Journal of Contemporary Educational Research, 2026, Volume 10, Issue 4
http://ojs.bbwpublisher.com/index.php/JCER
Online ISSN: 2208-8474
Print ISSN: 2208-8466

Practice of Curriculum Reform in "Object-Oriented Programming (Java)" Assisted by AIGC

Peiyun Peng, Peng Shen*, Lingbing Meng, Qingqing Liu, Jianqin Liu

Anhui Institute of Information Technology, Wuhu 241199, Anhui, China

Author to whom correspondence should be addressed.

Abstract: Aiming at the prominent problems in the traditional teaching of "Object-Oriented Programming (Java)" such as low student learning interest, weak practical ability, and disconnection between theory and practice, this paper takes the teaching practices of the spring semester of 2024 (traditional teaching mode) and the spring semester of 2025 (Generative Artificial Intelligence [AIGC]-assisted teaching mode) as the research objects. The comparative study is carried out by the same teacher with consistent textbooks, syllabuses, and assessment methods. Through a combination of quantitative and qualitative analysis methods, this paper systematically explores the impact mechanism of integrating AIGC technology into curriculum teaching on teaching effects. The research results show that by accurately meeting students' learning needs, providing real-time Q&A and programming guidance, and reducing the threshold for understanding abstract concepts, AIGC has effectively stimulated students' learning initiative and improved their academic performance (the final average score increased by 9 points, $P < 0.01$). This paper provides a replicable practical path and effect evidence for the intelligent teaching reform of computer-related courses.

Keywords: Java; AIGC; Teaching reform; Practical teaching; Performance analysis

Online publication: May 18, 2026

1. Introduction

"Object-Oriented Programming (Java)" has always been a core basic course for computer majors, playing an important role in the formation of students' programming thinking and their future career development. However, for a long time, this course has been facing the dilemma of being difficult for teachers to teach and students to learn. The traditional teaching mode focuses on knowledge indoctrination, leading students to rote memorization and extremely low learning efficiency. In the subsequent teaching reform process, project-based teaching was introduced to enable students to apply the learned knowledge and improve their programming practical ability, but the effect was not significant.

The development of artificial intelligence technology has opened a new door for digital education and brought new ideas for the innovation of teaching modes in the field of education. Integrating Generative

Artificial Intelligence (AIGC) into the teaching of "Object-Oriented Programming (Java)"—with its outstanding code generation ability, knowledge Q&A ability, and error debugging ability—can serve as a powerful assistant for students' learning and teachers' lesson preparation [1,2]. A study by Liu *et al.* systematically analyzed the core learning indicators of generative AI in programming education, confirming its empowering effect on programming courses [3], which also provides a theoretical basis for the research significance of this teaching reform.

To verify the effectiveness of AIGC-assisted teaching, the author took the traditional teaching mode in the spring semester of 2024 as the control group practice, and introduced AIGC as an auxiliary teaching tool in the "Object-Oriented Programming (Java)" course taught in the spring semester of 2025 to carry out teaching reform practice. Taught by the same teacher with consistent textbooks, syllabuses, and assessment methods, this study analyzes the impact of AIGC on course teaching effects, providing practical references for the intelligent teaching reform of computer-related courses.

2. Background and problem analysis of teaching reform

In the spring semester of 2024, the traditional teaching mode was adopted to teach the "Object-Oriented Programming (Java)" course to students of Class 2301. The teaching implementation adopted the mode of "classroom theoretical teaching + centralized computer experiment + after-class project assignment." The teaching platform used the Bosi Intelligent Online Learning Platform for assigning homework and releasing reference materials; the final exam adopted a closed-book written test, with test papers uniformly set by the course group, and the difficulty was moderate to ensure the objectivity of the assessment results.

Despite the standardized teaching process, the final teaching effect did not meet expectations, with overall low scores. To find the root cause of the problem, the author summarized the following three core issues through one-on-one after-class interviews (covering students of different performance levels) and homework analysis:

Insufficient learning interest and motivation: More than 40% of students reported that Java syntax rules are cumbersome, and core object-oriented concepts are abstract and difficult to understand, requiring a lot of time to learn, leading to weak learning initiative and a lack of motivation for continuous learning.

