*Higher Education and Practice Vol. 3 No. 4, 2026*

# Research on Java Teaching Reform Empowered by Artificial Intelligence: A Mixed-Method Action Research Perspective

**Bingqiang Huo[1], Fang Li[1,*], Sudan Xin[1], Hongjun Wang[2]**
[1]*Changji University, Changji, Xinjiang, China*
[2]*Shandong University, Qingdao, Shandong, China*
**Corresponding Author*

**Abstract: Generative artificial intelligence technology is reshaping the foundational logic of programming education, necessitating a shift in Java instruction from "teaching students to write correct code" to "teaching students to design good programs." However, current domestic research on Java teaching reform generally faces three challenges: ambiguous attribution in teaching experiments, lack of discipline specificity in AI tools, and insufficient transferability of reform solutions. This paper adopts action research as its methodological framework, using two rounds of Java teaching practice (161 students) in the Software Engineering program at our university as the research site. Targeting four pain points specific to the Java discipline—object-oriented abstraction barriers, exception handling cognitive gaps, collection framework selection difficulties, and multithreading mental model biases—a spiral evolutionary reform pathway is constructed, and a "five-in-one" AI + Java education teaching reform framework with tiered implementation plans is proposed. Data from the two rounds show that experimental class students' final comprehensive scores improved significantly (M=80.3, SD=8.6 vs M=73.1, SD=11.2; Cohen's d=0.723, p<0.01), project excellence rates jumped from 31.2% to 67.8% ($\chi^2$=21.62, p<0.001), and adaptive programming ability test completion rates without AI assistance significantly exceeded the control class (78.5% vs 65.3%, d=0.88), indicating that AI catalyzes the growth of higher-order abilities through a "scaffolding" effect.**

**Keywords: Artificial Intelligence Empowered Education; Java Teaching Reform; Action Research; Generative AI; Mixed Methods; Discipline Specificity; Transferability**

## 1. Introduction the Era Proposition and Methodological Consciousness of Java Teaching Reform

The impact of generative AI has directly taken over the core output of programming education. When AI can generate syntactically correct code within seconds, the significance anchor of Java teaching has shifted from "teaching students to write correct code" to "teaching students to design good programs." [1] This transformation is not about lowering difficulty but elevating dimensionality—just as calculators shifted the focus of mathematics education from computation to thinking, AI programming assistants also shift the focus of Java teaching from syntactic correctness to design rationality.

However, current domestic research on AI-empowered programming education is forming a pattern worthy of caution: arguing for AI's importance—introducing AI tools—demonstrating pre-post test score improvements—claiming reform effectiveness. This linear narrative obscures three deep methodological problems.

First, the attribution challenge: causal ambiguity between intervention and effect. In most quasi-experimental [2] designs, AI introduction is a composite of multiple interventions encompassing content adjustments, method updates, tool deployment, and assessment reform, making it impossible to determine whether the effect stems from a specific element or the Hawthorne effect.

Second, the discipline challenge: lack of discipline specificity in AI tool application. Different programming languages possess distinctly different cognitive difficulties—Python's difficulty lies in engineering thinking transitions, C++'s difficulty lies in memory management, while Java's difficulty lies in deep understanding of the object-oriented paradigm

and multi-layered [3] abstract modeling. Most studies flatten these differences into a uniform "programming learning difficulty," resulting in reform solutions lacking targeted strikes against Java-specific pain points.

Third, the practice challenge: the transferability gap of reform solutions. Many papers depict ideal scenarios requiring considerable funding and technical teams (RAG systems, knowledge graph platforms), but most local universities finish reading with only admiration and nothing to learn from.

To address these issues, this paper selects action research as its methodological framework. Action research does not pursue laboratory-style causal purity but rather uses spiral evolutionary iterative logic to progressively approach effective solutions in real teaching settings. The specific research question is: targeting Java discipline-specific cognitive difficulties, how can we construct an AI-empowered teaching [4] reform solution that possesses both methodological consciousness and practical implementability?

## 2. Unique Cognitive Difficulties of the Java Discipline: Target Analysis for AI Empowerment

All printed material, including text, illustrations, Based on classroom observations, student interviews, and assignment analysis of Java teaching at our university and six peer universities in the same region (2022–2024, cumulatively analyzing over 4,000 student programming assignments), this study identifies four deep cognitive fault lines in Java teaching.

