International Journal for Quality Research 17(2) 313–324
ISSN 1800-6450

Nikola Dimitrijević
Nemanja Zdravković
Vladimir Milićević[1]

Article info:
Received 09.04.2022.
Accepted 11.12.2022.

UDC – 37.018.43:004
DOI – 10.24874/IJQR17.02-01

AN AUTOMATED GRADING FRAMEWORK FOR THE MOBILE DEVELOPMENT PROGRAMMING LANGUAGE KOTLIN

Abstract: *With the recent rise of the Kotlin programming language as the main contender for Android mobile development, very few courses in Higher Education Institutions exist which incorporate Kotlin as one of the main languages. In addition, various online course platforms which offer learning Kotlin are still very low in number, and the ones that do exist are expensive. In this paper, an e-learning framework for the Kotlin programming language is presented, supporting automatic grading of given assessments. This framework is aimed at students who already have basic knowledge of Java (or similar) programming languages, and want to switch to mobile development. The solution focuses on the development of an interactive course in Kotlin. Furthermore, to compare our solution with commercially available ones, we point out the disadvantages of currently available Kotlin courses, such as the level of previous knowledge needed, or a need for a specific development environment.*

Keywords: *autograders, e-Learning, Java, Kotlin, mobile development*

1. Introduction

In the wake of the COVID-19 pandemic, many Higher Education Institutions (HEIs) are switching from the traditional classrooms, to blended and/or completely online learning. The ad- vantages of online learning includes more flexibility in student participation, and easy scaling to a large number of enrollments (Galan et al., 2019). In addition, web-based learning, especially in the fields of Computer Science (CS), Information Technology (IT) and Software Engineering (SE) require innovative tools to support new teaching and learning methodologies, such as automatic graded systems, virtual and augmented reality systems, and an overall gamification of learning (European Commision, 2021). The shift from in-classroom learning to learning from home has impacted CS, IT, and SE students perhaps the most, as they needed to rely on a home computer for learning assignments and tasks.

Since its inception over a quarter of a century ago, the Java programming language is still among the most popular programming languages for general use (IEEE Spectrum, 2021, TIOBE index, 2021, Gavrilović et al., 2018). As a general-purpose language, Java has gained its popularity by being platform-independent, attributable to the Java Virtual Machine (JVM). The use of JVM has led to new programming languages, such as Groovy, Scala, and most notably Kotlin (Urma, 2014). Since Google announced Kotlin as one of the officially supported languages for Android development in 2017 due to its properties of being concise, expressive, null-safe, the ever-increasing

[1] Corresponding author: Vladimir Milićević
Email: vladimir.milicevic@metropolitan.ac.rs

popularity of Kotlin is evident (Oliveira et al., 2020).

In this paper, we present a web-based application for learning the Kotlin language with autograding support, suitable to follow a mobile development course offered to CS, IT, and SE students. The rest of the paper is organized as follows: Section 2 presents our motivation and literature overview on autograders, and points as the main advantages of Kotlin for mobile development, with highlighted similarities and differences to Java. In Section 3, we present our autograder-supported system, provide a proposed course curriculum, as well as details on assessment design. Section 4 presents results and discussion regarding students' experience using our system for learning Kotlin. Finally, Section 5 concludes the paper and gives plans for future research.

2. Motivation

Courses in mobile application development offered to CS, IT, and SE students are gaining more popularity, due to the pervasive availability of devices such as smartphones, tablets, and similar smart portable devices, and their extensive use of different Internet-based services (Bruzual et al., 2020, Burd et al., 2012). In addition, this trend is supported by a high demand in industry, resulting in opportunities to both students and institutions which offer these courses (Fenwick et al., 2011). The authors of this paper are motivated to firstly investigate published works regarding courses for mobile development offered by HEIs and commercial platforms, as well as the rising popularity of the Kotlin programming language, in order to better understand the need of a specifically-designed course for Kotlin with autograder support.

The number of portable smart device users is increasing every year, as shown in Figure 1 (Riadi, 2017, Swidan et al., 2021). Following this trend, HEIs, as well as other learning platforms, such as Massive Open Online Courses (MOOCs) and Small Private Online Courses (SPOCs), have started to incorporate mobile development courses in their offers. For instance, Coursera offers more than 300 results when searching for mobile development (Coursera, 2021).

