ISSN: 2289-7844 | e-ISSN: 0127-9750
DOI: https://doi.org/10.37134/jictie.vol10.2.9.2023

Journal of ICT in Education (JICTIE)
Volume 10, Issue 2, Pages 142-154, 2023

Research Article

Transitioning from Scratch to Java: A Mixed Method Investigation into Students' Learning Processes

Siti Sakinah Mohd Yusof[1*], Chow Lai Kim[2], Indrani Gopal[1], Julia Ahmad[3], Jun Yasmin Ahmad[3], Sulaiman Abd Anter[4]

[1]*Unit Sains Komputer, Kolej Matrikulasi Perak, Perak, Malaysia; bm-1621@moe-dl.edu.my*, bm-1466@moe-dl.edu.my*
[2]*Unit Kimia, Kolej Matrikulasi Perak, Perak, Malaysia; bm-1444@moe-dl.edu.my*
[3]*Unit Bahasa Inggeris, Kolej Matrikulasi Perak, Perak, Malaysia; bm-1473@moe-dl.edu.my, bm-1594@moe-dl.edu.my*
[4]*Anbar Technical Institute, Middle Technical University, Al Anbar, Iraq. sulaima699n@gmail.com*

Received: 31 October 2023; Revised: 21 November 2023; Accepted: 19 December 2023; Published: 29 December 2023

**corresponding author*

Abstract

This research examines the transition from Scratch to Java programming, highlighting high achievement among participants. The study includes 70 students and utilizes mixed methods, combining quantitative analysis and qualitative narratives. Findings demonstrate the effectiveness of Scratch Visual Teaching Aids (VTA) in developing educational games, understanding algorithm concepts, and improving programming projects. Students' attitudes towards Scratch VTA show increased interest, concentration, and enjoyment in learning. Quantitative analysis indicates positive tool capabilities (mean scores ranging from 3.25 to 3.46). Qualitative narratives reveal varying perceptions of the transition, with prior programming knowledge influencing the shift. Challenges include adapting to the learning session and mastering Java's syntax and concepts. This study enhances programming education practices, providing recommendations for educators during the Scratch-to-Java transition.

Keywords: programming education, algorithm, Scratch, Java, visual teaching aid

INTRODUCTION

The transition from Scratch to Java programming represents a significant shift in students' learning processes within the field of computer programming. Scratch, a visual-based programming environment, allows students to explore programming concepts through a block-based interface. In contrast, Java programming requires students to engage with text-based coding, utilizing syntax, classes, and functions. This transition poses both opportunities and challenges for students as they navigate new programming concepts and languages (Kao et al., 2022).

Understanding the learning processes that students undergo during this transition is crucial for educators and researchers to develop effective instructional strategies and support mechanisms. By investigating the impact of transitioning from Scratch to Java programming, educational stakeholders can gain insights into the students' achievements, the role of prior programming knowledge, and the challenges they face during this transition. Furthermore, exploring the strategies and approaches students employ to overcome these challenges can contribute to the development of targeted interventions and scaffolding techniques (Kaya et al., 2021).

While previous studies have examined aspects of programming education and transitions between programming languages (Oka Kurniawan et al., 2021; Akkaya & Akpinar, 2022; Allen et al., 2022), limited research has specifically focused on the process of transitioning from Scratch to Java programming. This research aims to fill this gap by providing a comprehensive investigation into students' learning processes during this transition. By examining the interplay between prior programming knowledge, challenges encountered, and strategies employed, this study seeks to shed light on the factors that contribute to successful learning outcomes and inform educational practices.

Problem Statement

The transition from Scratch to Java programming poses challenges for students, and understanding their learning processes during this transition is vital for effective pedagogical interventions (Kao et al., 2022). While Scratch provides a visual-based programming environment, Java programming requires a shift towards text-based programming. This fundamental shift in programming paradigms can hinder students' learning experiences and impede their progress. Additionally, limited research exists on the specific learning processes involved in transitioning from Scratch to Java programming (Paiva et al., 2022).