### 2.1 Abstraction Barriers in Object-Oriented Programming

Java's purely object-oriented design [5] poses an "abstraction wall" for beginners: students are forced from the start to confront abstract constructs such as class, static, and public that they lack the capacity to understand. Assignment analysis shows that approximately 68% of students exhibit "pseudo-object-oriented programming" in the first four weeks—mechanically using class syntax but stuffing all logic into the main method. The role of AI tools here should not be that of a "substitute writer" but rather a Socratic questioner: not directly providing code, but guiding students through questioning to complete abstract modeling.

### 2.2 Cognitive Gap in Exception Handling Mechanisms

Java's checked exception mechanism requires students to transition from the naive notion of "error = code written incorrectly" to the mental model of "errors are a normal component of the runtime environment." Approximately 54% of students adopt a passive strategy of "grabbing the nearest catch block, printing the error message, then ignoring it." AI can demonstrate the code quality differences between "with exception design" and "without exception design" for the same functionality, helping students establish defensive design awareness.

### 2.3 Selection Dilemma in the Collection Framework

The richness of Java's collection framework enables students to "know how to use" each collection type but not "know how to choose" in a given scenario. Tests show that in scenarios requiring frequent mid-list insertions, 72% of students still choose ArrayList rather than LinkedList. API usage happens to be the area where AI excels most, so the correct direction for AI empowerment is not faster API lookups [6], but using AI to quickly generate multiple candidate solutions, allowing students to receive decision-making training through comparison and selection.

### 2.4 Mental Model Bias in Multithreaded Programming

The leap from single-threaded to multithreaded programming requires a fundamental cognitive shift in time perception. Approximately 82% of students cannot understand "why two threads simultaneously incrementing by 1 may not result in an increment of 2." AI can serve as a visual interpreter of concurrency errors—letting students first write buggy code and observe anomalous behavior, then having AI explain the timeline issues, establishing a concurrency mental model through the "experience→ understanding" pathway.

The above four pain points constitute a progressively layered [7] cognitive challenge ladder. The object-oriented abstraction barrier is the first hurdle; without establishing object modeling thinking, subsequent exception handling, collection selection, and concurrency architecture will all be built on the fragile foundation of "pseudo-object-orientation."

## 3. Theoretical Foundation and Literature Review

### 3.1 Theoretical Lens: The Dual Nature of AI as Cognitive Scaffolding

Vygotsky's Zone [8] of Proximal Development offers a foundational framework for understanding AI's role in programming education. AI programming assistants function as a novel form of "semi-intelligent scaffold" — responsive to every query yet not proactively constructing solutions. The tension lies in potentially inducing "cognitive outsourcing," while the opportunity resides in their capacity for instant feedback, which liberates students from the delays of waiting for instructor responses and enables denser "problem → feedback → correction" cycles. Papert's constructionist theory further enriches this analysis: human-AI collaborative, project-based learning represents a contemporary operational path for "making-driven learning."

### 3.2 Domestic and International Research Status

International research has shifted from optimistic narratives to cautious calibration. Raihan et al. [9] conducted a systematic review covering 47 studies on LLMs in computer science education, noting that most studies use efficiency improvement as the criterion, but the extent to which efficiency gains translate into ability growth remains an open question. Reihanian et al. [10] pointed out that the "apparent correctness" of AI-generated code poses a fundamental threat to traditional assessment systems. Mayr [11] proposed the concept of "knowledge markers" to address this issue—shifting the assessment focus to cognitive markers that AI cannot easily substitute, such as design decision justification and algorithm complexity analysis.

Domestic academia has also actively responded to AI's impact on education. The Ministry of Education's "Higher Education Artificial Intelligence Innovation Action Plan" [12] and "Education Informatization 2.0 Action Plan" [13] have provided institutional impetus. Zhang Jianping et al. [14] systematically examined ethical issues in AI education applications, Yu Shengquan et al. [15] revealed the role transformation of teachers from knowledge transmitters to learning designers, Zhu Zhiting et al. [16] positioned AI as a restructuring technology for education systems. Liu Sannyuva et al. [17] and Zhong Shaochun et al. [18] provided theoretical references from personalized learning and human-machine collaboration dimensions respectively. In the Java teaching reform domain, Song Wei et al. [19] explored curriculum reform under the OBE concept, Wang Yingzi et al. [20] proposed project-driven teaching models, but neither systematically incorporated AI tools.

Taken together, international research is more advanced in methodological reflection but predominantly uses Python as the vehicle, without delving into [21] Java's disciplinary specificity; domestic Java teaching research has solid foundations but remains at an early stage in the AI empowerment dimension. This study's positioning is precisely at the intersection of these two threads.