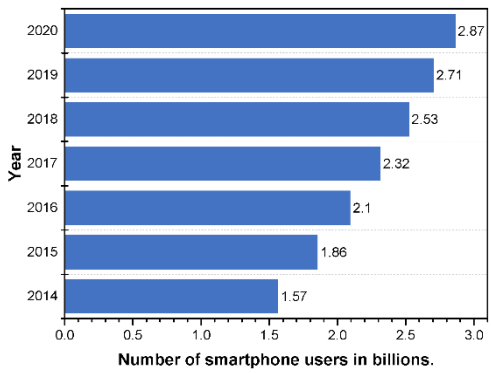

Figure 1. Number of smartphone users worldwide from 2014 to 2020.

2.1. Related work

The idea of systems that can automatically grade students' input, referred to autograders throughout the paper, is not new, especially in STEM universities. In these universities, as the majority of the assignments are in the form of writing a computer program to run a specific task. Indeed, the first paper can be traced down to over fifty years ago (Forsythe & Wirth, 1965). Autograders help teaching staff by reducing the work load of having to manually grade all students' assignments. Furthermore, the bias of the teaching staff is also removed. Although present for over half a century, autograders have evolved with the emergence of new technologies and programming languages (Caiza et al., 2013, Douce et al., 2005, Shah, 2003). We can distinguish three generations of autograders. The first generation of these systems was simple, and were mostly tied to lower-level programming languages. By checking a strict set of successive instructions written by a student, the autograder would yield "right" or "wrong" as an answer. These early

programs were able to check only simple programming assignments, and supported mostly the procedural programming paradigm. The second generation employed operating system tools, and programming languages such as C, C++ and Java were used to build these systems (Deldago-Perez et al., 2020, Insa et al., 2021). In turn, these systems could also check problems in their respective languages. Finally, the third generation of autograders emerged with the rise of the high- speed Internet and modern web and mobile development technologies. In these systems, used in HEIs and MOOC/SPOC platforms alike, a web application is hosted on a server. A learner can write their program in a browser, without the need of an interpreter or compiler or an Integrated Development Environment (IDE) installed on their home computer. In addition, this generation of autograders often supports different programming languages with several programming paradigms, such as object-oriented programming or functional programming.

As was pointed out in (Krusche & Seitz, 2018), many existing autograder tools exist today, both as stand-alone commercial platforms or as a part of an HEI's Learning Management System (LMS). The majority of these systems are usually custom-tailored for specific programming languages and specific requirements as well. Furthermore, autograders still suffer from technical and pedagogical issues. Technical issues include over- all security and protection from potential malicious code written in the assignment, as well as integration problems, while pedagogical issues include a non-uniform grading system, often set by each autograder's developers. Some autograders still use the same correct/incorrect system, while others may use a step- by-step per test-case grading system (Bruzal et al., 2020). As of writing this paper, the authors could not find a uniform model or recommendation for grading.

2.2. The Kotlin programming language

As was stated in (Urma R. G., 2014), the extensive use of Java and its JVM produced several new Java-based programming languages, such as Kotlin, which began its development by JetBrains in 2010, releasing a stable version in 2016. In this subsection, the authors of this paper will highlight the differences of Java and Kotlin, and highlight the reasons Google mentioned it is "concise, expressive, and designed to be type and null-safe" (Oliveira et al., 2020).

As Kotlin is based on Java, it is a statically typed language, with full support for object-oriented and functional programming paradigms. Indeed, Kotlin can be used both in object-oriented and in functional programming style, or in a mix of both styles (Kotlin Language Documentation, 2021). Although Kotlin uses JVM, it can also be compiled into JavaScript, and into machine code as well. Due to Java interoperability, it is fairly easy to use any existing Java framework and/or library. Other than mobile application development, Kotlin is increasingly being used for backend development, and even older frameworks, such as Spring, provide support for Kotlin (Arhipov, 2020). In addition, Kotlin's support for coroutines helps build server-side applications that scale to massive numbers of clients with modest hardware requirements.

Whereas static methods are used for function declaration outside classes in Java, Kotlin allows the declaration of function outside the classes. The approach in Java results in classes whose instances are never created, and static methods are called instead (Gotseva et al., 2019). The main feature of Kotlin is the support for non-nullable types, making applications less susceptible to the so-called null point dereference, i.e. `NullPointerException`.

