ADVANCED PROGRAMMING LANGUAGES

Fall 2024, CM 0632. MSc in Computer Science and Information Technology, Ca' Foscari University of Venice

What is this course about?

It's about fun programming!

We'll take an in depth look at the advanced concepts and techniques of modern programming languages (functional, object-oriented and concurrent) together with an introduction to the foundational tools from programming language theory and type theoretic frameworks.

Most of the development will focus on **Scala**, which provides a nice framework where modern techniques have been integrated based on a principled design, but we will draw also from other statically typed languages from which Scala' design is derived, such as **ML, Haskell, Java, Rust** as well as, occasionally, on dynamically typed languages like **Python**.

Contents

Introduction

- Evolution of programming languages design and implementation
- Introduction to functional programming in Scala.

Typing foundations of programming languages

- Untyped Lambda Calculus and its operational semantics
- Typing: types, type systems, type safety

Functional data and function modeling

- Immutable non-encapsulated data structures
- Lists and trees: first-order, higher-order and polymorphic functions

Subtyping and subtype polymorphism

- The substitution principle and the subtype relation

- Typing and Subtyping: nominal vs structural

Object-Oriented data modeling (in Scala)

- Encapsulated data types: Scala's traits, classes, and hierarchy of types

Type parametrization

- Generics, variance and type bounds
- Collections

Ad hoc polymorphism

- From overloading to constrained parametric polymorphism (with F-bounded polymorphism)
- Qualified types and type classes

Strict vs Non-strict evaluation

- Evaluations strategies: foundations and implementation consequences
- Lazy evaluation at work: thunks, lazy lists and infinite data structures

Error handling

- Exceptions: strengths, weaknesses and alternatives.
- Options and Try

Monads and monadic programming

- Conceptual framework
- Applications: State, IO

Textbooks and other resources

- [FP-Scala] Functional Programming in Scala. P. Chiusano, R. Bjarnason. Manning Press. 2015
  Source code available at: https://github.com/fpinscala/fpinscala/
- [TAPL] Types and Programming Languages. B.C. Pierce. The MIT Press. 2002.
- Papers and websites that will be made available as we progress through the course (pls refer to the moodle site for the course

Weekly timetable

- Tuesday 14:00 - 15:30, Aula B
- Wednesday 12:15 - 13:45, Aula B

Exam and grades

Will be encouraging collaborative work and try to make the course interactive. The final grade will be based on three elements:

- active class partecipation
- a written test
- (collaborative) work on programming projects / readings of research articles

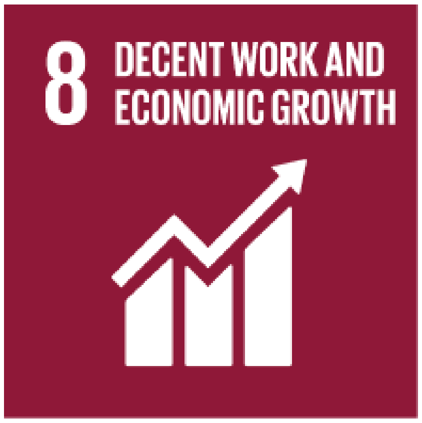
Why take this course?

Because . . .

- it will be fun!
- you'll learn amazing programming techniques
- you'll be exposed to some of the most fascinating pieces of computer science theory

Follow-up homework

As you (should) know, for each course we teach, we are asked to identify one or more goals of the U.N. Agenda for Sustainable Development which we believe are related to the subject of the course. Well, for this course I have chosen the following one: it's more a wish for you and you future than anything else 😉



A more interesting exercise, is to find out which programming languages are more environmental friendly and why. [Hint: google will help]