According to a study by Mohd Adnan and Hamid (2020), achieving strong academic performance is essential for matriculation students because it expands their opportunities in selecting fields for further studies at the Degree level. However, insufficient proficiency in programming problem-solving skills becomes a hurdle for students, impeding their ability to attain favourable outcomes (Mohd Adnan & Hamid, 2020). Another study by Hai Hom et al. (2020) also highlights that impressive academic achievements play a significant role in shaping the path of matriculation students and influencing their decisions for higher education and prospects. Nevertheless, a lack of problem-solving skills in programming subjects poses a challenge in making well-informed decisions.

In a study conducted by Mohd Yusof (2017), well-planned instructional strategies, incorporating effective techniques and appropriate technology integration, are believed to profoundly affect students' learning outcomes. This approach not only contributes to the overall development of students but also provides many opportunities for active participation in the learning process. The use of suitable media is considered beneficial for educators in organizing and delivering instruction systematically and efficiently. However, meticulous planning is essential to ensure desired results, not only in terms of the acceptance of the instructional process but also in fulfilling the instructional objectives.

Therefore, there is a need to investigate and gain insights into the students' learning processes during this transition, including the impact of prior programming knowledge, the challenges faced, and the strategies employed to overcome these challenges. This research aims to address these gaps in knowledge and provide valuable insights to inform educational practices and support students' successful transition from Scratch to Java programming. Furthermore, there is a necessity for the provision of a visual teaching aid to facilitate students' acquisition of programming skills.

Objectives and Research Questions

This research aims to achieve four main objectives:

- To substantiate the development of a visual teaching tool, Scratch Visual Teaching Aids (VTA) as an indispensable resource to assist students in their programming learning process.
- To understand the role of prior programming knowledge and experience in facilitating or hindering the transition to Java programming.
- To explore the challenges faced by students when transitioning from Scratch to Java programming.

The corresponding research questions delve into these objectives by examining the impact of the transition on students' achievement, the role of prior knowledge, and the specific challenges experienced:

- What is the impact and effectiveness of a visual teaching tool as a vital resource in supporting students' programming learning process?
- What role do prior programming knowledge and experience play in facilitating or hindering students' transition from Scratch to Java programming?
- What challenges do students encounter during the transition from Scratch to Java programming?

Conceptual Framework

Lev Vygotsky, a Russian psychologist, introduced the concept of the zone of proximal development (ZPD), which refers to the gap between an individual's current developmental level and their potential development with guidance from a more knowledgeable other (MKO) such as senior lecturers (Vygotsky & Cole, 1978).

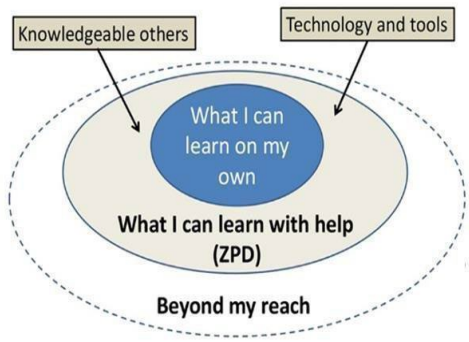

Figure 1: Vygotsky's zone of proximal development (McLeod, 2018)

The ZPD acknowledges learners' capacity for higher achievement when appropriately supported and challenged. It highlights the importance of recognizing learners' existing skills while fostering growth. By identifying and engaging students within their ZPD, educators can customize instruction to individual needs, providing the right balance of challenge and support for optimal learning (Vygotsky & Cole, 1978).

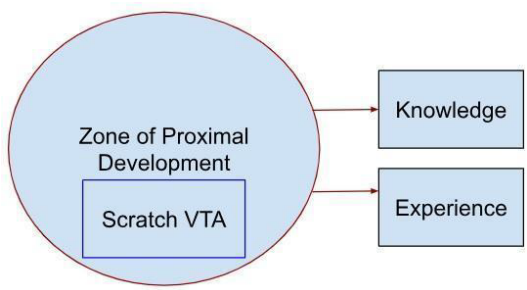

Figure 2: Conceptual framework

The utilization of Scratch VTA within the zone of proximal development can provide a promising framework for facilitating a successful transition from Scratch to Java programming, ultimately enhancing students' programming knowledge and experience.

