

Advanced Programming [AP-2024] – Andrea Corradini

Detailed Syllabus

This document lists the topics presented along the course using the PDF slides published on the course web page [<http://pages.di.unipi.it/corradini/Didattica/AP-24/>]. The reading material consists of the slides presented during the course AND of the additional documents listed below for each topic.

[**Note:** The topics marked **[Optional]** will not be asked by the lecturer during the oral exam, unless they are chosen by the student].

1. Languages and Abstract Machines. Compilation and interpretation schemes.

[Chapter 1 [<http://pages.di.unipi.it/corradini/Didattica/AP-24/DOCS/GM-ch1.pdf>] of book *Programming Languages: Principles and Paradigms*, by M. Gabrielli and S. Martini.]

2. Runtime Systems, Introduction to the JVM and the Java Memory Model

a. JVM internals

[*JVM Internals*, by J.D. Bloom, <http://blog.jamesdbloom.com/JVMInternals.html>]

b. The JVM Instruction Set

[*Java Code To Byte Code - Part One*, by J.D. Bloom, http://blog.jamesdbloom.com/JavaCodeToByteCode_PartOne.html]

c. **[Optional]** See also [Chapter 2 of *The Java Virtual Machine Specification, Java SE 8 Edition* <https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf>]

d. The Java Memory Model

[J. Manson, W. Pugh, and S.V. Adve. [The Java memory model](#). In POPL '05: 378-391]

[Optional] See also [Section 7.4 (Memory Model) of *The Java Language Specification*, <https://docs.oracle.com/javase/specs/jls/se17/html/index.html>]

3. Software Components

a. An introduction to Software Components

[Chapters 1, and 4 of **[COMP]**¹ - *Software Components: Beyond Object-Oriented Programming*. C. Szyperski, D. Gruntz, S. Murer, Addison-Wesley, 2002.]

b. Software Components: the Sun approach, JavaBeans

[Sections 14.1 (p. 261-269), 14.3 (p. 284-293) of **[COMP]**¹
[Sections 1, 2, 6, 7, 8 of *The JavaBeans API Specification*,
<http://pages.di.unipi.it/corradini/Didattica/AP-24/DOCS/JBS.101.pdf>]

c. Reflection in Java

[*The Java Tutorial on the Reflection API*, excluding *Arrays and Enumerated Types*
<https://docs.oracle.com/javase/tutorial/reflect/index.html> .]

d. Annotations in Java

[*The Java Tutorial on the Reflection API*,
<https://docs.oracle.com/javase/tutorial/java/annotations/index.html>

e. Frameworks and Inversion of Control: Decoupling components; Dependency Injections; IoC Containers

[*Inversion of Control*, by Martin Fowler,
<https://martinfowler.com/bliki/InversionOfControl.html>]

¹ Selected chapters of book [COMP] can be downloaded from the course web page.

[*Inversion of Control Containers and the Dependency Injection pattern*, by Martin Fowler, <https://martinfowler.com/articles/injection.html>]

f. On designing Software Frameworks

[*Using classic problems to teach Java framework design*, by H.C. Cunningham, Yi Liu and C. Zhang, Science of Computer Programming 59 (2006),
<http://pages.di.unipi.it/corradini/Didattica/AP-24/DOCS/FrameworkDesign.pdf>]

4. Polymorphism

a. A classification of Polymorphism

b. Polymorphism in C++: inclusion polymorphism and templates

[Overloads and Templates in C++ <http://www.cplusplus.com/doc/tutorial/functions2/>]

[Inclusion polymorphism in C++, <http://www.cplusplus.com/doc/tutorial/polymorphism/>]

[Templates in C++, <http://www.cplusplus.com/doc/oldtutorial/templates/>]

c. Java Generics, Type bounds and subtyping, Subtyping and arrays in Java, Wildcards, Type erasure

[Java Generics <https://docs.oracle.com/javase/tutorial/java/generics/index.html>]

d. The Standard Template Library: an overview

[*The Standard Template Library Tutorial*, by Johannes Weidl: Page 4, 12 and parts of Chapter 4 "*Learning STL*",

<http://pages.di.unipi.it/corradini/Didattica/AP-24/DOCS/stl-tutorial-Weidl.pdf>],

e. Generics and inheritance: invariance, covariance and contravariance in Java and other languages

[Covariance and Contravariance in Java,

<https://dzone.com/articles/covariance-and-contravariance>]

[**Optional**] [Covariance and Contravariance in C#,

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/covariance-contravariance/>]

[**Optional**] [Covariance and Contravariance in Scala,

<http://blog.kamkor.me/Covariance-And-Contravariance-In-Scala>]

5. Functional Programming

a. Introduction to functional programming

[Section 10.1 and 10.2 of Chapter 10 of *Programming Language Pragmatics*, by Michael Scott, 3rd edition.

