Modern Code

Organizational Guidelines for Practical Exercises

Faculty of Computer Science and Digital Health

October 2, 2025

The Modern Code course consists of theoretical lectures and practical exercises. Attendance is tracked through attendance lists.

The goal of the practical exercises is to demonstrate the application of theory from the lectures and strengthen practical experience with programming concepts, development tools, and problem-solving methodologies. The exercises deepen and complement the lectures, often beginning each exercise session with lecture-style presentation of new material or brief review of lecture content.

1 Grading scheme

The exercise grade is determined at the end of the semester based on the following assessment criteria:

(1) **Quality of submitted solutions** for exercise assignments (see below)
(2) **Short tests** during exercise sessions

Criterion (1) contributes **50%** and criterion (2) contributes **50%** to the exercise grade.

1.1 Homework Assignments

- **Schedule:** First assignment (UE1) this session, then every second session (UE3, UE5, UE7, UE9, UE11, UE13, UE15)
- **Total:** 8 homework assignments worth **24 points each**
- **Due date:** Usually midnight before the next exercise session
- **Late submissions: -4 points per day**
- You can skip **ONE assignment**
- Assignments only count if you reach at least **6 points (25%)**
- Assignments become progressively more challenging throughout the semester

1.2 Short Tests

- **Number:** 3 short tests during the semester (15 minutes each)
- **Grading:** Only the **best 2 tests** count toward your grade
- **Missing tests:** Missing one test is fine (no penalty)
- **All three tests:** Taking all three may provide a point upgrade if one test turns out poorly
- **Format:** Short practical coding exercises during exercise sessions.
- **NO AI OR MATERIALS ALLOWED!**

2 Submission Requirements

2.1 Digital Submission via Moodle

All solutions must be submitted digitally in the Moodle course using the designated upload areas for each assignment. Submit before the specified deadline (usually midnight). A cover page is **mandatory** for every submission and must include your **Name** and **Time spent in hours**. Missing information results in **-1 point** per missing detail, while a missing cover page entirely results in **-3 points**.

2.2 File Naming Convention

- PDF format: `MC_UEXX_NACHNAME.pdf`
- ZIP format (if additional files): `MC_UEXX_NACHNAME.zip`
  - Create folder: `MC_UEXX_NACHNAME`
  - Include all required files (including PDF)
- Replace `XX` with assignment number (e.g., 01, 02, etc.)
- Replace `NACHNAME` with your last name
- Ex. `MC_UE01_Krauss.pdf`
- **-1 point** deducted for incorrect naming

3 Documentation Requirements

Unless stated otherwise on the assignment sheet, the following requirements apply.

3.1 Required Sections for Programming Assignments

Programming assignments must include three main sections:

(a) Solution Idea: Explain your approach and algorithm. This demonstrates your understanding of the problem and shows you planned before coding, which helps instructors understand your reasoning for partial credit.

(b) Code: Complete, working Java code. This is the core deliverable that shows you can implement solutions. The code must be functional to demonstrate programming competency, and **without working code, the assignment counts as not submitted**. Code must be included in the PDF for easy review and grading.

(c) Tests: Test cases (screenshots or copy pastes of console output) and verification of your solution. This proves your code works with different inputs, shows you understand edge cases, and demonstrates systematic verification rather than just "it works on my machine."

3.2 Point Deductions for Missing Sections

- Missing Solution Idea (a): **-4 points**
- Missing Tests (c): **-4 points**
- Missing Code (b): **Assignment not submitted**

3.3 Formatting Requirements

- Use **light background** (no black screenshots)
- Inverted color schemes for code or tests: **-4 points**
- Solutions must be readable and follow proper format
- Solutions that do not meet the required format or are illegible are considered not submitted

4 Academic Integrity

4.1 Individual Work

Solutions must be worked out by students individually. **Teamwork is not allowed**. In case of justified doubts about authorship, all suspected team members' solutions will be graded as not submitted (0 Points). Please take this seriously, as this is plagiarism and a serious violation of academic integrity. Extreme or repeated violations may result in severe consequences, up to repeating the grade or expulsion from the university.

5 AI-Assisted Coding

5.1 AI Usage Policy

AI tools are allowed and strongly encouraged for all assignments. Use AI as a learning tool to enhance your understanding and productivity. AI can help with code generation, debugging, optimization, and learning new concepts. The goal is to learn how to effectively collaborate with AI in modern software development.

You must document your AI usage in the Solution Idea section of each assignment. Include a brief "AI Disclosure" describing which AI tools you used (e.g., GitHub Copilot, ChatGPT, Claude), how you used them (code generation, debugging, explanations), what you learned from the assistance, and how you modified or improved the AI-generated code. **No points are deducted for AI usage** - this is about transparency and learning. Focus on demonstrating your understanding of the final solution.

Examples of disclosure:

- "AI Disclosure: ChatGPT was used for grammar correction"
- "AI Disclosure: GitHub Copilot helped me debug and fix the following bug: ..."
- "AI Disclosure: Cursor was used for code generation from my pseudocode"
- "AI Disclosure: The entire submission was generated by Claude Code"

5.2 Best Practices for AI Use

When using AI tools, **always understand the code** rather than just copy-pasting output. **Test thoroughly** as AI-generated code may have bugs or edge cases. Use AI as a starting point, then **modify and improve** it for your specific needs. Learn from AI explanations to deepen your understanding, but **be critical** and verify AI suggestions for correctness. Most importantly, **document your process** to show your thinking and decision-making.

Don't submit AI output without understanding it, and **don't rely solely on AI** - you need to demonstrate your own problem-solving skills. Always be transparent about AI usage and remember that **AI is not allowed during short tests**, which are individual assessment moments. You need to learn how to apply hard skills - the ability to code and solve problems independently.

6 Getting Help

- **Course Discussion Forum:** Check Moodle for peer help
- **Office Hours:** Check with your instructor or tutor for available times
- **Programming Resources:** Use official Java documentation and IDE / AI help
- **Academic Support:** Contact the faculty for technical or academic issues

Good luck with your Modern Code journey!