Significance of Study

Incorporating Scratch VTA as an intervention in this research holds great value. It offers students a structured and interactive platform to navigate the transition from Scratch to Java programming. If tutorials provide too much detailed information, users may be able to follow instructions but can feel overwhelmed or bored (Basawapatna et al., 2019). By engaging in hands-on coding activities, students can develop their programming skills and computational thinking while bridging the gap between

visual-based programming (Scratch) and text-based programming (Java). The inclusion of this intervention allows researchers to assess its impact on learning outcomes, explore challenges faced by students, and inform instructional strategies for successful transitions in programming education.

Scope and Limitation

This research has several limitations that should be considered. Firstly, the sample size may be limited, which could affect the generalizability of the findings to a larger population. Secondly, the research is limited to students from a select few classes at a Malaysian pre-university. Factors such as prior programming knowledge, instructional approaches, or available resources may differ between institutions or educational systems. Additionally, time constraints may affect the depth of data collection or the ability to follow up with participants over an extended period.

LITERATURE REVIEW

Social constructivism lends credence to instructional pedagogy by defining the teacher's role in the teaching and learning process. This theory emphasizes the role of social interaction, collaboration, and cultural contexts in learning (Vygotsky, 1978). This suggests that teachers should employ learner-centred but teacher-guided teaching methods that offer appropriate scaffolding support such as the zone of proximal development (ZPD) and their potential development with guidance from a more knowledgeable other (MKO) such as educators at the right time to promote effective learning (McLeod, 2018). This support assists in the development of episodic memory, enabling students to remember and recall past events by using specific episodes or experiences as reference points. These episodes act as benchmarks, enhancing retention and recall, and ultimately contributing to the learning process (Kanno, 2018).

The transition from Scratch to Java programming represents a significant shift in students' learning processes within the field of computer programming. According to Brod (2021), the influence of prior knowledge on learning is contingent upon several determinants. Specifically, these determinants include the activation of prior knowledge, its relevance to the learning task, and its congruence with the new information. By examining these determinants, we can gain insights into the complex relationship between prior knowledge and learning success. Additionally, it is crucial to note that inter-individual and environmental factors were not considered in this analysis.

Students may encounter difficulties in grasping complex programming concepts, such as object-oriented programming and algorithmic thinking (Végh & Czakóová, 2023). Understanding these challenges is crucial for designing effective instructional approaches and providing appropriate support to students. Hence, creators and assessors of programming activities should take special care in crafting engaging elements that captivate students' attention and sustain their motivation, encouraging them to continue programming even after completing the activity instructions (Repenning & Basawapatna, 2016). Visual and graphical information is easier to understand and remember than written words

(Strömbäck et. al., 2022). It presents complex data concisely, supported by reliable statistics, without burdening the reader with lengthy explanations (Parveen & Husain, 2021). According to Mohd Yusof (2017), interactive teaching aids have been found to enhance students' understanding of programming concepts and subsequently improve their achievements in programming to optimize programming instruction; it is advisable to incorporate these teaching aids into the learning process.

Educators engaged in teaching programming subjects, particularly to first-year students, often encounter difficulties in molding students' comprehension of programming concepts, leading to an impact on their problem-solving skills (Mohd Yusof, 2017). Many first-year computer science students perceive programming skills as intricate and challenging to acquire. In addition to instructing programming concepts, educators have made efforts to instill essential skills like critical thinking, analytical thinking, and problem-solving in students aspiring to pursue a programming career. There is a greater focus on syntactic, semantic, and pragmatic aspects. Despite students being able to identify syntax, semantics, and errors in flowcharts or pseudo code, they often commit logical errors due to a weak understanding and cognitive skills. Mohd Yusof, Kohlit, et al. (2018) reported that many students find programming to be a challenging and difficult subject to learn and comprehend. This challenge stems from the difficulty students face in understanding the logic of solving each programming problem. The concepts of algorithms are particularly challenging to grasp without sufficient practice (Mohd Yusof et al., 2022).