<http://pages.di.unipi.it/corradini/Didattica/AP-24/DOCS/Scott-ch10.pdf>]

b. [Optional] A digression on the lambda-calculus [Introduction to Lambda Calculus,

<http://www.inf.fu-berlin.de/lehre/WS03/alpi/lambda.pdf>]

6. Haskell

a. Introduction to Haskell, Laziness, Call by need, Basic and compounds types, Patterns and declarations, Function declarations

[Introduction to Haskell, by John C. Mitchell,

<http://pages.di.unipi.it/corradini/Didattica/AP-24/DOCS/Ch5.pdf>]

[An excellent tutorial on Haskell: <http://learnyouahaskell.com>, Sections "Introduction" and "Starting out"]

[Basic Types and Type Classes: <http://learnyouahaskell.com/types-and-typeclasses>]

[Functions in Haskell: <http://learnyouahaskell.com/syntax-in-functions>]

b. List comprehension, Algebraic Data Types, Higher-order functions, Recursion

[Recursion: <http://learnyouahaskell.com/recursion>]

[Higher-order functions: <http://learnyouahaskell.com/higher-order-functions>]

- c. **Type classes in Haskell**
[Type Classes in Haskell, by John C. Mitchell,
<http://pages.di.unipi.it/corradini/Didattica/AP-24/DOCS/Ch7.pdf>]
- d. **The Maybe constructor and composition of partial functions.**
Monads in Haskell
[A very short tutorial on Monads
<http://www.idryman.org/blog/2014/01/23/yet-another-monad-tutorial/>]
[Monads as Containers, [https://wiki.haskell.org/Monads as containers](https://wiki.haskell.org/Monads_as_containers)]
[Monads as Computations, [https://wiki.haskell.org/Monads as computation](https://wiki.haskell.org/Monads_as_computation)]
- e. **[Optional] The IO Monad**
https://en.wikibooks.org/wiki/Haskell/Understanding_monads/IO
[https://wiki.haskell.org/IO inside](https://wiki.haskell.org/IO_inside)

7. Functional programming in Java 8

- a. **Lambdas in Java 8**
[Lambda Expressions in Java
<http://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>]
- b. **The Stream API in Java 8**
[Aggregate Operations in Java
<https://docs.oracle.com/javase/tutorial/collections/streams/index.html>]

8. An overview of Rust

Reading material: Slides presented in class and selected sections of

<https://doc.rust-lang.org/book/index.html>, including:

- a. **Ch. 3 – Common Programming Concepts; Ch. 4 – Understanding Ownership**
- b. **Ch. 6 – Enums and Pattern Matching; Ch. 10 - Generic Types, Traits and Lifetimes**
- c. **Ch. 15 - Smart Pointer; Sec. 19.1 - Unsafe Rust**

9. An overview of Python

- a. **Introduction to Python: Basic and Sequence Datatypes, Dictionaries, Control Structures, List Comprehension**
[Selected topics from The Python Tutorial <http://docs.python.org/tutorial/>, till Sec. 4.7 (Defining functions) and from Sec. 5 (Data Structures) and 6 (Modules)]
- b. **Python: Function definition, Positional and keyword arguments of functions, Functional Programming in Python, Using higher order functions: Decorators**
[Section 4.8 (More on defining functions) of The Python Tutorial
<http://docs.python.org/tutorial/>]
[Primer on Python Decorators,
<https://realpython.com/blog/python/primer-on-python-decorators/>]
- c. **Python: Classes and Instances, Single and Multiple Inheritance, Magic Methods for operator overloading**
[Sections 9 (Classes) of the Python Tutorial, <http://docs.python.org/tutorial/>]
- d. **Garbage Collection and the Global Interpreter Lock (GIL).**
[Inside the Python GIL, by David Beazley:
<http://pages.di.unipi.it/corradini/Didattica/AP-24/DOCS/InsideThePythonGIL.pdf>]