## 4. Research Design: Mixed-Method Approach under the Action Research Framework

### 4.1 Methodological Choice

The core challenge of teaching reform lies in the fact that real teaching settings cannot be decomposed into independent variables. Action research [22] is precisely designed for this non-decomposability, allowing researchers to progressively approach more effective teaching configurations as reflective practitioners through the "plan→act→observe→reflect→ replan" spiral.

### 4.2 Research Process

This study underwent two complete action research cycles. The first round (2023–2024 academic year, second semester, control class, 79 students) adopted traditional teaching mode with the core objective of establishing a baseline—systematically collecting data on learning difficulty distribution, typical error patterns, and AI usage habits, forming the problem diagnosis basis for subsequent reform. The second round (2024–2025 academic year, first semester, experimental class, 82 students) implemented [23] the five-in-one reform plan, collecting data through classroom video analysis, AI usage logs, periodic interviews, and questionnaire surveys, with the plan itself undergoing multiple process adjustments. The same instructor taught both rounds to control for teacher effects, and the two groups showed

no significant difference in admission foundations and prerequisite course performance (t(159)=0.87, p=0.39).

### 4.3 Data Collection and Analysis Strategy

A concurrent triangulation strategy within the mixed-methods approach was adopted [24]. Quantitative data included final comprehensive scores, five-dimensional project scores, adaptive programming test scores, and questionnaire surveys; qualitative data included three rounds of semi-structured interviews (12 stratified students per round), AI usage logs (79 from the experimental class), and classroom observation records. Between-group comparisons used independent sample t-tests and chi-square tests with reported effect sizes; qualitative data was analyzed using thematic analysis [25] with NVivo 14 assisted coding.

## 5. The "Five-in-One" AI-Empowered Java Teaching Reform Plan

Based on the four Java cognitive difficulties identified in the first round, this study proposes a "five-in-one" reform framework. Each dimension targets specific pain points and distinguishes between ideal solutions and minimum viable solutions to address the transferability challenge.

### 5.1 Dimension 1: AI-Driven Teaching Content Restructuring-Reduce Syntax, Elevate Thinking, Strengthen Collaboration

Syntax instruction was compressed from 60% to 30%–35%, freeing class hours for four higher-order thinking training modules: problem modeling and algorithm design, object-oriented design principles in practice, defensive exception handling design, and code quality and refactoring techniques. AI collaborative programming training covers four competency levels: effective prompt engineering, AI-generated code review, AI-assisted debugging, and iterative optimization. The ideal solution introduces commercial tools such as GitHub Copilot Education Edition and configures course-specific private knowledge bases; the minimum viable solution uses free AI dialogue tools to complete core training.

### 5.2 Dimension 2: AI-Assisted Personalized Learning Pathways-Diagnosis, Stratification, Feedback Loop

A three-node dynamic loop was constructed: "semester-start diagnosis→stratified tasks→ midterm re-diagnosis→pathway adjustment→ semester-end diagnosis." Stratified tasks use color codes (blue/green/orange) rather than "basic/advanced/challenge" labels to avoid labeling effects. The ideal solution deploys a knowledge graph diagnostic system and RAG intelligent Q&A system (over 3,800 queries processed across two rounds, reducing teacher volume by approximately 45%, with nighttime queries accounting for 31%); the minimum viable solution collects learning data via questionnaires, has teachers manually stratify and design differentiated tasks, and establishes course group peer

### 5.3 Dimension 3: Human-AI Collaborative Project-Based Practice-Five-Stage Development Process

The five-stage process [26] (requirements analysis→AI-assisted design→collaborative coding→AI-assisted testing→human review and optimization) is designed with "alternating leadership": humans retain sovereignty at key decision nodes, while AI releases efficiency in standardized execution phases. The AI usage log mechanism is the core design for preventing over-reliance. Log analysis revealed a counter-intuitive phenomenon: the top 25% of students by AI usage frequency also scored highest on adaptive programming tests (M=85.2), suggesting that high-frequency usage does not equal dependency but may represent intensive "learning→feedback" cycles. The ideal solution covers full-process automated tool chains; the minimum viable solution requires only free AI tools and shared documents.

### 5.4 Dimension 4: Multi-Dimensional Intelligent Assessment System-Process, Product, and Ability Triangle

In the three-dimensional assessment system (process 40% + product 35% + ability 25%), the most differentiating component is the "adaptive programming ability test"—students independently complete programming tasks in a time-limited environment without internet or AI, referencing Mayr's [27] "knowledge markers" concept to promote learning through assessment. The ideal solution deploys code detection and automated analysis systems; the minimum viable solution conducts tests in regular computer labs with network disconnected, substituting peer review for automated code

review.