Regarding data types, Java and Kotlin share more similarities than differences. The main difference is the base class. Namely, the base class in Java is Object, while Kotlin uses Any. Kotlin types are further divided into

nullable and non-nullable. In Java, primitive types cannot be used as generic types, as they do not support method calls, and hence cannot obtain a null value. To overcome these restrictions, each primitive type is paired with a wrapper class. Primitive data types in Kotlin are stored on the stack, and they exist only in their field of view. In reference types, an address (reference) is stored in the stack, in the heap of which is the object itself. In Kotlin, the super-type for all types is called Any; however, it cannot contain a null value of null. Kotlin uses the `Any?` type when a null value is necessary. This applies for all data types – a question mark after the type name explicitly allows that variable to contain a null value. Similarly, when accessing an object of nullable types, the question-mark operator is used instead of the dot operator. An overview of data types in both languages is summarized in Table 1 (Gotseva et al., 2019). In addition, primitive and reference data types are differently organized in the memory. Kotlin does not use the keyword new for new object creation, whereas Java dedicates memory space for this newly created object. In Kotlin, an object is created by calling the constructor just like any ordinary function in the language.

Constructors are used in both languages to create objects as instances of a class. In addition, both languages support the declaration of multiple constructors. However, in Kotlin, a primary constructor exists, which is declared outside the class body, while other constructors which are therefore secondary are declared in the class body. The role of the primary constructor is to initialize the class, while the secondary constructor(s) help to include additional logic while initializing the class (Gotseva et al., 2019). A more detailed analysis of the similarities and differences can be found in (Gotseva et al., 2019), where the authors highlight not only data types, but also operations and expressions, main statements, functions and subroutines, and a more detailed comparison in classes and objects.

Table 1. Data type similarities between Kotlin and Java (Gotseva et al., 2019).

Data Type	**Kotlin**	**Java**
Integer	Integer Byte, Short, Int, Long	byte, short, int, long
Floating point	Float, Double	float, double
Boolean	Boolean	boolean
Alphanumeric character	Char	char
Alphanumeric string	String	string
Null object	null	null
Base class	Any	Object

The authors of this paper highlight the properties of Kotlin which Java does not have in Table 2, in a similar manner as conducted in (Dimitrijević et al., 2021).

Table 2. Features of Kotlin programming language which are different from Java (Dimitrijević et al., 2021).

Feature Available?	**Kotlin**	**Java**
Lambda expressions and Inline functions	Yes	No
Extension functions	Yes	No
Checked exceptions	No	Yes
Null-safety and smart casts	Yes	No
Primary constructors	Yes	No
First-class delegation	Yes	No
Type inference for variable and property types	Yes	No
Wildcard types	No	Yes
Declaration-site variance and Type projections	Yes	No
Range expressions	Yes	No
Operator overloading	Yes	No
Ternary operator	No	Yes
Data classes	Requires less code	Requires more code
Static members	No	Yes
Coroutines	Yes	No

3. System model

The constant increase of mobile device use, the gamification of education in all levels has led to an increase of student motivation, engagement and achievements throughout their studies, as was pointed out in (Garcia et al., 2020, Hursen & Bas, 2019, Khaleel et al., 2015, O'Connor & Cardona, 2019). Therefore, a new paradigm in distance learning, namely mobile learning, or M-learning, has emerged. M-learning, which is a subset of distance education, provides learners opportunities to learn through educational mobile applications (Kayaalp & Dinc, 2021). All this has led to the increase of courses in mobile development, offered by both commercial learning platforms and HEIs alike (Modesti, 2021).

In this paper, we present an autograder-supported framework for learning the Kotlin programming language, intended for mobile software development.

Our proposed system is based on a previously implemented autograder used for the purpose of learning basic programming concepts, but also objective- oriented programming. These UI components are shown in Figure 2. The upper part of the UI screen is divided into three areas. The upper right area presents the lesson, the lower right area presents the assignments, while the left side represents the code editor.

According to the given assignment, a learner can append existing code, edit existing code, or write new code from scratch. Every assignment consists of a group of tests to check the correctness of the learner's solution.

Finally, the lower part of the screen is the output of the autograder, which runs multiple tests after the student submits their code. For this purpose, we have applied the JUnit framework for Kotlin (Test code using Junit in JVM, 2021).

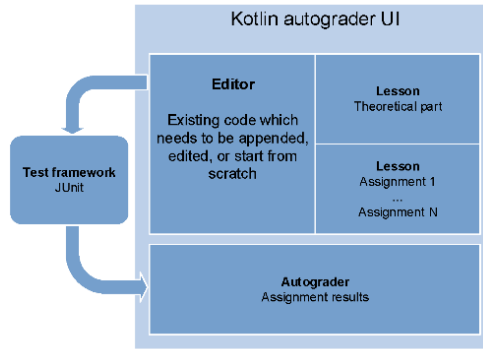

Figure 2. User Interface for the Kotlin autoautograder system.

