software, as they do not alter the state outside their scope [2].

In recent years, functional programming has gained traction in the software development industry, particularly in the development of large-scale, concurrent applications. Languages like Haskell and Scala, and features in Java 8 such as lambda expressions, have brought functional programming concepts to the forefront of software design [3].

Functional programming's emphasis on immutability and stateless functions makes it particularly well-suited for concurrent and distributed systems. In these systems, managing state and dealing with mutable data can lead to complex issues. Functional programming alleviates these challenges, making it easier to write safe and efficient concurrent code [4]. The influence of functional programming extends to big data processing and cloud computing. Frameworks like Apache Spark and Hadoop MapReduce, which are fundamental in the processing of large datasets, employ functional concepts such as higher-order functions and lazy evaluation to efficiently process data across distributed systems [5, 6]. Functional programming offers a different approach to software development, focusing on pure functions and immutable data. Its principles are increasingly being integrated into modern software practices, influencing the development of concurrent, distributed, and big data applications. As the industry continues to evolve, the relevance and application of functional programming are likely to expand further.

2. Functional Programming and Big Data

In the rapidly evolving landscape of software development and data processing, functional programming has emerged as a key paradigm, not just in theory but also in practical applications. This paradigm shift is particularly evident in the realm of big data and distributed computing, where functional programming principles have become integral to the design and operation of major platforms. A prime example of this trend is the significant role functional programming plays in existing popular platforms like Hadoop and Apache Spark. These platforms, which are at the forefront of handling large-scale data processing, leverage functional programming concepts to enhance efficiency, scalability, and fault tolerance. This integration of functional programming into such platforms is not incidental but stems from several core advantages that this programming style offers in dealing with complex data processing tasks [5, 6].

2.1. Immutability and Parallelism

Functional programming promotes the use of immutable data, which is crucial for efficient parallel execution of operations. In the context of distributed systems like Hadoop and Apache Spark, immutability allows data operations to be executed in parallel without the risk of state conflicts or the need for complex locking mechanisms. This is particularly important in big data processing, where parallelism plays a key role in achieving high performance [6].

2.2. MapReduce Paradigm

Hadoop utilizes the MapReduce paradigm, deeply rooted in functional programming. MapReduce consists of map and reduce functions, which are concepts taken from functional programming. The map function processes input data and transforms it into intermediate results, while the reduce function aggregates these results into the final output. This model is naturally suited to the functional style of programming, where functions are first-class citizens and shared state is avoided [7].

2.3. Apache Spark and Functional API

Apache Spark, often used for big data processing, provides an API that is strongly inspired by functional programming, especially in its Scala interface. Spark uses concepts like RDD (Resilient Distributed Dataset) and Datasets, which enable functional operations such as map, filter, reduce, and aggregate on distributed data sets. These operations allow developers to write code that is both expressive and efficient for parallel execution [5].

2.4. Scalability and Resilience

The functional approach in Spark and Hadoop contributes to better scalability and resilience of the systems. Side-effect-free functions are easier to test and understand, which is crucial in complex distributed systems. Also, the functional approach facilitates the re-execution of operations in case of failures, important for system resilience [8].

Functional programming plays a key role in the design and implementation of modern distributed data processing systems, such as Hadoop and Apache Spark. The principles of functional programming, such as immutability, problem decomposition into functions, and avoidance of shared state, are essential for the efficiency, scalability, and resilience of these systems.

3. Curriculum for Functional Programming in Java 8 and Beyond

This curriculum is meticulously crafted to guide learners through the intricacies of functional programming in Java, particularly emphasizing the transformative features introduced in Java 8 and further refined in subsequent versions. As Goetz [9] insightfully points out, the advent of Java 8 marked a paradigm shift in Java programming, steering it towards the functional programming approach1. Additionally, Schildt [10] provides a comprehensive beginner's guide to Java, which is an invaluable resource for understanding the basics and nuances of Java programming. Warburton [11] further complements this by offering a focused exploration of lambda expressions and functional programming in Java 8, making it accessible to a broader audience.

This curriculum is ideally suited for individuals who have a foundational understanding of Java and are keen to explore the realm of functional programming, which has become increasingly relevant in modern software development. In this journey, learners will delve into the essence of functional programming as conceptualized in Java, starting from the basic principles and gradually progressing to more advanced features and concepts. The curriculum not only covers the theoretical aspects but also provides practical insights and applications, ensuring a holistic understanding of functional programming in the context of Java's ongoing evolution.

Table 1
Curriculum for Functional Programming.

