

# Software design

## Course introduction

---

Paolo Burgio  
paolo.burgio@unimore.it



**UNIMORE**  
UNIVERSITÀ DEGLI STUDI DI  
MODENA E REGGIO EMILIA

High Performance  
Real Time **Lab**



**SE NON LO SAI  
SPIEGARE  
IN MODO SEMPLICE,  
NON L'HAI CAPITO  
ABBASTANZA BENE**

- Einstein



# THE EXAM

---

## **Mandatory** written test

- › A mix of multiple (closed) responses, and open questions
- › Up to 30/30
- › 3 dates

See Q&A on website

*Optionally*, to improve your mark you can choose between

- › Oral (3-4 questions)
- › Project (recommended for internship and theses)
- › (typically **+4/-∞**)

# Course material

---

## Course website

<http://hipert.unimore.it/people/paolob/pub/ProgSW/index.html>

## Course slides

- › Available on Moodle, early preview on github
  - <https://github.com/HiPeRT/ProgSW.git>
- › Typically, enough to pass the exam, also if not attending classes
- › Hands-on exercises



## Textbooks

- › See course website
- › Add reference at the end of each slides block





# Required skills

---

Object oriented programming

- › You should know about Java..but any OO language is fine!
- › MS C# is the most modern/advanced for the things I'll teach you
- › IDE (Eclipse? VS Code? CLion?)

The Python  problem

- › PY and other “dynamic” languages are less suitable for large-scale methodologies
- › Better for faster prototyping

Tools for creating diagrams

- › MS Visio, Miro, Draw.io, even powerpoint!!!

Passion, passion, passion!!



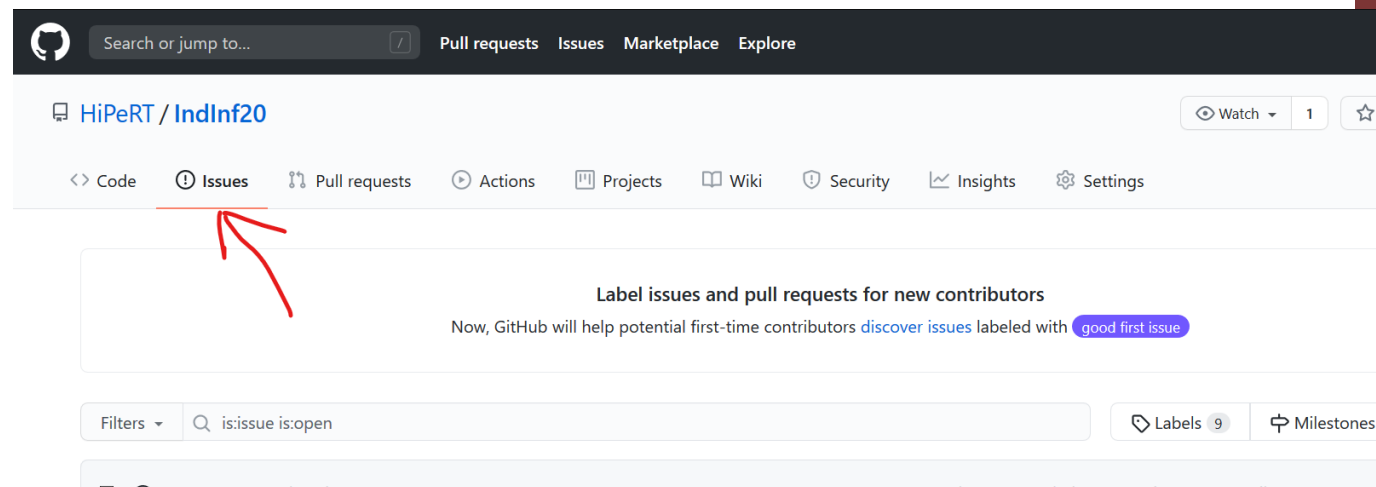
# How to contact me?

AKA: ricevimento

› [paolo.burgio@unimore.it](mailto:paolo.burgio@unimore.it)

But let's try something different..

- › For every question, open a ticket ("issue") on GitHub
  - <https://github.com/HiPeRT/ProgSW.git>
- › So, all of your colleagues will enjoy the answers
- › Netiquette: before asking, search in "issues"





What is this about?