Our application supports both web-and mobile interfaces. The application's architecture is given in Figure 3. The client-side application is entirely reliant on the set of services which are implemented as a separate project. The application itself has a monolithic architecture due to performance reasons. The server-side is comprised of the following components. The role of the Java Spring Book application with REST services is to run tasks, which client applications communicate using HTTP protocol. The MySQL relational database and file system are used for data storage, and the SMTP server is an external component used for email notifications.

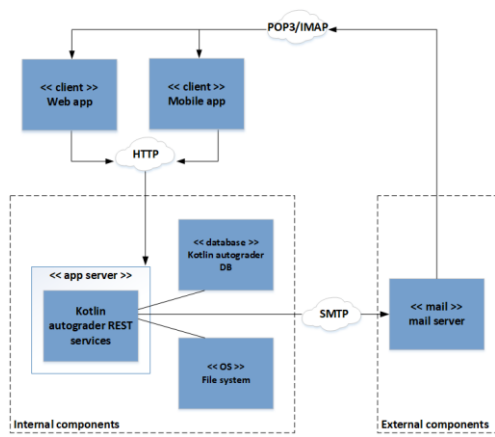

Figure 3. Architecture diagram.

The component diagram is shown in Figure 4. The Public compo- nent are used by learners for their activities such as access to lessons. The Administration component is used for management, and it uses Lesson Management and User Management, both implementing CRUD operations. The RDBMS server component is used for data storage, and is implemented as a relation model using JPA ORM framework. The Kotlin Compiler and Runner are used for compiling code inputs into executable code and executing and verifying assessment results, respectively. Finally, Figure 5 shows a sequence diagram of running an assessment.