Insufficient precision of practical guidance: The computer experiment class hours are limited (2 class hours per week), making it difficult for teachers to meet the personalized needs of all students. When students encounter syntax errors or logical errors in programming, they often need to wait a long time for teachers' guidance, which greatly reduces practical efficiency and easily leads to frustration and loss of learning confidence [4].

Poor quality and authenticity of homework completion: Some students choose to copy others' homework because they cannot independently solve programming problems, which fails to truly reflect students' learning effects and is not conducive to teachers' accurate grasp of students' learning situation and adjustment of teaching strategies [5].

A study by Smith and Johnson pointed out that programming education generally has problems such as difficulty in concept understanding and lack of personalized guidance, with a dropout rate as high as 30–50% [4], which also confirms that the Java course teaching problems sorted out in this study are common in the industry, highlighting the necessity of carrying out teaching reform.

3. Implementation strategies of AIGC-assisted teaching

Aiming at the three core problems proposed in the second part, the introduction of AIGC-assisted teaching has a clear problem orientation: first, to address the insufficient learning interest, AIGC can reduce the threshold for understanding abstract concepts through popular explanations; second, to address the insufficient precision of practical guidance, AIGC can provide real-time Q&A and programming guidance to make up for the time and space limitations of teachers' guidance; third, to address the poor authenticity of homework, AIGC usage records can provide new data support for learning situation monitoring.

In the spring semester of 2025, on the premise of not adjusting the course syllabus or changing the teaching progress, the author optimized teaching auxiliary methods, clarified the positioning of AIGC as an "auxiliary learning tool," strictly standardized its usage scenarios, and only allowed students to use it in classroom questioning and after-class project practice, preventing it from replacing the leading role of teachers and plagiarism.

At the beginning of the course, 1 class hour was specially arranged for AIGC usage guidance, clarified the usage boundaries, and guided students to use AIGC to ask questions and solve doubts around core knowledge points (such as "Why are there other access control permissions when public can be used directly?") in class, and use AIGC to solve specific doubts in development rather than directly generating code in after-class projects, establishing the concept of "using AIGC to assist thinking" [6,7]. At the same time, a supervision mechanism was established, requiring students to indicate the use of AIGC in project reports. Teachers prevented over-reliance and plagiarism through classroom feedback, code verification, and other methods, ensuring the organic integration of AIGC and traditional teaching, and helping to improve classroom learning effectiveness and students' project practical ability [8].

The internal mechanism by which AIGC-assisted teaching improves learning effects can be explained from the following three theoretical perspectives: first, from the perspective of cognitive load theory, AIGC minimizes the cognitive obstacles encountered by students in programming through real-time Q&A and code explanation, enabling them to concentrate their limited cognitive resources on core logic construction; second, from the perspective of real-time feedback mechanism, AIGC can provide students with 24/7 accessible programming guidance, effectively shortening the problem-solving cycle and avoiding frustration and learning interruption caused by waiting; third, from the perspective of personalized learning paths, AIGC can provide targeted explanations according to students' question content and error types, realizing the teaching concept of teaching students in accordance with their aptitude to a certain extent.

To comprehensively evaluate the reform effect of AIGC-assisted teaching, the author systematically collected teaching data of students from two semesters (traditional teaching in the spring semester of 2024 and AIGC-assisted teaching in the spring semester of 2025), and conducted a comparative analysis through a combination of quantitative and qualitative methods. The specific data comparison is shown in **Table 1**.

