00
Programmazione ad Oggetti
Introduzione al corso

Mirko Viroli
mirko.viroli@unibo.it

C.D.L. Ingegneria e Scienze Informatiche
Alma Mater Studiorum—Università di Bologna, Cesena

a.a. 2025/2026

Docenti

Titolare del corso: Prof. Mirko Viroli

- e-mail — mirko.viroli@unibo.it
- homepage — https://www.unibo.it/sitoweb/mirko.viroli

Modulo di laboratorio: Prof. Danilo Pianini

- e-mail — danilo.pianini@unibo.it
- homepage — https://www.unibo.it/sitoweb/danilo.pianini

Tutors: Ing. Martina Baiardi e Ing. Angela Cortecchia

- e-mail — martina.baiardi@unibo.it
- e-mail — angela.cortecchia@unibo.it

Contatti con gli studenti

Chi contattare

- Mirko Viroli: contenuti tecnici del corso + organizzazione del corso
- Pianini: strumenti e esercizi di laboratorio

Attraverso il forum

- Per domande la cui risposta è di interesse generale
  (e quindi tutte le domande tecniche)

Via mail al docente

- Per questioni personali
- Per domande che consentono risposte concise

Ricevimento

- (annunciato nelle Home Page dei docenti)

Sito del corso

Sito virtuale di Ateneo

- `https://virtuale.unibo.it/course/view.php?id=71105`
- sarà il luogo degli avvisi (e notifiche), forum di discussione, produzione di materiale
- tutti gli studenti che seguono il corso si iscrivano, e lo tengano d'occhio
- in particolare: è molto utile avere le slide sottomano a lezione

Organizzazione generale del corso

Lezioni aula (due da 3 ore la settimana)

- Illustrano i concetti teorici, metodologici e pratici
- Basate su slide proiettate (ma non solo)

Laboratorio (turni da 3-4 ore a settimana)

- O il lunedì o il martedì (il che vi lascia un giorno libero)
- Illustra ulteriori aspetti metodologici e pratici
- Con esercizi necessari alla comprensione e alla sperimentazione
- È parte integrante del corso

Studio a casa (almeno 4 ore a settimana) – p.e., nel giorno libero

- Rilettura slide, esperimenti pre- e/o post-laboratorio
- È essenziale se volete rimanere in pari...

Un sinonimo del corso: "Progr. di Sistemi Software"

Elementi essenziali

- Costruzione del software, e quindi di sistemi
- Analisi problemi, e organizzazione di soluzioni
- Tecniche base ed (alcune) avanzate di programmazione ad oggetti
- Introduzione a principi/tecniche di "programmazione moderna"
- Elementi di gestione di un progetto software
- Utilizzo di strumenti integrati di sviluppo

Questo corso nel più ampio contesto dei vostri studi

- Curriculum (progetti software, Java, OOP moderna, . . . , C#)
- Enfasi sull'approccio metodologico
- Target di qualità piuttosto elevato
- È cruciale dedicargli subito il tempo necessario

Programma (di massima) del corso

Parti principali

- Elementi base di programmazione OO e Java
- Polimorfismo (ereditarietà, subtyping, genericità)
- Librerie (I/O, grafica, concorrenza)
- Integrazione col paradigma funzionale (lambda, streams)
- Pattern e buone pratiche di programmazione

Testi di riferimento (non necessario l'acquisto)

Programmazione in Java

- B.Eckel. Thinking in Java, 4th edition.
- J.Block. Effective Java, 4th edition.

Altri riferimenti

- E.Gamma et.al. Design Patterns Elements of Reusable Object-Oriented Software.
- R.Martin. Clean Code: A Handbook of Agile Software Craftsmanship

Software

Java

- Framework Java: OpenJDK 21 (Open Java Development KIT)
  - `https://openjdk.org`
- Integrated Development Environment: Visual Studio Code + Java plugin
  - `https://code.visualstudio.com/`
  - `https://code.visualstudio.com/docs/languages/java`
- Altri strumenti
  - Git
  - Gradle

Istruzioni sull'installazione (sul PC di casa)

- Già disponibili su "virtuale"
- `https://unibo-oop.github.io/software-installation/`
- Molto importante rendersi operativi a casa nel giro di una settimana!
- L'uso di VSCode sarà necessario solo fra 2/3 settimane

$\Rightarrow$ sarebbe consigliato l'uso del sistema operativo Linux

Sugli LLM e la generazione di codice

LLM – Large Language Model

- una tecnica di AI, che si basa sull'addestramento di una rete neurale a predire la continuazione di un testo
- sono reti a molti miliardi di parametri/sinapsi (come e più del cervello umano)
- vengono allenate con quantità enormi di testi (libri, riviste, siti web)
- sono intrinsecamente in grado rispondere a domande, in modo "statisticamente" coerente col loro training set

LLM e generazione del codice

- molti strumenti riescono a generare codice a partire da una formulazione del problema, alcuni anche in modo integrato all'IDE (GPT, Bard, LLama, Copilot,...)
- il codice prodotto è generalmente corretto solo in casi relativamente semplici
- a livello professionale cominciano a essere molto usati, e lo saranno sempre più

Come vanno usati in questo corso?

- tassativamente, NON vanno usati (né in lab, né a casa, né all'esame)
- è necessario imparare senza scorciatoie, per poterli usare in modo proficuo in futuro

Esame

Prova scritta

- Durata 1.5 ore circa, svolta in laboratorio
- Verifica mirata di capacità tecniche, di problem-solving, buona progettazione OO

$\Rightarrow$ Forniremo esercizi-tipo in laboratorio

$\Rightarrow$ I temi d'esame del passato sono disponibili e abbastanza indicativi

$\Rightarrow$ Svolta in laboratorio

Discussione progetto

- Progetto sviluppato in gruppo, 70 ore circa a testa (vedi i 12 CFU)
- Concordato col docente prima di iniziare
- Da relazionare/presentare bene, poi discusso su appuntamento
- Consegne con deadlina scelta da voi (entro 4 mesi) ma a quel punto stringente:
  non durante le lezioni, idealmente entro metà febbraio

$\Rightarrow$ I dettagli discussi a metà corso

$\Rightarrow$ Regole d'esame (in versione draft) su "virtuale" molto presto

Prerequisiti

Buona conoscenza

- tecniche di programmazione imperativa/strutturata
- costruzione e comprensione di semplici algoritmi e strutture dati

Attenzione a chi è già "fluente" in linguaggi ad oggetti

- Java o C#
- disimparare le "bad practice" richiede umiltà e fatica...

Attenzione a chi usa Copilot e altri LLM e già non ne può fare a meno...