# Software design methodologies

---

AKA: software engineering

- › Not really the same thing, but it helps!
- › Focus on the **process**, not on the **code**

One solution to rule 'em all?

- › Multiple methodologies, tailored for different needs
- › All of them boil down to few principles/families
- › If you structure the process, you can create tools to automate it!
  - Ex: folder structure for your documents

Why?

- › “Few weeks of debugging can save you 2 hour of design”
- › “Weeks of coding can save you hours of planning”
- › Code/Technical debt





# TECHNICAL DEBT

I DON'T UNDERSTAND WHY IT TAKES SO LONG TO ADD A NEW WINDOW.



@VINCENTDNL



# ***"But...will this restrict my talent?"***

---

**Absolutely NOT!!**

- › On the contrary, it will help you express your talent to its full potential

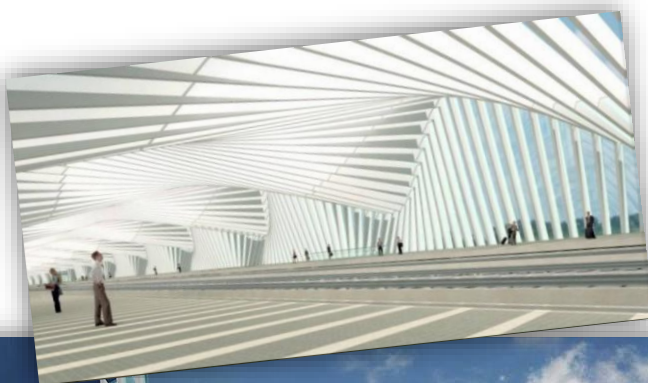
This course will:

- › teach you collaborative teamwork
- › help you removing repetitive tasks that take unnecessary time, by automating them
- › let you produce good documentation, and yet no unnecessary documentation

Where's the catch?

- › + 15/20% time, but well spent!

- 





# Documentation: why?

---

No-one programs alone!!

(...no-one programs **large scale software** alone)

Code documentation to

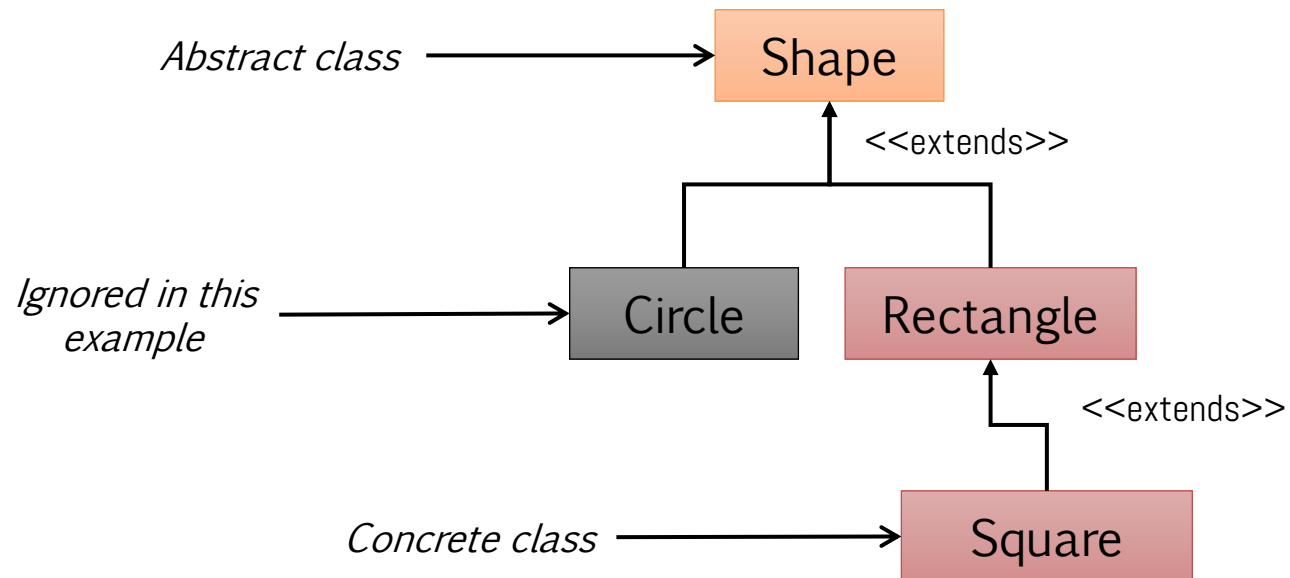
- › Teach how to use our SW artifacts
- › Explain how to modify them
- › Show how to test/deploy/use them/their code in other projects...

