Software Engineering

Introduction to the module

Outline

- Some ground rules
- What is this course module about?
- Objectives
- Structure
- Assessment
- References
- Study method

Rule #1: in the class...

- Please **keep quite**, ***unless*** you have something to share with the **TEACHER & CLASS**!
- Ask questions and raise issues if you have any:
  - share your thoughts with the TEACHER & CLASS!
  - in an orderly manner!

Rule #2: How to communicate...

- In the lectures and tutorials
- Use the designated Google classroom.
- Do NOT use email to ask questions that should have been asked using the above methods.
  - e.g. you may NOT get a reply if you email teaching staff about the assignments

    or about something that you don't understand in a lecture/tutorial.

What is this module about?

- A practical introduction to software engineering
- Consists of two parts:
  - Part I: enhanced object oriented programming with annotation
  - Part II: Software engineering method
    - a **disciplined** software development **process**
    - apply to a small-to-medium-sized software development project

Module objectives

- Apply core OOP concepts and techniques
- Apply a software engineering method
- Apply a practical and relatively formal requirement engineering technique
- Apply UML to analyse and design software

Key features!

- Program **design is essential**
- BUT there are **different design methods**
- You will be **required to follow** a specific (very practical!) design method:
  - to **strictly follow** the design rules
  - your design and code are **marked automatically**!

    → ***follow*** the rules will result in a ***good mark***

    OR ***failure*** to do so will give you a ***low mark*** !!!

Choice of PL

- Why Java?
  - pre-requisites and experience
  - an object oriented programming language
- Applicable to other OO (-related) programming languages:
  - e.g. C++, C#, PHP

