Object Oriented Programming
(04JEYLM, 04JEYOA, 04JEYSM)

A.A. 2022/2023

SoftEng
http://softeng.polito.it

Version 1.1.0

Teachers

- Stefano Di Carlo
  - Dip. Automatica e Informatica
    - IV Piano

☎ 011 564 7080
✉ stefano.dicarlo @ polito.it
🌐 https://smilies.polito.it

- Gianluca Amprimo

✉ gianluca.amprimo@polito.it

- Roberta Bardini

✉ roberta.bardini@polito.it

Modalità di lavoro proposta

Tre tempi

- ↶ Prima delle lezioni
- ▼ Durante le lezioni ufficiali
- ⇝ Altri momenti

Collaboration Tools

- Virtual Classroom @ PoliTo
  - Lezioni in streaming + registrazioni
  - Unidirezionale (no chat!)

Dropbox

  - Cartella del corso con tutto il materiale

Telegram

  - Comunicazioni, annunci e interazioni
  - https://t.me/+z8pTosSy_cQ4MzVk

Schedule

- Tuesday 14.30 – 16.00
  - Room R2
- Wednesday 14.30 – 16.00
  - Room R1
- Thursday 10.00 – 13.00
  - Aula R1

- Mercoledì 16.00 – 19.00
  - Laib 3
  - Due squadre a settimane alterne

Laboratori a partire dalla terza settimana (16 Marzo)

Calendario Laboratori

	Squadra 1 A – CON	Squadra 2 COR – G
Lab 1 – Basics	16 / 3	23 / 3
Lab 2 – Inheritance	30 / 3	13 / 4
Lab 3 – Collections	20 / 4	27 / 4
Lab 4 – Stream	4 / 5	11 / 5
Lab 5 – I/O	18 / 5	25 / 5
Lab 6 – Riepilogo	1 / 6	8 / 6

COURSE ORGANIZATION

Topics

- Software Engineering
  - Software Life Cycle
  - Design
  - Test
  - Configuration management
  - Object-oriented paradigm
- Java programming language
  - Java syntax
  - Standard libraries

Objectives

- Understand how software development works
- Become familiar with the basic development support instruments
- Learn the Java language
- Acquire capability to write and test simple Java programs
- Learn using development tools

Tools

Organization of the course

- Lectures (~50h)
  - Software Engineering (~15h)
  - Java (~35h)
- Classroom exercises (~20h)
  - Examples (~10h)
  - Assignment solutions (~10h)
- Lab work (~15h)
  - Every week (since W3)

Labs

- LAIBs
  - 1.5h with Teaching + Student Assistants
  - 1.5h with Student Assistant
- Assignments
  - Programs to be completed/modified
  - Similar process as in the final exam
- Assessed but not graded
  - Essential for final exam
  - You must be able to use all the software tools in order to pass the exam

The only way to learn a programming language is by coding.

This is the way!

Prerequisites

- Mandatory
  - Procedural programming (e.g. C)
- Recommended
  - Abstract data types
    - Lists, trees etc.
  - Algorithms
    - Sort, search, list insert etc.

Initial self-assessment

- Do you know enough ''C''?

- Or https://softeng.polito.it/survey/271692?lang=it

Self-assessment questions

- Proposed during the course
- A set of closed answer questions
- Instrument to enable your self-assessment
  - Useful for us to detect possible problems
- Web based
  - Not anonymous
  - Results not used for grading

Software

- Mandatory
  - JDK 11.0
    https://docs.aws.amazon.com/corretto/latest/corretto-11-ug/downloads-list.html
  - Eclipse IDE – 2022-12
    - https://www.eclipse.org/downloads/packages/
- Useful
  - Astah UML – (free student license)
    - http://astah.net/editions
  - Papyrus plug-in for Eclipse
  - Any UML modeling tool

https://oop.polito.it/doc/ReferenceSoftware_it.html

Final Exam

Final Exam

- Part I: Programming (~85%)
  - Step I: during exam write the code
  - Step II: at home fix the code
- Part II: Theory (~15%)
  - Closed answer written questions
- Rules
  - 2 hours

Final Exam – Programming

- Abilities verified
  - Analyze simple textual requirements
  - Design a solution to address problem
  - Write correct and complete Java program
  - Use development tools
  - Understand unit tests and their reports

Final exam – Programming

- Phase 1 – in the lab, at exam time
  - Develop Java application, given
    - a textual specification of requirements
    - a skeleton code for the main functions
  - Submit initial version

- Phase 2 – at home, later
  - Receive acceptance tests results
  - Fix the app
  - Submit final version
    - Within given deadline (~5 days)

Final Exam – Assessment

- Programming
  - Functional correctness
    - Proportion of tests passed by the program version delivered in the lab
  - Rework to fix / complete program
    - Amount of changes between lab version and final version
- Theory
  - Correct answers

READINGS

Readings – Java

- Java Documenation
  - http://www.oracle.com/technetwork/java/javase/documentation/index.html
- Arnold, Gosling, Holmes. "The Java Programming Language – 4th edition", Addison-Wesley, 2006
- B.Eckel, "Thinking in Java", Prentice Hall, 4th Ed., 2006
  - https://www.mindviewllc.com/quicklinks/
- R. Urma, M. Fusco, A. Mycroft. "Modern Java in Action: Lambdas, streams, functional, and reactive programming." Manning, 2019.
  - https://www.manning.com/books/modern-java-in-action
- B.Eckel. "On Java 8", Mindview, 2018
  - http://www.onjava8.com/

Readings – Sw Engineering

- Bruegge, Dutoit. *Object-Oriented Software Engineering Using UML, Patterns, and Java*. Pearson, 2009
- *ISO/IEC/IEEE Std 12207-2008 for Systems and Software Engineering – Software Life Cycle Processes*
  - http://ieeexplore.ieee.org/document/4475826/

Readings – Test

- ISO/IEC/IEEE, Std 29119-1 Software and systems engineering – Software testing – Part 1: Concepts and definitions, 2013.
- ISTQB, Certified Tester Foundation Level Syllabus, 2001
  - http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4

Readings – Config Management

- Collins-Sussman, Fitzpatrick, Pilato. *Version Control with Subversion*, 2001
  - http://svnbook.red-bean.com
- IEEE Std 828-2012 *Standard for Configuration Management in Systems and Software Engineering*, 2012
- Semantic Versioning
  - http://semver.org

Readings – Design

- M.Fowler, K. Scott, *UML Distilled*, 3rd ed. Addison-Wesley, 2003.
- E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.
- E.Freeman, E.Freeman, K.Sierra, B.Bates. *Head First Design Patterns*, O'Reilly, 2004