Module Title	Topics
Module 1 - Introduction to Functional Programming in Java	Overview of Functional Programming Evolution of Java: From Imperative to Functional Functional Programming Concepts: Pure Functions, Immutability
Module 2	Syntax and Structure of Lambda Expressions

Lambda Expressions and Functional Interfaces	Functional Interfaces in Java (Function, Predicate, Consumer, Supplier) Method References and Constructor References
Module 3 Streams API	Introduction to Streams Stream Operations: filter, map, reduce, collect Advanced Stream Operations: flatMap, sorted, distinct Infinite Streams and Lazy Evaluation
Module 4 Collectors and Data Processing	The Collectors Class Grouping and Partitioning Data Collecting into Maps and Custom Collections
Module 5 Optional and New Date/Time API	Using Optional to Avoid NullPointerException New Date and Time API in Java 8 Best Practices with Optional and Date/Time API
Module 6 Parallel Data Processing and Concurrency Enhancements	Parallel Streams and their Underlying Mechanism Fork/Join Framework Enhancements in Concurrency API
Module 7 Advanced Topics and New Features in Later Java Versions	Java 9 and beyond: Enhancements in Streams and other APIs Reactive Programming with Java Project Loom and Virtual Threads
Module 8 Project and Practical Applications	Designing a Functional Interface-based Application Real-world Use Cases of Functional Programming in Java Project: Building a Data Processing Application using Java Streams

4. Autograder and functional programming

4.1. Overview of autograders

Autograders are software tools that assist educators, including professors and teaching assistants, by automating the grading process of student assignments. This not only reduces the workload of manually evaluating each submission but also eliminates potential biases from the grading process. This technology is particularly beneficial for students in Computer Science (CS) and Electrical Engineering (EE), where assignments predominantly involve coding tasks. The concept of autograders in these fields has a history spanning over fifty years [12, 13].

Initially, autograders were basic, capable of assessing only straightforward programming tasks and primarily supporting procedural programming. These early systems typically provided binary feedback, labeling student work as either "correct" or "incorrect" based on a rigid set of criteria. The latest generation of autograders has developed alongside advancements in high-speed internet and web technologies [14-19]. These systems often feature a web-based application, allowing students to code directly in a browser without needing a local interpreter, compiler, or integrated development environment (IDE). This approach also lessens the hardware demands on students' personal or campus computers. Modern autograders are more versatile, supporting multiple programming languages and paradigms, including object-oriented and functional programming, unlike their predecessors, which were mostly confined to procedural programming.

Despite the availability of various autograder tools, as noted in the literature [17], many are custom-designed for specific languages and requirements. Besides tools created for higher education institutions, there are also specialized autograders in Massive Open Online Courses (MOOCs) and commercial platforms, often tailored for particular courses or programming language basics. Autograders face two primary challenges: technical and pedagogical [20]. Technical challenges involve ensuring security against potentially harmful code in assignments and integrating with the Learning Management Systems (LMS) of educational institutions. Pedagogically, the inconsistency in grading systems is a concern. While some autograders still rely on the basic correct/incorrect feedback model, more advanced systems use a detailed step-by-step test case evaluation. However, there is no standardized model or widely accepted grading recommendation for these systems.

4.2. Proposed model

We have proposed a system that was primarily developed for learning basic and object-oriented programming, and in this paper, it has been expanded to include support for functional programming. The system we propose is built upon an existing autograder system, originally developed for teaching fundamental programming concepts and object-oriented programming [15-18]. As depicted in Figure 1 on the following page, this system includes:

- Initial Assessment: Students begin by taking a self-assessment on data structures to establish a starting point. This could be a brief quiz or a set of multiple-choice questions..
- Learning Module: Students are introduced to advanced data structure topics through lessons provided in written, video, and animated formats.
- Assessment type 1: Students are tasked with a relatively easy to moderately challenging exercise. They write code in a browser-based text editor, where a Java compiler operates server-side. The exercise includes some pre-written code.
- Assessment type 2: More challenging exercises are presented, requiring students to write code from scratch in a text editor without any pre-written code.
- Progress Tracking: After completing the assessments for each topic, the autograder system notifies students of their success in the current topic. Progression to subsequent topics is contingent on successful completion of previous ones. At the course's conclusion, students retake the initial assessment (or a similar one) to gauge their learning progress.

After each of the topics' assessments, an autograder system informs the student if they have passed the current topic successfully, and can only continue with the next topic if the previous is passed. At the end of the whole course, the student completes the same (or similar) baseline test to self-assess their progress.