Ismail et al. (2018) highlighted that problem-solving is a cognitive skill essential for students, particularly in analyzing problem scenarios and devising effective solutions. Students need to comprehend problems, design potential solutions, and construct and implement codes during the learning process (Ismail et al., 2018). In Mohd Yusof, Mohd Rufin et al. (2018), basic mathematical skills, especially arithmetic, are deemed crucial for solving mathematical problems. Researchers have established a correlation between fundamental mathematical knowledge and programming fundamentals (Mohd Yusof, Mohd Rufin, et al., 2018; Mohd Yusof et al., 2021). Programming necessitates proficiency in problem-solving, and students with weak mathematical abilities encounter similar challenges in learning programming (Mohd Yusof, Mohd Rufin, et al., 2018).

According to Mohd Adnan and Hamid (2020), a more structured approach to algorithm development can enhance the effectiveness of acquired knowledge. Algorithm development is fundamental in programming, and students should be exposed to basics that are easy to understand and engaging, preventing the perception that programming is overly difficult (Mohd Adnan & Hamid, 2020). Similarly, Hai Hom (2020) affirmed that programming problem-solving involves a cognitive process for problem-solving skills in computer science subjects, starting with reading and understanding the questions, planning the solution, and writing the program to address it.

Incorporating gamification into the learning process was explored in a study by Papadakis and Kalogiannakis (2019), focusing on its impact on students' academic performance and motivation. The findings offer valuable insights for the design of game-based learning experiences. The use of Classcraft in the educational experiment, as discussed by Papadakis and Kalogiannakis (2019), proved

interesting and partially achieved the study's objectives, particularly in motivating learning and encouraging participation through gamified elements. The study emphasizes the significance of motivation and engagement in education, as these factors are deemed essential for task completion and the development of specific behaviours. Despite programming having been studied for over half a century, the confusion between computer science and everyday computer use persists, as highlighted in the research by Papadakis and Kalogiannakis (2019). The study underscores the importance of addressing these challenges in the educational landscape.

METHODOLOGY

A suitable methodology for this research would be a mixed methods approach, which combines qualitative and quantitative data collection and analysis. This approach allows for a comprehensive understanding of the research topic by integrating qualitative narratives, quotes, and descriptions to capture the complexities of the phenomenon, and quantitative data in the form of pre and post-test scores to provide a numerical dimension. The qualitative data were analysed using thematic analysis, a qualitative method, while the quantitative data were analysed using PSPP statistical techniques to examine the differences in pre and post-test scores. By combining both types of data, the research can provide a more robust and holistic understanding of the impact of Scratch VTA intervention. A group of 70 students enrolled in a two-year program at a Malaysian pre-university from the 2021-2023 batch participated in this study. These students, representing seven classes, were chosen using stratified random sampling. The selection criteria were based on their enrolment in classes during their final semester (4th semester). The Scratch VTA refers to the revised edition of teaching manuals that underwent validation by two experts from Pahang Matriculation College and Sultan Idris Education University. 210 projects were created in the second semester during practical classes that took about two (2) hours once a week for 14 weeks. A set of questionnaires were given at the end of the second semester while a structured interview was conducted at the end of the fourth semester. Both questionnaire and interview data were collected through Google Forms. Four (4) lecturers from a Malaysian pre-university validated the questionnaires and interview questions. Figure 3 shows a research timeline diagram for this research. The terminal shape marks the start and end of this research while the diamond shape shows the action of using Scratch VTA. The parallelograms show input (70 students) and output (210 projects). Lastly, the rectangles indicate the process of giving questionnaires and interviews.