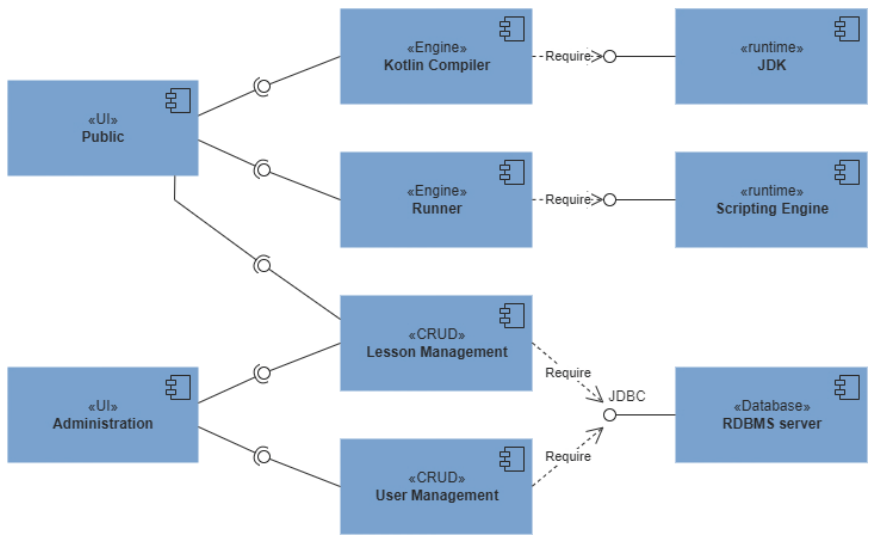

Figure 4. Component diagram

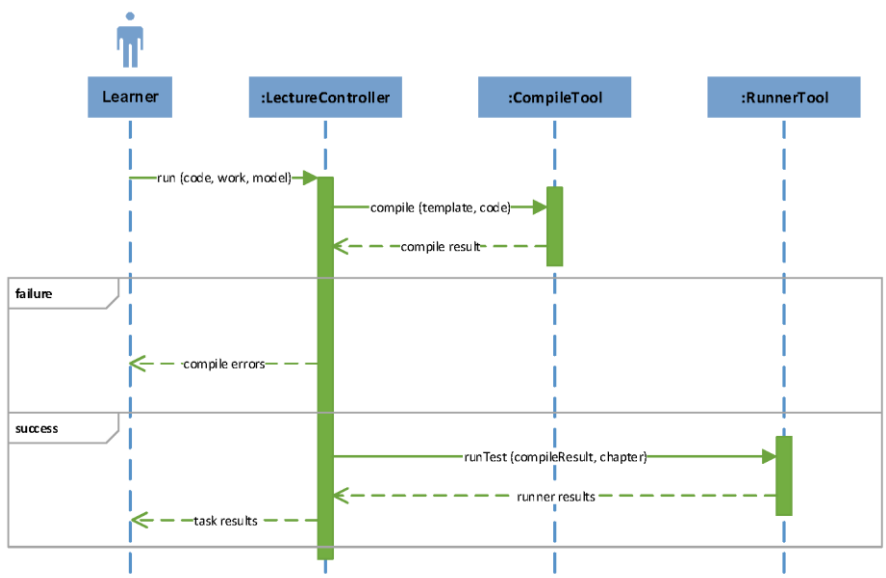

Figure 5. Sequence diagram

3.1. Assessment design

The proposed curriculum includes topics and assessments with different levels of complexity, paired with different functionalities needed for the realization of the given topic. Furthermore, some of the software components developed in the earlier assessments could be reused for later ones. This process of completing a given topic consists of four stages, and is shown in Figure 6.

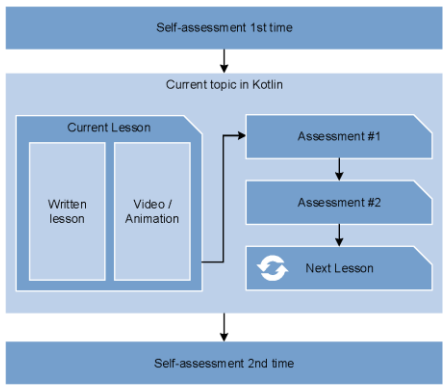

Figure 6. Components of the learning system model.

Baseline: The learner firstly completes a self-assessment in order to build a baseline. This self- assessment can be in the form of a short quiz, or multiple-choice questions.

Theoretical part: The learner is presented a given topic for the Kotlin programming language in written, video and/and animated form.

Assessment type 1: An easy-to-medium difficulty assignment is given to the learner. Afterwards, the learner inputs code within a text editor in-browser, with part of the code given to the learner.

Assessment type 2: A medium-t-hard assignment is given to the learner, with no code previously given, i.e. the learner writes from scratch. In both assessment types, the JUnit for Kotlin test framework is running as a server-side application, using multiple test cases in a step-by-step grading system. After each of the topics' assessments, the autograder informs the learner 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 learner completes the same (or similar) baseline test to self-assess their progress.

3.2. Curriculum proposal

Following the established curricula for Java and/or mobile development courses, we have designed a curriculum for this course which has four sections. A summary is given in Table 3, with sub-topics omitted. As of writing this paper, the authors included only general topics for learning the Kotlin language, and we plan to expand the topics with the emphasis on mobile development.

As an example, in Figure 7 we can see the screenshots of our application, in web and mobile formats, where the assessment is the evaluation of the Binomial Coefficient. The code listing for the autograder is given in Listing 1.

Listing 1. Test case listing for Binomial Coefficient task.

```
var result : List < SimpleResult > =
ArrayList ()
var actual = C(6, 2)
var isSuccess = actual == 15L
result += SimpleResult ("C(6, 2)",
isSuccess , 15, actual )
actual = C(3, 3)
isSuccess = actual == 1L
result += SimpleResult ("C(3, 3)",
isSuccess , 1, actual )
actual = C(7, 5)
isSuccess = actual == 21L
result += SimpleResult ("C(7, 5)",
isSuccess , 21, actual )
actual = C(13 , 7)
isSuccess = actual == 1716 L
result += SimpleResult ("C(13 , 7)",
isSuccess , 1716 , actual )
```

Table 3. Curriculum proposal.

Basics	Intro to OOP	Functional	Advanced OOP
Hello, World!	Creating Classes	Lambdas	Extension Functions
Data Types	Properties	Collections	Overloading
String Templates	Constructors	Member References	Data Classes
Expressions	Constraining Visibility	Higher-Order Functions	Interfaces
Loops	Packages	Manipulating Lists	Complex Constructors
Functions	Lists	Building Maps	Secondary Constructors
Nullable Types	Variable Argument Lists	Sequences	Inheritance
Non-Null Assertions	Sets	Local Functions	Abstract Classes
Exception Handling	Maps	Folding Lists	Composition
The Nothing Type	Property Accessors	Recursion	Class delegation