Licensing!!!



# Communicate

- › A standardized way of expressing concepts

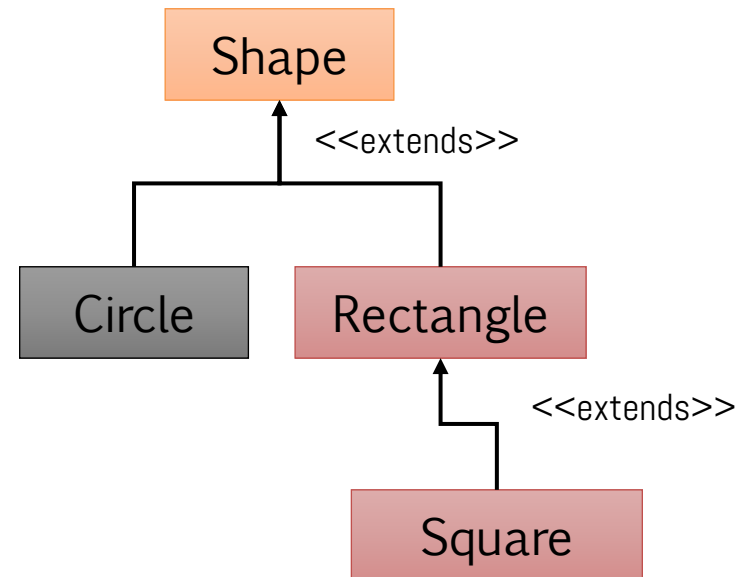




# Of rectangles and squares

A classical example

› Do you remember that?





# The story so far...

---



Programming is taught using very simple concepts

- › Object-Oriented Programming
- › Often **mixed** with procedural/functional programming
- › **Dynamic typing** as powerful mean to deal, e.g., with polymorphism

Clutter is removed

- › Pointers and explicit memory management are powerful yet dangerous!
  - C++ managed, RUST try to get rid of them
- › Parallel programming hidden within libraries, e.g., Python modules
- › Production code often is written in languages more suitable for fast prototyping (e.g., Python)

Software structured in monolithic manner

- › Middlewares are often seen as a library, or a communication mechanism
- › E.g., MPI, MQTT, .netCore



# The present: agile and microservices

---

How do we develop today?

- › Traditional project management and tools are still used
- › New generation of agile methodologies to enable quick Time-To-Market
- › *“Code today, deploy tomorrow, get worried about that next year”*