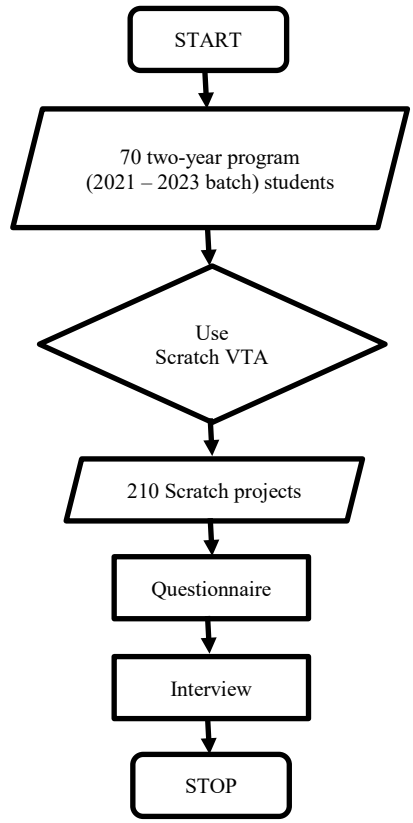

Figure 3: Flow of research

FINDINGS

This research collected both qualitative and quantitative data to gain a comprehensive understanding of the topic. The qualitative data included detailed narratives, quotes, and descriptions, while the quantitative data consisted of scores from questionnaires. The combination of both types of data provided a holistic perspective, blending qualitative richness with quantitative insights.

The online questionnaire was built using Google Forms. It consists of three sections: Section A - demographic information (items no. 1 to 4), Section B – tool capability (items no. 5 to 8), and Section C - student attitude (items no. 9 to 13). Section A includes race (item no.1), gender (item no.2), program module (item no.3), and tutorial class (item no.4). The item distribution and item code for Section B and Section C are presented in Table 2. This questionnaire utilized a 4-point Likert scale to get specific responses (on agreement) without the 'neutral' option. The scale ranges from 1 (totally disagree), 2 (totally agree), 3 (agree), and 4 *(*totally agree). The reliability statistic for this questionnaire is r=0.8 so it is proven that it is well constructed and can be used to gauge the respondents' opinions.

According to Section A, the study included a total of 70 students, comprising 35 students from Module 3 classes and 35 students from Module 4 classes. Out of the total participants, 24 were male (34.29%) and 46 were female (65.71%). Furthermore, 36 participants belonged to Module 2 (51.43%), while the remaining 34 participants were from Module 3 (48.57%).

Table 1: Questionnaire items and description (Sections B and C)

Section	Item code	Description	Mean	SD
B TOOL CAPABILITY	B1	Scratch VTA helped me achieve my objective of developing an educational game.	3.25	.47
	B2	Blocks with different colour schemes in Scratch VTA assist me in determining different algorithm concepts.	3.46	.50
	B3	Scratch VTA is suitable for my game development project.	3.35	.51
	B4	Scratch VTA is easy to use.	3.36	.54
C STUDENT ATTITUDE	C1	Scratch VTA makes me interested in learning algorithms programming).	3.33	.56
	C2	Scratch VTA motivates me to concentrate on working on my project.	3.28	.51
	C3	Scratch VTA helps to improve my understanding of different algorithm concepts.	3.35	.53
	C4	Scratch VTA promotes interaction between my lecturer and me.	3.28	.51
	C5	I enjoy learning using Scratch VTA.	3.38	.54

Table 1 demonstrates the tool capability of students (respondents) that helps prepare them to transition from Scratch to Java programming. Students who found that Scratch VTA helped them to develop educational games had a mean of 3.25 (SD=.47), while those who found blocks with different colour schemes in Scratch VTA assisted them in determining different algorithm concepts had a mean of 3.46 (SD=.50). Additionally, the mean for Scratch VTA found to be suitable for a game development project is 3.35 (SD=.51), whereas the mean for Scratch VTA easy to be used is 3.36 (SD=.54). Further examination on students' attitude showed that Scratch VTA makes them interested in learning algorithms(programming) with a mean of 3.33 (SD=.56), while Scratch VTA motivates them to concentrate on their projects with a mean of 3.28 (SD=.51). A mean of 3.35 (SD=.53) exemplified that Scratch VTA improves understanding of different algorithm concepts. It is also found that Scratch VTA promotes interaction between students and their lecturer with a mean of 3.28 (SD=.51) and students enjoy learning using Scratch VTA with a mean of 3.38 (SD=.54).

Table 2 outlines a structured interview consisting of five pivotal questions that delve into participants' initial expectations, encountered challenges, and reflections on the learning process during their transition from Scratch 3.0 to Java programming. Participants openly shared their motivations and anticipated outcomes as they navigated this shift in programming languages. The interview thoughtfully explored instances of challenges and difficulties faced during the transition, offering valuable insights into the obstacles encountered along their learning journey. Moreover, participants engaged in discussions regarding perceived similarities and differences between Scratch 3.0 and Java programming, encompassing concepts, syntax, and problem-solving approaches. The evaluation of participants' confidence levels in Java programming skills, in comparison to the initial stages of transition, provided noteworthy insights, with participants attributing changes in confidence to various

factors. The interview concluded by encouraging participants to reflect on their transition experience, providing an opportunity for them to share insights into aspects they wished they had known or done differently.