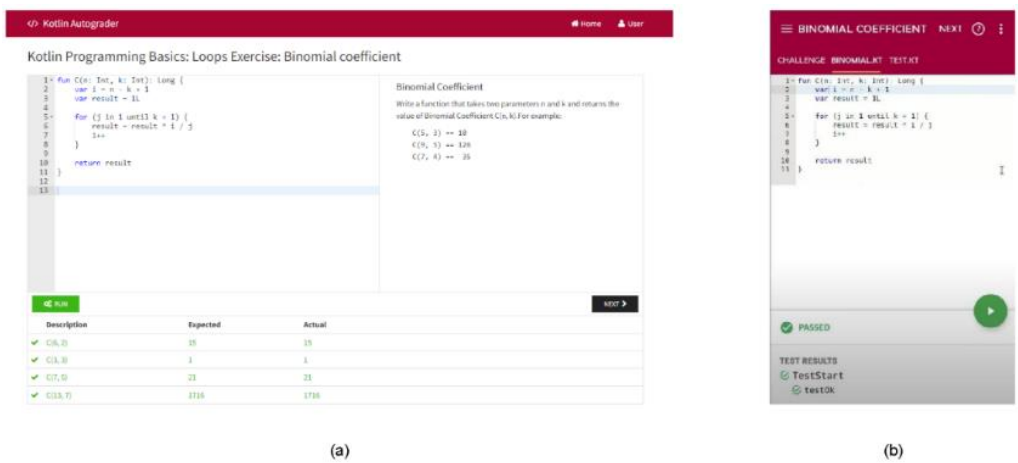

Figure 7. Screenshot of the web app client (a) and mobile client (b). Mobile client screenshot was taken in an emulator.

4. Results and discussion

The authors of this paper have firstly conducted a short three-question survey regarding a career in mobile software development, as well as students' knowledge about the Kotlin programming language. Students of 1st and 4th years of IT and SE bachelor studies were questioned. There was a total of 60 students that answered in the 1st year, and 24 students in the 4th year. The results are given in Fig. 8. Furthermore, the authors asked 4th year students to evaluate their experience using our solution, compared with literature found online, and the results are in Fig. 9. These students were already familiar with a similar language such as Java. Results show that 1st year students are still not sure about their career path, and only 37% have heard of Kotlin, whereas 4th year students have a better understanding of what Kotlin is used for. Interestingly, a higher percentage of 1st year students would like to learn Kotlin, even if some of them have not heard of the language previously.

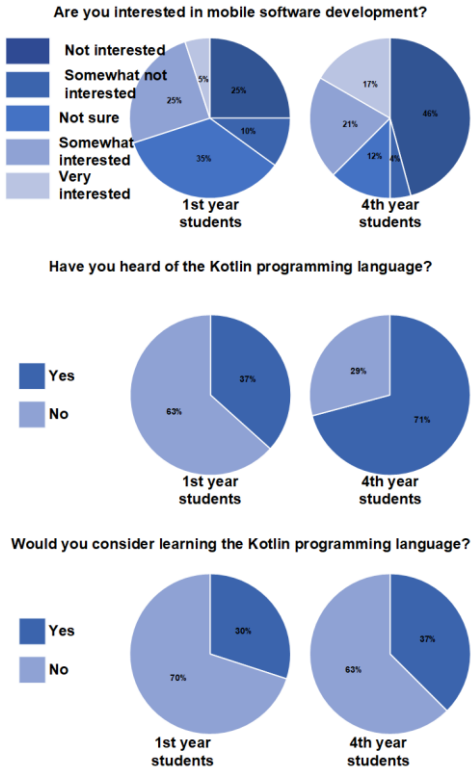

Figure 8. Survey regarding mobile development careers and Kotlin. Students of 1st and 4th year were questioned

The second set of graphs shows 4th year students' experience using our autograder framework, and the results are favorable compared to learning using only a book. Furthermore, 83% of students questioned would recommend this approach to learning a new programming language.

In particular, the importance of the application of the observed solution is reflected in the increase in passability in course CS330 - Development of mobile applications, which focuses on the Android operating system with a strong support in the Kotlin programming language. The following table follows three generations of students, all with the aim of introducing this learning tool into the course learning system.