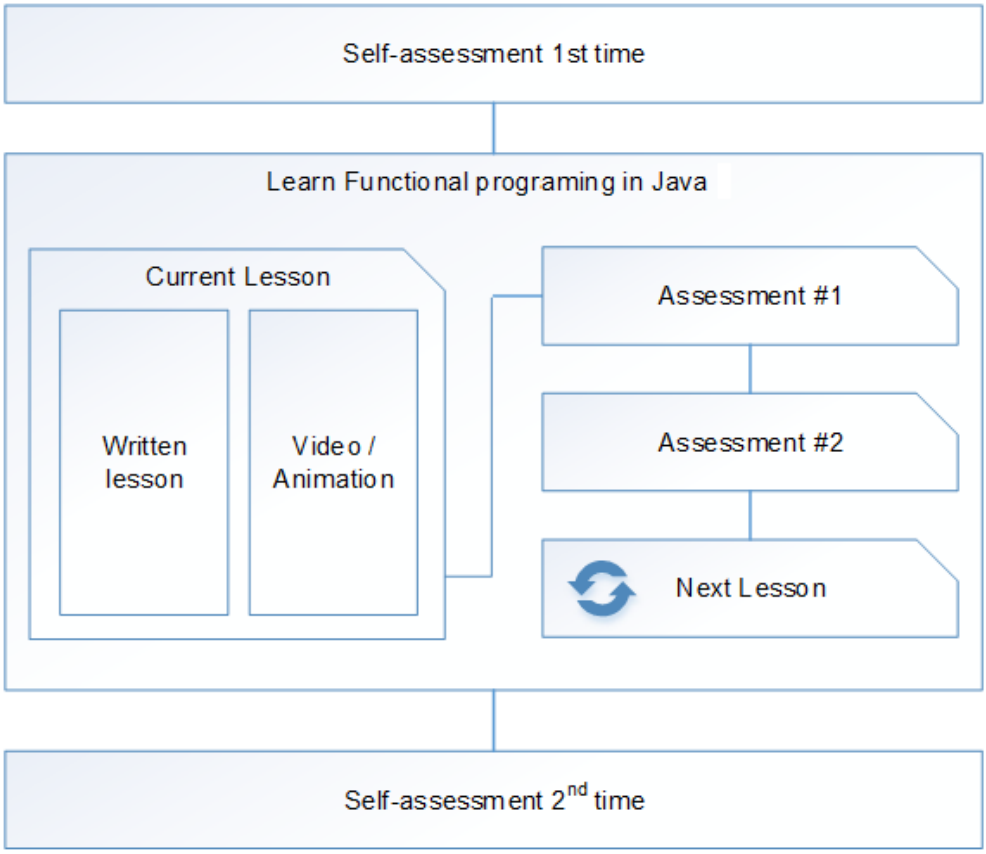

Figure 1: Components of the learning system model

4.3. Comparison of code using imperative and functional approaches

Within the framework of our developed autograder system, a variety of tasks are designed to enhance the learning experience of students. One such task is as follows:

The system presents students with a specific challenge: Convert the code shown in Figure 2 into code that effectively utilizes functional programming, leveraging the Java Stream API, and employing the methods filter() and collect(). Figure 2 serves as a practical illustration, demonstrating how a particular programming task can be approached in two distinct ways in Java: the traditional imperative approach and the modern functional approach. This task, specifically, involves filtering a list of numbers, retaining only those that are greater than 10. The imperative method is shown as a starting point, and students are tasked with transforming this into a more streamlined, functional version using Java 8 features.

This exercise not only reinforces their understanding of functional programming concepts but also allows them to directly compare and contrast different programming paradigms within Java, thereby deepening their comprehension and skill in the language. This task is representative of the kind of practical, hands-on challenges that are integral to our autograder system, designed to provide students with real-world programming scenarios that enhance their coding skills and theoretical knowledge."

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class ImperativeExample {
    public static void main(String[] args) {
        List<Integer> numberList = Arrays.asList(1, 12, 8, 20, 5, 18);
        List<Integer> filteredList = new ArrayList<>();

        for (Integer number : numberList) {
            if (number > 10) {
                filteredList.add(number);
            }
        }

        System.out.println(filteredList); // Ispisuje: [12, 20, 18]
    }
}
```

Figure 2: Example of code using imperative approach without functional programming

In the functional approach, we use the Java Stream API. The stream() method converts the list into a stream, the filter() method applies a lambda expression that specifies the filtering condition, and the collect() method gathers the results and returns them in a list. Figure 3 shows how the same task can be solved using a functional approach with Java 8.