Table 2: Structured interview questions

No	Question
1	How would you describe your initial expectations and goals when you started learning Java programming after using Scratch 3.0?
2	What challenges or difficulties did you encounter during the transition process? Can you describe specific instances or situations that posed challenges to your learning?
3	What were the similarities and differences you noticed between Scratch 3.0 and Java programming in terms of concepts, syntax, and problem-solving approaches?
4	How confident do you feel about your Java programming skills now compared to when you first started the transition process? What factors do you attribute to this change in confidence?
5	Looking back, is there anything you wish you had known or done differently during the transition from Scratch 3.0 to Java programming?

Table 3 highlights the qualitative narratives and descriptions of the 15 respondents on the impact of the transition on Java programming, the role of the prior programming language (Scratch), and challenges during the transition from Scratch to Java programming. Generally, some respondents indicated that the transition from Scratch to Java is difficult; while others expressed that, it is not very difficult. In contrast, some others found it to be either scary, useful, exciting, or boring. As for the role of the prior programming language (Scratch), switching from Scratch's visual programming interface to Java's text-based syntax was difficult initially because writing error-free code demanded attention to detail and a solid understanding of Java's syntax rules. In addition, students were able to handle simple coding tasks independently but they relied on tutorials for the more challenging ones. Students who learned Scratch were more confident and found the transition to Java easier.

Finally, the responses to challenges and difficulties during the transition (from Scratch to Java) varied from difficulty in adapting to the learning session to the fact that it took time to internalize the precise syntax, including the proper placement of parentheses, semicolons, and curly braces. The other challenge faced was confusion with reserved words, where students forgot to include symbols, resulting in the Java program not running properly. Overall, the main challenge in Java programming compared to Scratch was the need for more critical thinking and decision-making, whereas, in Scratch, students only have to select the complete flow. In Java, students had to learn different flow patterns for various control structures.

Table 3: Thematic analysis of interview data

No	ID	Impact of transition on Java	Role of prior programming language (Scratch)
1	AB1A	It is hard.	Switching from Scratch's visual programming interface to Java's text-based syntax posed initial difficulties. It took time to learn the precise syntax and writing error-free code demanded attention to detail and a solid understanding of Java's syntax rules.

Table 3: Thematic analysis of interview data

No	ID	Impact of transition on Java	Role of prior programming language (Scratch)
2	AB1B	(I am) excited.	Initially, I lacked confidence when transitioning to learning Java programming. However, through dedicated time and effort, my confidence grew over time.
3	AB2A	I believe I will improve my Java skills because I already have a foundation in programming from learning Scratch in the previous semester.	On a scale of 1 to 10, I would rate myself at a 7 because the use of Java has indirectly helped me gain a deeper understanding of programming compared to my initial experience with Scratch.
4	AB2B	It would be a little bit difficult because I would be confused.	I can handle simple coding tasks independently, but for more challenging ones, I still rely on tutorials for guidance around 60% of the time.
5	AF1A	I expect to enhance my Java skills due to my prior experience with Scratch programming during the previous semester.	As of now, I am more confident with my Java programming skills compared to when I first started with Scratch.
6	AF1B	Hard.	It is easier because I have learned Scratch.
7	AF1C	Normal.	7.5/10.
8	HB1A	The structured process in Scratch makes it easier to use Java.	In the past, I felt less confident because I feared making numerous errors, which would lead to stress. Now, I feel more confident.
9	HB1B	It is easy to understand.	Not very confident.
10	HB2A	The usage of Scratch 3.0 is easier.	Slightly increased.
11	HF1A	Java is boring.	70% more confident.
12	HB2B	Very useful.	Very good
13	HF1B	I thought I would not ace it at first.	I barely remember the simplest coding before. Now I can do it without anyone's help.
14	HB2C	I was a bit scared because Java was harder than Scratch.	8/10.
15	HF1C	I'd be more adept at using it	6 out of 10.