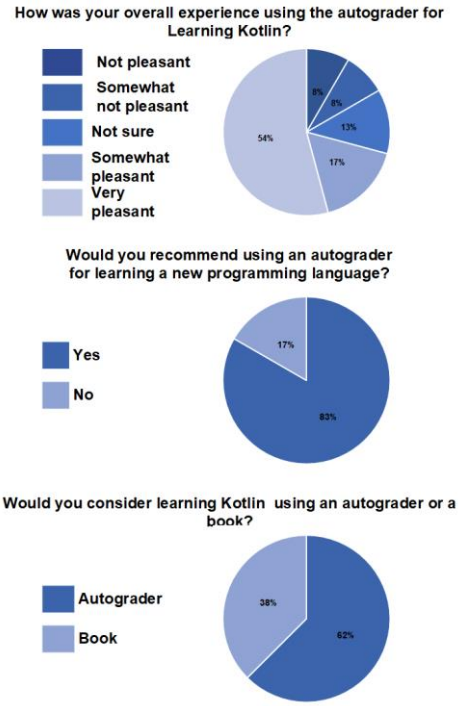

Figure 9. Survey after using the Kotlin autograder framework. Students of 4th year were questioned.

Table 4. Course: CS330 – Mobile Application Development passing progress.

Course: CS330 – Mobile Application Development			
School year	2017 - 2028	2018-2019	2019 - 2020
Number of students	74	64	97
Passed exam	37	38	60
Pass percentage	50.00	59.37	61.85

From the table it is possible to follow the progress of success in exam passing in course CS330 - Mobile Application Development over three consecutive school years. In the first year, there was no learning support in the form of an autograder-supported system. In the second year, the mentioned support is introduced with partial acceptance by the students. The third school year marks a significant interest of students, based on the positive experiences and

recommendations of students from the previous school year.

In coming years, the goal is to affirm students to use the created tools more intensively in order to achieve the projected pass rate of 66.6%.

5. Conclusion

This article presented an autograder-supported system for Kotlin exercises, aimed at those already familiar with Java. The proposed system and course curriculum shown to be effective in terms of student learning based on an analysis of exercise submissions and a 4th year student feedback survey, offering a simpler learning curve than courses encountered online. Furthermore, with the inclusion of detailed theoretical knowledge, clips and/or animations, the offered lessons would feel less like a step-by-step tutorial, but rather like a full course for CS and IT students. The addition of an advanced autograder allows our solution to hide the answers, encouraging students to think about the problem and not getting the right solution for the specific test or tests. in particular, the paper presented the progress in passing the course Mobile Application Development after the introduction of support in the form of autograder-supported system.

As the mobile development continues, it is planned to add more specific learning topics for Kotlin Android developers, and to further expand this type of system for multiple programming languages such as Python and Java (basic, intermediate and advanced topics).

Acknowledgment: This paper was supported in part by the Blockchain Technology Laboratory at Belgrade Metropolitan University, Belgrade, Serbia, and in part by the Ministry of Education, Science and Technological Development, Republic of Serbia (Project III44006).

References:

Arhipov, A. (2020). *Server-Side Development with Kotlin: Frameworks and Libraries*. Available at https://blog.jetbrains.com/kotlin/2020/11/server-side-development-with-kotlin-frameworks-and-libraries/.

Bruzual, D., Montoya Freire, M. L., & Di Francesco, M. (2020, June). *Automated assessment of Android exercises with cloud-native technologies*. In Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (pp. 40-46). Doi: 10.1145/3341525.3387430

Burd, B., Barros, J. P., Johnson, C., Kurkovsky, S., Rosenbloom, A., & Tillman, N. (2012). *Educating for mobile computing: addressing the new challenges*. In Proceedings of the final reports on Innovation and technology in computer science education 2012 working groups (pp. 51-63). Doi: 10.1145/2426636.2426641

Caiza, J. C., & Álamo Ramiro, J. M. D. (2013). Programming assignments automatic grading: Review of tools and implementations. *International Journal of Engineering Education*, *29*(5), 1183-1192.

Coursera (2021). Mobile App Development Courses. Available at https://www.coursera.org/.

Dimitrijević, N., Milićević, V., Zdravković, N., Cvijanović D. (2021). *Learning the Kotlin programming language using an autograding system*. Proc. of the 12th International Conference on e-Learning. (pp. 137–141).

Douce, C., Livingstone, D., & Orwell, J. (2005). Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing (JERIC), 5*(3), 4-es. Doi: 10.1145/1163405.1163409

Fenwick Jr, J. B., Kurtz, B. L., & Hollingsworth, J. (2011, March). *Teaching mobile computing and developing software to support computer science education*. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 589-594). Doi: 10.1145/1953163.1953327

Forsythe, G. E., & Wirth, N. (1965). Automatic grading programs. *Communications of the ACM, 8*(5), 275-278.