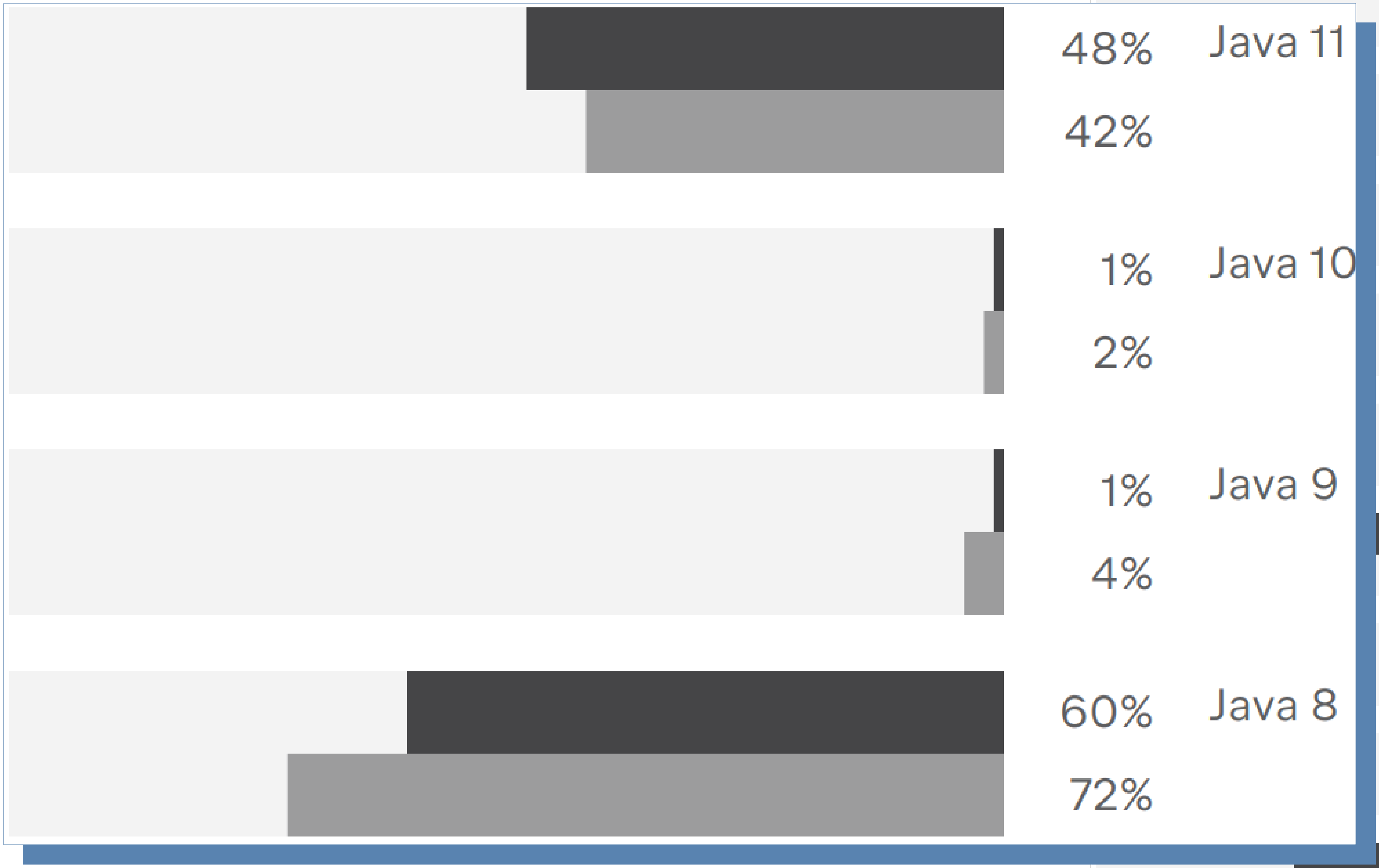
Java

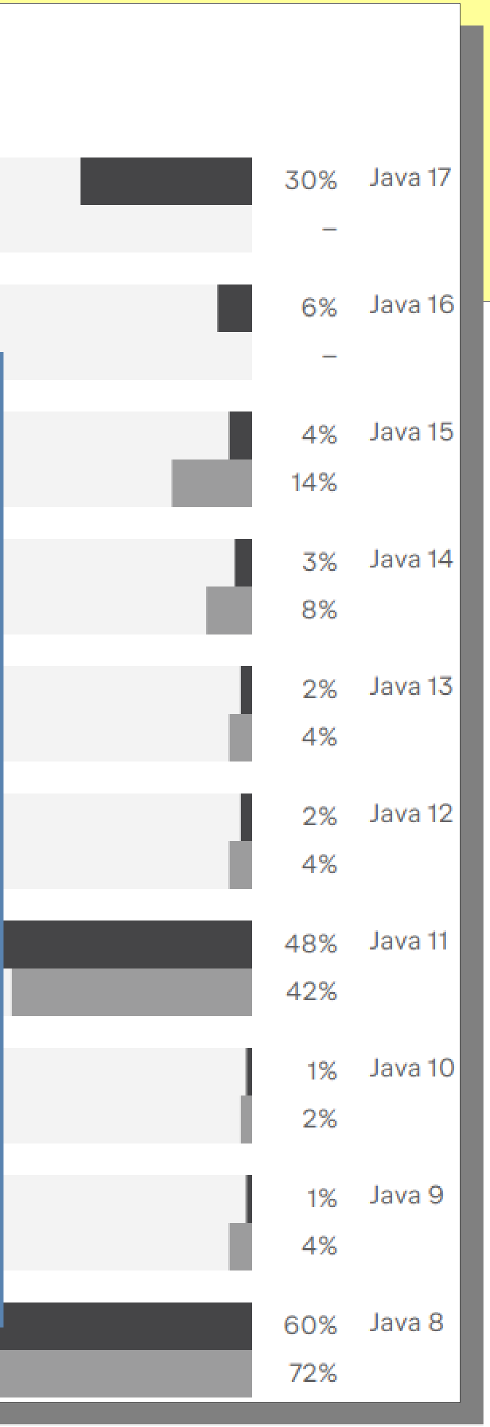
- Java (>= 1.8):
  - a popular Long-Term Support (LTS) version
  - still used in many systems (see survey)
- In this course: **Java 11+**
  - a next LTS version from Java 1.8
  - supports type inference (from Java 10), new lambda syntax
- Recent LTS version(s) that are gaining popularity:
  - Java 17 (LTS)

Java version usage

https://www.jetbrains.com/lp/devecosystem-2022/java/

Which version of Java is regularly used?





Part I: Enhanced OOP

- **Enhanced OOP:** using annotation in OOP:
  - enhance design with rules
  - compile-time design validation
- **Type hierarchy**
- **Code robustness** with exceptions

Part II: Software engineering

- Software engineering **method**
  - a structured and well-defined process
- **Requirement** engineering
  - focus: requirement analysis
- **Design**
- **Implementation**

Assessment

- 2 assignments:
  - increasing level of complexity
  - A2 depends on A1
- A2: a small SE project
  - to understand how a software is developed and to develop an extension for it
  - incremental development and in-line with the lectures
  - finish the software in the last week!
  - deliverables: software + technical report

Homeworks

- Home works include:
  - tutorial answers
  - additional exercises (depends)
- Complete homeworks to maintain learning progress

Main references

Main Coursebook:

Duc M. L., Object Oriented Program Development, 2023

Text book:

- Liskov B. and Guttag J., *Program Development in Java: Abstraction, Specification, and Object Oriented Design*, Addison Wesley, 2001

Supplementary:

- Sommerville I., *Software engineering*, 10th ed, Pearson, 2015
- Larman C., *Applying UML And Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, 3rd ed, 2004

Main references: Java

- Java (>= 1.8) API documentation:
  - https://docs.oracle.com/en/java/javase/17/docs/api/index.html
- Java tutorials:
  - https://docs.oracle.com/javase/tutorial
- Relevant links are provided on GClassroom

Study approach

- **Learn by doing**: *practice makes perfect!*
  - try the code examples
  - do the exercises
- **Self study**: *you must put in effort!*
  - read the lectures and related resources
  - submit home works each week
- **Get help**: *don't delay your problems!*
  - ask questions in lectures
  - make the most of the tutorials!
  - use the forum

Q & A