### 5.5 Dimension 5: AI Literacy and Ethics Education-Cognition, Norms, and Responsibility Modules

The cognition module uses "AI independent completion vs. student independent completion vs. human-AI collaboration" comparative experiments to give students an intuitive sense of AI capability boundaries; the norms module replaces vague "no cheating" policies with "positive list + negative list"; the responsibility module uses case discussions to address ethical issues such as AI copyright, code security, and algorithmic bias. The minimum viable solution requires only the instructor to project-demonstrate 2–3 "AI code catastrophe" cases and organize classroom discussions.

### 5.6 Systematic Transformation of Teacher Roles

The reform requires teachers to undergo three transformations: from "expert on the podium" to "designer of learning scenarios," where the core work becomes crafting learning activities that elicit deep thinking; from "provider of correct answers" to "creator of cognitive conflicts," where the value of the teacher's deliberate probing questions is heightened when AI can instantly answer most questions; from "single-discipline instructor" to "human-AI collaboration coach," as teacher professional development must incorporate the dimension of "AI pedagogical knowledge."

## 6. Teaching Effectiveness: Quantitative Evidence and Qualitative Findings

### 6.1 Academic Performance: Overall Improvement and Variance Convergence

The experimental class final comprehensive score mean was 80.3 (SD=8.6), the control class 73.1 (SD=11.2), with significant difference ($t(159)=4.54$, $p<0.001$, Cohen's $d=0.723$), representing a medium-to-large effect size. The experimental class coefficient of variation decreased from 15.3% to 10.7%, and the low-score (<60) proportion dropped from 11.4% to 3.7%, indicating that AI-empowered personalized learning pathways had a pronounced "bottom-lifting" effect on disadvantaged students. Multiple struggling students reported: "Before, getting stuck meant waiting until next week's lab session to ask the teacher; now at least I can ask AI for a starting point."

### 6.2 Project Work Quality

The experimental class project excellence rate jumped from 31.2% to 67.8% ($\chi^2(1)=21.62$, $p<0.001$, Cramér's $V=0.37$), and code standardization (SonarQube pass rate) improved from 71.3% to 89.6% (Cohen's $d=1.525$). The dramatic improvement in code standardization should be understood as "students learned to use AI to review and improve code" rather than "AI replaced quality awareness."

### 6.3 Adaptive Programming Ability: AI as Training Tool, Not Crutch

Under AI-removed conditions, the experimental class adaptive programming test completion rate was 78.5%, the control class 65.3% ($t(159)=4.03$, $p<0.001$, Cohen's $d=0.88$), a large effect size. Sub-dimension analysis showed the most prominent improvement in problem decomposition and modeling (+22.4%), followed by code organization (+18.7%), with the smallest improvement in syntax memorization (+8.2%), validating the reform premise of "shifting the center of gravity upward to the design level."

### 6.4 Learning Experience and Satisfaction

Anonymous surveys showed 91.5% satisfaction in the experimental class (control class 73.8%), with 94.2% believing AI enhanced the learning experience. Qualitative interviews revealed three thematic clusters: (1) "AI is a personal tutor, not a ghostwriter"—students developed strategic AI usage literacy; (2) "The project made me feel like a programmer for the first time"; (3) "The AI log helped me see the improvement in my own question quality"—from "write a student class for me" to "review whether this code violates the Single Responsibility Principle."

### 6.5 Teacher Workload Optimization

The intelligent Q&A system handled approximately 45% of common questions, reducing the teacher's mailbox from 60–80 emails per week to 30–40. Automated code standardization review shortened each lab report grading time from approximately 15 minutes to approximately 8 minutes, allowing teachers to focus attention on higher-level judgments of design logic.

**Acknowledgments**

This work was supported by the University-level Research Project of Changji University (Grant No. KY2024020), the Changji University Key Laboratory of Artificial Intelligence and Computing Power Applications, the Xinjiang Social Science Fund Project (Grant No. 2024BXW139), and the Xinjiang Higher Education Teaching Reform Project (Grant No. XJGXJGPTB-2024079).

**References**

[1] Avouris N, Sgarbas K, Caridakis G, et al. Teaching Introduction to Programming in the Times of AI: A Case Study of a Course Re-design. arXiv preprint arXiv:2508.06572, 2025.

[2] Raihan N, Siddiq M L, Santos J C S, et al. Large Language Models in Computer Science Education: A Systematic Literature Review. arXiv preprint arXiv:2410.16349, 2024.