DISCUSSION AND RECOMMENDATION

The research objectives were successfully addressed through the findings. The Scratch VTA was found to be an effective resource for assisting students in their programming learning process. It positively affected students' programming capabilities, improving their understanding of algorithm concepts and game development. The transition from Scratch to Java programming was influenced by students' prior programming knowledge, with those familiar with Scratch finding the transition easier. Challenges during the transition included adapting to the learning session, understanding Java's syntax, and the need for critical thinking. Based on the research findings, the following recommendations are proposed to optimize the learning experiences and outcomes for students transitioning from Scratch to Java programming, (i) incorporate interactive visual teaching aids like Scratch VTA to enhance programming education and support students' understanding of programming concepts, (ii) encourage students to acquire foundational programming skills through Scratch before transitioning to advanced languages, facilitating a smoother transition, (iii) provide targeted support to help students overcome challenges during the transition, including adapting to Java's syntax and developing critical thinking skills, and (iv) offer continuous guidance and mentorship from experienced instructors or peers to assist students in overcoming challenges and building confidence during the transition. By

implementing these recommendations, students can benefit from enhanced teaching methods, smoother transitions, targeted support, practical application of skills, and ongoing mentorship.

CONCLUSION

The research study's conclusion supports the objectives and research questions. The findings confirm the value of Scratch VTA in assisting students with programming learning (objective 1). The qualitative narratives shed light on the challenges faced during the transition, emphasizing the role of prior programming knowledge and directly relating to objective 2. These findings also align with objective 3, which explores the challenges faced by students transitioning from Scratch to Java programming. The narratives highlight difficulties in adapting to the learning session, internalizing Java's syntax, and grappling with reserved words. The need for critical thinking and decision-making in Java programming, compared to the more straightforward flow-based approach in Scratch, emerges as a significant challenge. These insights provide valuable information on the specific difficulties encountered during the transition, informing educational practices to better support students in overcoming these challenges.

ACKNOWLEDGMENTS

This work was supported by the Kolej Matrikulasi Perak, Perak, Malaysia.

CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

DECLARATION OF GENERATIVE AI

During the preparation of this work, the authors used Gen AI tools to enhance the clarity of the writing. After using the tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

DATA AVAILABILITY STATEMENT

Data available within the article or its supplementary materials.

REFERENCES

Akkaya, A., & Akpinar, Y. (2022). Experiential serious-game design for the development of knowledge of object-oriented programming and computational thinking skills. *Computer Science Education*, *32*(4), 476-501.

Ismail, A., Mohd Yusof, S.S. & Ubaidullah, N. H. (2018). The impact of using visual programming environment towards college students' achievement and understanding in programming. *The International Journal of Multimedia & Its Applications*. https://doi.org/10.5121/ijma.2018.10606.

Allen, O., Downs, X., Varoy, E., Luxton-Reilly, A., & Giacaman, N. (2022). Block-based object-oriented programming. *IEEE Transactions on Learning Technologies*, *15*(4), 439-453.

Basawapatna, A., Repenning, A., & Savignano, M. (2019). The zones of proximal flow tutorial: designing computational thinking cliffhangers. *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*, 428-434. https://doi.org/10.1145/3287324.3287361.

Brod, G. (2021). Toward an understanding of when prior knowledge helps or hinders learning. *Science of Learning*, *6*(1), 24. https://doi.org/10.1038/s41539-021-00103-w.

Kanno, T. N., & Nzewi, U. M. (2018). *Issues in curriculum development and implementation in Nigeria*. Lagos: Foremost Educational Services Ltd.

Kaya, Z., Kaya, O. N., Aydemir, S., & Ebenezer, J. (2021). Knowledge of student learning difficulties as a plausible conceptual change pathway between content knowledge and pedagogical content knowledge. *Research in Science Education*, 1-33.

Kao, Y., Matlen, B., & Weintrop, D. (2022). From one language to the next: Applications of analogical transfer for programming education. *ACM Transactions on Computing Education*, *22*(4), 1-21.