```java
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class FunctionalExample {
    public static void main(String[] args) {
        List<Integer> numberList = Arrays.asList(1, 12, 8, 20, 5, 18);

        List<Integer> filteredList = numberList.stream()
                .filter(number -> number > 10)
                .collect(Collectors.toList());

        System.out.println(filteredList); // Ispisuje: [12, 20, 18]
    }
}
```

Figure 3: Example of code using functional approach with Java 8

5. Conclusion

In this paper, we have explored the characteristics of functional programming and proposed a comprehensive curriculum for functional programming in Java 8 and its subsequent versions. We have also presented significant enhancements to an existing interactive system for the online learning of functional programming in Java, emphasizing support for automatic grading of assessments. These improvements are meticulously designed for students who have a foundational understanding of Java

programming and are keen to transition to functional programming paradigms. The introduction of an automatic grading system has streamlined the learning process, offering immediate and personalized feedback, which is vital for effective learning. The updated curriculum, encompassing comprehensive modules on key functional programming concepts in Java, such as lambda expressions and the Stream API, ensures that learners are thoroughly prepared to comprehend and implement these advanced programming techniques.

Moreover, the integration of an interactive coding environment within the system fosters hands-on learning and experimentation, which is crucial for a deep understanding of functional programming nuances. The system's provision of personalized learning paths addresses the varied needs of students, accommodating different learning styles and paces.

In conclusion, the enhancements made to the interactive system mark a significant progression in the realm of programming education. By aligning the system with the latest trends in Java programming and concentrating on the practical application of functional programming concepts, we have developed a robust educational platform. This platform not only imparts knowledge but also readies students for the evolving demands of the software development industry. Our approach effectively bridges the gap between traditional Java programming and the contemporary functional programming paradigm, equipping learners with the necessary skills and knowledge to thrive in today's technological landscape.

6. Acknowledgements

This paper was supported by the Ministry of Education, Science and Technological Development, Republic of Serbia (Project III44006).

7. References

[1] P. Hudak, Conception, evolution, and application of functional programming languages, ACM Computing Surveys, 1989.
[2] M. Odersky, L. Spoon, & B. Venners, Programming in Scala, Artima, 2010.
[3] P. Van Roy, & S. Haridi, Concepts, Techniques, and Models of Computer Programming, MIT Press, 2004.
[4] P. Haller, & M. Odersky, Scala Actors: Unifying thread-based and event-based programming, Theoretical Computer Science, 2009.
[5] Zaharia, M., et al. (2012). Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation.
[6] J. Dean, & S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Communications of the ACM, 2008.
[7] R. Lämmel, Google's MapReduce Programming Model — Revisited, Science of Computer Programming, 2008.
[8] Armbrust, Michael, et al. Spark sql: Relational data processing in spark. In: Proceedings of the 2015 ACM SIGMOD international conference on management of data. 2015. p. 1383-1394.
[9] B. Goetz, Java 8: Functional features and libraries, Oracle, 2014.
[10] H. Schildt, Java: A Beginner's Guide, Seventh Edition, McGraw-Hill Education, 2017.
[11] R. Warburton, Java 8 Lambdas: Functional Programming for the Masses, O'Reilly Media, 2014.
[12] G. E. Forsythe, and N. Wirth, Automatic grading programs, Communications of the ACM, vol. 8, no. 5, pp. 275-278, 1965.
[13] H. Aldriye, A. Alkhalaf, and M. Alkhalaf, "Automated grading systems for programming assignments: A literature review," International Journal of Advanced Computer Science and Applications, vol. 10, no. 3, pp. 215-221, 2019.
[14] C. Douce, D. Livingstone, and J, Orwell, "Automatic test-based assessment of programming: A review," Journal on Educational Resources in Computing (JERIC), vol. 5, no. 3 pp. 4-es, 2005.

[15]J. C. Caiza and J. M. del Alamo, “Programming assignments automatic grading: review of tools and implementations,” In Proc. of the 7th international technology, education and development conference (INTED2013), pp. 5691, 2013.
[16]G. Haldeman, M. Babeş-Vroman, A. Tjang, and T. D. Nguyen. “CSF: Formative Feedback in Autograding,” ACM Transactions on Computing Education, vol. 21, no. 3, 2021.
[17]S. Krusche and A. Seitz, “ArTEMiS: An automatic assessment management system for interactive learning,” in Proc. of the 49th ACM Technical Symposium on Computer Science Education, pp. 284-289, 2018.
[18]Fangohr, H., O'Brien, N., Prabhakar, A. and Kashyap, A., 2015. “Teaching Python programming with automatic assessment and feedback provision,” arXiv preprint arXiv:1509.03556.
[19]D. Radošević, T. Orehovački, and A. Lovrenčić, “Verificator: educational tool for learning programming,” Informatics in Education, vol. 8, no. 2, pp. 261-280, 2009.
[20]G. E. Forsythe, and N. Wirth, “Automatic grading programs,” Communications of the ACM, vol. 8, no. 5, pp. 275-278, 1965.