Galan, D., Heradio, R., Vargas, H., Abad, I., & Cerrada, J. A. (2019). Automated assessment of computer programming practices: the 8-years UNED experience. *IEEE Access, 7*, 130113-130119. Doi: 10.1109/ACCESS.2019.2938391

Garcia, I., Pacheco, C., Méndez, F., & Calvo-Manzano, J. A. (2020). The effects of game-based learning in the acquisition of "soft skills" on undergraduate software engineering courses: A systematic literature review. *Computer Applications in Engineering Education, 28*(5), 1327-1354. Doi: 10.1002/cae.22304

Gavrilović, N., Arsić, A., Domazet, D., & Mishra, A. (2018). Algorithm for adaptive learning process and improving learners' skills in Java programming language. *Computer Applications in Engineering Education, 26*(5), 1362-1382. Doi: 10.1002/cae.22043

Gotseva, D., Tomov, Y., & Danov, P. (2019, October). *Comparative study java vs kotlin. In 2019 27th National Conference with International Participation (TELECOM)* (pp. 86-89). IEEE. doi: 10.1109/TELECOM48729.2019.8994896

Hursen, C., & Bas, C. (2019). Use of gamification applications in science education. *International Journal of Emerging Technologies in Learning (iJET), 14*(01), 4. https://doi.org/10.3991/ijet.v14i01.8894

IEEE Spectrum (2021). *Top Programming Languages*. Available at https://spectrum.ieee.org/top-programming-languages/.

Insa, D., Pérez, S., Silva, J., & Tamarit, S. (2021). Semiautomatic generation and assessment of Java exercises in engineering education. *Computer Applications in Engineering Education, 29*(5), 1034-1050. Doi: 10.1002/cae.22356

Kayaalp, F., & Dinc, F. (2022). A mobile app for algorithms learning in engineering education: Drag and drop approach. *Computer Applications in Engineering Education, 30*(1), 235-250. Doi: 10.1002/cae.22453

Khaleel, F. L., Ashaari, N. S., Meriam, T. S., Wook, T., & Ismail, A. (2015, January). *The study of gamification application architecture for programming language course*. In Proceedings of the 9th international conference on ubiquitous information management and communication (pp. 1-5). Doi: 10.1145/2701126.2701222

Kotlin Language Documentation 1.6.10 (2021). Available at https://kotlinlang.org/docs/home.html/.

Krusche, S., & Seitz, A. (2018, February). *Artemis: An automatic assessment management system for interactive learning*. In Proceedings of the 49th ACM technical symposium on computer science education (pp. 284-289). Doi: 10.1145/3159450.3159602

Modesti, P. (2021). A script-based approach for teaching and assessing Android application development. *ACM Transactions on Computing Education (TOCE), 21*(1), 1-24. Doi: 10.1145/3427593

O'Connor, P., & Cardona, J. (2019). Gamification: A pilot study in a community college setting. *Journal of Education, 199*(2), 83-88. Doi: 10.1177/0022057419848371

Oliveira, V., Teixeira, L., & Ebert, F. (2020, February). *On the adoption of kotlin on android development: A triangulation study. In 2020 IEEE 27th International Conference on Software Analysis*, Evolution and Reengineering (SANER) (pp. 206-216). IEEE. doi: 10.1109/SANER48275.2020.9054859

Riadi, I. (2017). Forensic investigation technique on android's blackberry messenger using nist framework. *International Journal of Cyber-Security and Digital Forensics*, 6(4), 198-206.

Shah, A. R. (2003). Web-cat: A web-based center for automated testing (Doctoral dissertation, Virginia Tech).

Test code using JUnit in JVM (2021). Available at https://kotlinlang.org/docs/ jvm-test-using-junit.html/, 2021.

TIOBE index (2021). Available at https://www.tiobe.com/tiobe-index/java/.

Urma, R. G. (2014). *Alternative languages for the JVM a look at eight features from eight JVM languages*. Available at https://www.oracle.com/technetwork/articles/java/ architect-languages-2266279, 2014.

Nikola Dimitrijević
Faculty of Information Technologies, Belgrade Metropolitan University, Belgrade, Serbia
nikola.dimitrijevic@metropolitan.ac.rs

Nemanja Zdravković
Faculty of Information Technologies, Belgrade Metropolitan University Belgrade, Serbia.
nemanja.zdravkovic@metropolitan.ac.rs

Vladimir Milićević
Faculty of Information Technologies, Belgrade Metropolitan University Belgrade, Serbia
vladimir.milicevic@metropolitan.ac.rs