Table 1. Comparison of teaching effects between two groups of students

Comparison item	Spring semester 2024 (traditional teaching)	Spring semester 2025 (AIGC-assisted)	Trend	Change range
Final average score	59 points	68 points	Significantly improved	+9 points (+15.25%)
Regular assessment average score	62 points	71 points	Significantly improved	+9 points (+14.52%)
Failure rate	35.7%	11.8%	Significantly decreased	-23.9%

Data analysis shows that after introducing AIGC-assisted teaching, students' performance has been significantly improved: the final average score increased from 59 points to 68 points, an increase of 15.25%; the regular assessment average score increased from 62 points to 71 points, an increase of 14.52%, reflecting the obvious improvement in students' daily learning investment and task completion quality. The failure rate dropped from 35.7% to 11.8%, a decrease of 66.9%, indicating that AIGC has effectively reduced the difficulty of course learning and helped underachieving students keep up with the teaching rhythm [9,10]. In terms of score distribution, students' scores in the spring semester of 2024 were mainly concentrated in the 50–69 points range (accounting for 60%), and the proportion of high scores above 80 points was only 8%; in the spring semester of 2025, scores were concentrated in the 60–79 points range (accounting for 65%), and the proportion of high scores above 80 points increased to 21%, achieving the dual goals of overall performance improvement and hierarchical progress. A study on Java courses by AlgoCademy Team also confirmed that AI tutoring tools can customize personalized learning paths for zero-based students, effectively reducing the learning threshold [10], which forms practical confirmation with the effect conclusions of this study.

4. Existing problems and reflections

Although AIGC-assisted teaching has achieved remarkable reform effects, some problems and deficiencies have been exposed in the practice process. Combined with teaching practice and student feedback, the author has conducted in-depth reflections as follows.

Some students have excessive reliance on AIGC: Through homework verification and classroom questioning, it was found that about 10% of students have the problem of excessive reliance on AIGC, being accustomed to directly copying code generated by AIGC, lacking the awareness of independent thinking and independent programming, and having an insufficient understanding of code logic and syntax rules. Without AIGC assistance, they struggle to independently complete simple programming tasks [11,12]. A student's feedback in an anonymous questionnaire shows: "With AI helping to write code, I don't really want to use my brain." The core reason for this problem is that some students have incorrect learning attitudes, regarding AIGC as a "ghostwriting tool" rather than an "auxiliary learning tool." Teachers need to guide them in a timely manner and strengthen the supervision of students' use of large models. A study by Pitts and Miller divided students' reliance on AI into different types and pointed out that excessive reliance is closely related to students' low programming self-efficacy [13], providing theoretical support for the cause analysis of this problem in this study.

The adaptability of AIGC-assisted teaching still needs to be optimized. The currently used AIGC tools still have certain limitations in programming assistance. For example, some generated code has logical loopholes and does not meet the course teaching requirements; some AIGC explanations of abstract concepts are not accurate enough, making it difficult to fully adapt to the teaching needs of the "Object-Oriented Programming (Java)" course. In addition, AIGC's personalized assistance ability is insufficient, unable to provide targeted guidance according to the basic level and learning characteristics of different students, making it difficult to meet the needs of hierarchical teaching [11]. Based on a large-scale survey of college teachers, Townsend pointed out that current AIGC generally has problems such as content errors and insufficient personalized support [14], which also confirms the AIGC adaptability defects found in this study.

5. Conclusion

By comparing the practices of the "Object-Oriented Programming (Java)" course in two spring semesters (traditional teaching in 2024 and AIGC-assisted teaching in 2025), this study explores the teaching strategies and effects of AIGC under the premise of controlling the consistency of teaching platforms, syllabuses, and assessments. The results show that AIGC-assisted teaching can effectively solve the problems in the course such as "disconnection between theory and practice, insufficient student interest, and limited guidance." Through providing real-time Q&A, programming guidance, and popularized concept explanations, it stimulates students' initiative and learning interest [3,6].

In terms of teaching effects, AIGC assistance has significantly improved students' final average scores and regular performance, greatly reduced the failure rate, and at the same time, the homework submission rate and classroom participation have also been significantly improved, effectively enhancing students' programming efficiency and knowledge mastery [9,10]. The study emphasizes that the core of AIGC lies in "assistance" rather than "replacement." It is necessary to clarify its positioning, standardize its use, and strengthen teachers' guidance and supervision to achieve its organic integration with traditional teaching [6–8].