[3] Ziegler A, Kalliamvakou E, Li X A, et al. Measuring GitHub Copilot's Impact on Productivity. Communications of the ACM, 2024, 67(3): 54-63.

[4] Manna S, Sett N. Need of AI in Modern Education: In the Eyes of Explainable AI (xAI). arXiv preprint arXiv:2408.00025, 2024.

[5] Mayr C M. Knowledge Markers: An AI-Agnostic Concept for the Design of Programming Courses. arXiv preprint arXiv:2604.06331, 2026.

[6] Reihanian I, Hou Y, Chen Y, et al. A Review of Generative AI in Computer Science Education: Challenges and Opportunities in Accuracy, Authenticity, and Assessment. arXiv preprint arXiv:2507.11543, 2025.

[7] Bassner P, Frankford E, Krusche S. Iris: An AI-Driven Virtual Tutor for Computer Science Education. arXiv preprint arXiv:2405.08008, 2024.

[8] Chen Y J, Zhang Y H, et al. Intelligent Computing Systems: From Deep Learning to Large Models. 2nd ed. Beijing: China Machine Press, 2024. ISBN: 978-7-111-75595-1

[9] Wang J, Lan Y, Chen X. Exploring the Modular Integration of "AI + Architecture" Pedagogy in Undergraduate Design Education. arXiv preprint arXiv:2512.13730, 2025.

[10] Bulut O, Beiting-Parrish M, Casabianca J M, et al. The Rise of Artificial Intelligence in Educational Measurement: Opportunities and Ethical Challenges. arXiv preprint arXiv:2406.18900, 2024.

[11] Missawa T, Koizumi A, Tamura R, et al. Exploring Utilization of Generative AI for Research and Education in Data-Driven Materials Science. arXiv preprint arXiv:2504.08817, 2025.

[12] Ministry of Education of the People's Republic of China. Action Plan for Artificial Intelligence Innovation in Higher Education Institutions. Document No. Jiao Ji [2018] 3, April 2, 2018.

[13] Ministry of Education of the People's Republic of China. Education Informatization 2.0 Action Plan. Document No. Jiao Ji [2018] 6, April 13, 2018.

[14] Zhang J P, Chen S P, Zhang J H. Ethical Issues and Governance Pathways of AI Applications in Education. e-Education Research, 2021, 42(3): 5-13.

[15] Yu S Q, Wang Q. Analysis of the Collaborative Path Development of "AI + Teachers". e-Education Research, 2019, 40(4): 14-22. DOI: 10.13811/j.cnki.eer.2019.04.002

[16] Zhu Z T, Hu J. Exploring the Essence and Research Prospects of Education Digital Transformation. China Educational Technology, 2022(1): 13-22.

[17] Liu S N Y, Yang Z K, Li Q. AI-Enabled Personalized Learning: Technical Approaches and Practical Pathways. China Educational Technology, 2021(1): 1-8.

[18] Zhong S C, Tang Y W, Wang C H. Research on Human-Machine Collaborative Teaching Models in Smart Classrooms. e-Education Research, 2020, 41(11): 5-12.

[19] Wu D, Wei X R, Lu C. Educational Reform and Development in the Era of Education Informatization 2.0. China Educational Technology, 2018(1): 25-31.

[20] Song W, Liu L, Zhao H. Java Programming Course Teaching Reform and Practice. Computer Education, 2021(5): 142-146.

[21] Wang Y Z, Zhang W. Exploration of Project-Driven Java Course Teaching Mode Reform. Software Guide, 2020, 19(8): 218-221.

[22] Li M, Chen J, Wang L. Experimental

Teaching Model for Java Programming. Research and Exploration in Laboratory, 2021, 40(6): 198-202.
[23]Chen X M. Qualitative Research Methods and Social Science Research. Beijing: Educational Science Publishing House, 2000.
[24]Huang R H, Zhou W, Du J, et al. Three Fundamental Computing Issues in Intelligent Education. Open Education Research, 2019, 25(5): 11-22.
[25]Yang Z K, Wu D, Chen M. Emerging Technologies Facilitating the Reconstruction of the Education Ecosystem. China Educational Technology, 2019(2): 1-5.
[26]Gu M Y. Educational Transformation in the Era of Artificial Intelligence. Peking University Education Review, 2023, 21(1): 2-12.
[27]Yuan Z G. Digital Transformation in Education: What to Turn and How. Journal of East China Normal University (Educational Sciences), 2023, 41(6): 1-12. DOI: 10.16382/j.cnki.1000-5560.2023.03.001.