Large scale projects require flexible and resilient structures

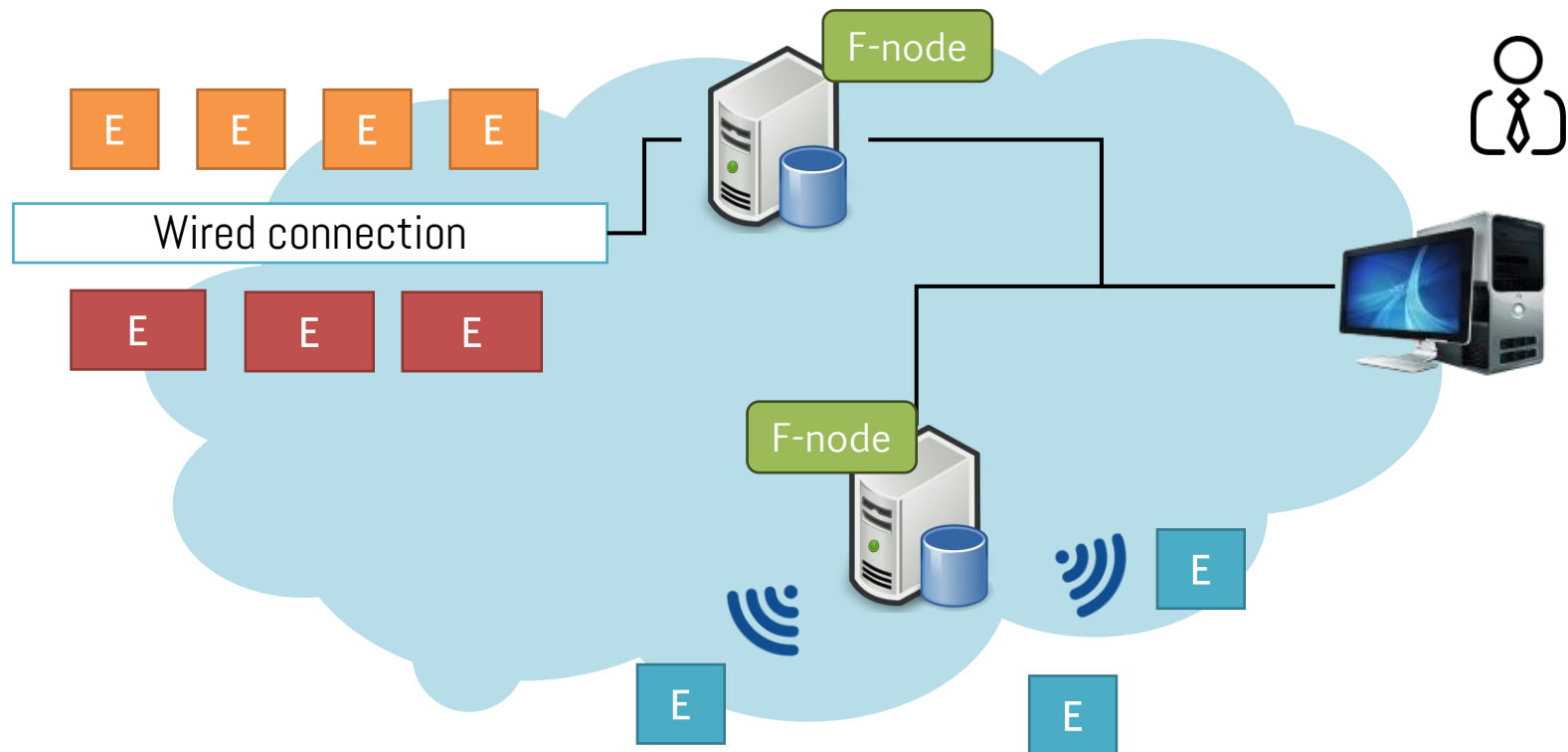
- › E.g., Google server farms
- › First design the communication channels and protocols
  - MQTT, Sockets, ROS with JSON, etc
  - Aka: the **interaction** between software components
- › ...then, design single SMALL components, with very restricted functionalities
  - DB, Web API/Frontend, aggregation logics...



# The future: CPS - Cyber Physical (Eco)Systems

Tight interaction with the environment

- › Computational edges with highest energy efficiency (Performance => Performance/Watt)
- › Highly scalable hierarchy of computers, with (multiple) Cloud/Fog nodes
- › Parallel/distributed computation: GPGPUs, multi-threading, MQTT, ROS, MPI...





# Our journey

---

## Will cover these main topics

- › Requirements, KPIs and test-driven design
- › Collaborative tools
- › Design tools: UML, E/R diagrams, ...
- › Design patterns
- › Advanced programming: how to structure?

..and...

- › Use-case: we'll meet some companies...
- › Use-case: HiPeRT's Fire: Roboracer.ai, UNIMORE MMR E-Driverless, UNIMORE Racing?

...always with an hands-on approach!

# Credits to

---

- › Prof. Cabri, for the material of the past “Progetto del SW” course



- › Uncle Bob (& Friends), for the insightful details on modern structures for SW  
(And for being always harsh in code/SW reviewing)





# References

---



## Course website

- › <http://hipert.unimore.it/people/paolob/pub/ProgSW/index.html>

## My contacts

- › [paolo.burgio@unimore.it](mailto:paolo.burgio@unimore.it)
- › <http://hipert.mat.unimore.it/people/paolob/>
- › <https://github.com/pburgio>