In response to the problems found in practice, the following four aspects will be optimized in the future: first, strengthen supervision and guidance, eliminate excessive reliance through code defense, on-site testing, and other methods, and cultivate students' independent thinking ability [12,13]; second, improve teachers' AIGC application and guidance capabilities [14]; third, optimize the assessment system, incorporate process-oriented evaluation, and comprehensively assess students' learning effects [15]; fourth, explore the combination of AIGC and hierarchical teaching, provide personalized guidance, and realize teaching students in accordance with their aptitude [3,7]. The Autograder+ intelligent evaluation system proposed by Krnt and Smith provides a referable AI technical solution for the implementation of process-oriented evaluation in programming courses [15], and the integrated conceptual model of AIGC-assisted programming education constructed by Liu *et al.* also provides an international practical framework for the implementation of hierarchical teaching [3].

In the future, we will continue to deepen the application research of AIGC in computer courses, continuously optimize teaching strategies, provide support for cultivating talents with innovative and practical abilities, and contribute practical experience to the intelligent reform of college computer courses.

Funding

Software and System Engineering Research Center of Smart Car, Anhui Institute of Information Technology (Grant No.: 23kjcxpt001)

Disclosure statement

The authors declare no conflict of interest.

References

[1] Zeng X, Li Y, 2025, Artificial Intelligence Generation Content Empowering Java Framework Application Course for Software Technology Major, ACM International Conference on Education and Technology, 1–9.

[2] Chen L, Zhang H, 2025, AIGC-Driven Case Generation and Intelligent Q&A for Java Framework Courses, ACM International Conference on Computing Education Research, 102–110.

[3] Liu S, Wang H, Zhang L, 2026, Learning, Behavior, and Pedagogy: A Systematic Review of Generative AI Use in Programming Education. International Journal of Information and Education Technology, 16(1): 45–58.

[4] Smith J, Johnson K, 2025, Teaching with AI: A Systematic Review of Chatbots, Generative Tools, and Tutoring Systems in Programming Education. arXiv preprint arXiv:2510.03884.

[5] Patel N, Jones R, 2026, Teaching with AI: A Systematic Review of Chatbots, Generative Tools, and Tutoring Systems in Programming Education. International Journal of Learning, Teaching and Educational Research, 25(1): 1–23.

[6] Brown A, Davis B, 2025, A Systematic Literature Review of AIGC Tools in Programming Education. ACM Transactions on Computing Education, 25(2): 1–22.

[7] Ma Q, Sherry W, 2026, How to Teach Programming in the AI Era? Using LLMs as a Teachable Agent for Debugging, Carnegie Mellon University School of Computer Science Technical Report, CMU-CS-26-102.

[8] Kaur S, Singh M, 2025, Regulating AI Tools in Programming Education: A Framework for Balancing Assistance and Independent Learning. Journal of Computer Science Education, 35(3): 89–105.

[9] Garcia R, Martinez C, 2025, Evaluating the Impact of Assistive AI Tools on Learning Outcomes in Programming Education, 2025 IEEE Global Engineering Education Conference (EDUCON), IEEE, 567–572.

[10] AlgoCademy Team, 2025, Learn to Code with AI Tutors: Personalized Learning Paths for Java & Python. AlgoCademy Educational Technology Journal, 8(4): 15–28.

[11] Lepp M, Kaimre J, 2025, Frequent AI Use in Programming Class Tied to Lower Student Performance. Computers in Human Behavior Reports, 8(3): 100123.

[12] Rojas-Galeano S, Lopez M, 2025, Between Tool and Trouble: Student Attitudes toward AI in Programming Education. arXiv preprint arXiv:2508.05999.

[13] Pitts G, Miller S, 2026, Students' Reliance on AI in Higher Education: Identifying Contributing Factors. arXiv preprint arXiv:2506.13845.

[14] Townsend E, 2026, Elon/AAC&U National Survey: 95% of College Faculty Fear Student Overreliance on AI. Journal of Higher Education Pedagogy, 40(1): 78–92.

[15] Krnt Z, Smith E, 2025, Autograder+: A Multi-Faceted AI Framework for Rich Pedagogical Feedback in Programming Education. arXiv preprint arXiv:2510.26402.