McLeod, S. (2018). Vygotsky. https://www.simplypsychology.org/vygotsky.html

Oka Kurniawan, Jégourel, C., Lee, N. T. S., De Mari, M., & Poskitt, C. M. (2021). Steps before syntax: Helping novice programmers solve problems using the PCDIT framework. *arXiv*:2109.08896.

Paiva, J. C., Leal, J. P., & Figueira, Á. (2022). Automated assessment in computer science education: A state-of-the-art review. *ACM Transactions on Computing Education, 22*(3), 1-40.

Papadakis, S., & Kalogiannakis, M. (2019). Evaluating the effectiveness of a game-based learning approach in modifying students' behavioral outcomes and competence, in an introductory programming course: A case study in Greece. *International Journal of Teaching and Case Studies*, *10*(3), 235-250.

Parveen, A., & Husain, N. (2021). Infographics as a promising tool for teaching and learning. *Journal of Emerging Technologies and Innovative Research*, *8*(8), 554-559.

Repenning, A., & Basawapatna, A. (2016). Drops and kinks: Modeling the retention of flow for hour of code style tutorials. *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*, 76–79. https://doi.org/10.1145/2978249.2978260.

Hai Hom, S. N., Ibrahim, H. H., Ibrahim, A., Mokhsin, M., & Abdul Talib, C. (2020) Assessment of using EZ-Prog: An easy color schematic model for programming problem solving. *ASEAN Journal of Teaching and Learning in Higher Education*, *12*(1). 31-41. https://ejournal.ukm.my/ajtlhe/article/view/39206

Mohd Yusof, S. S. (2017). *Keberkesanan penggunaan alat bantu mengajar interaktif bagi mengurangkan kesalahfahaman konsep pengaturcaraan dalam kalangan pelajar*. [Master dissertation, Universiti Pendidikan Sultan Idris]. UPSI Library. https://ir.upsi.edu.my/detailsg.php?det=3350.

Mohd Yusof, S. S., Ilias, K., Jabar, J., Ahmad Bakary, S. & Syed Nordin, S. A. (2021). Aplikasi alat pengarang grafik flowgorithm dalam meningkatkan kefahaman dan minat dalam pembelajaran algoritma pengaturcaraan pelajar matrikulasi. *Jurnal Penyelidikan Dedikasi*, *19*(2), 62-81.

Mohd Yusof, S. S., Ismail, A. & Abdul Aziz, N. A. (2022). Learning algorithm concepts by developing them in educational gameplay: From the perspective of college students in Perak. *Journal of Information and Communication Technology in Education*, *9*(3), 30–40. https://doi.org/10.37134/jictie.vol9.sp.1.3.2022.

Mohd Yusof, S. S., Kohlit, M., Maarof, F., & Abu Bakar, A. Z. (2018). Keberkesanan penggunaan alat bantu mengajar interaktif dalam pengajaran dan pembelajaran asas pengaturcaraan. *Jurnal Penyelidikan Dedikasi*, *15*, 62-79.

Mohd Yusof, S. S., Mohd Rufin, S. M., Zohedi, A. K., & Ng, C. H. (2018). Tinjauan penggunaan bahasa pengaturcaraan secara visual bagi mengurangkan kesalahfahaman konsep pengaturcaraan dalam kalangan pelajar. *Jurnal Penyelidikan Dedikasi*, *14*, 150-163.

Strömbäck, F., Mannila, L., & Kamkar, M. (2022). A weak memory model in Progvis: Verification and improved accuracy of visualizations of concurrent programs to aid student learning. *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research*, 1-12.

Végh, L., & Czakóová, K. (2023). Possibilities of using games in teaching and learning the basic concepts of object-oriented programming. *Proceedings of the 17th International Technology, Education and Development Conference*, 5329-5334. https://doi.org/10.21125/inted.2023.1383

Vygotsky, L. S., & Cole, M. (1978). *Mind in society: Development of higher psychological processes*. Harvard University Press.

Mohd Adnan, Z. & Hamid, J. (2020). Kesan model Polya dalam pembelajaran algoritma dalam kalangan pelajar: Satu kajian kes. *Journal of Humanities and Social Sciences*, *2*(3), 88-92. https://doi.org/10.36079/lamintang.jhass-0203.140.