Advanced Programming [AP-2021]
Detailed Syllabus
Andrea Corradini

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-21/]. 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 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-21/DOCS/GM-ch1.pdf] of book *Programming Languages: Principles and Paradigms,* by Maurizio Gabbrielli and Simone Martini.]

2. **Runtime Systems and Introduction to the JVM**
   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]

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-21/DOCS/JBS.101.pdf]
   c. **Reflection in Java**
      [*The Java Tutorial on the Reflection API*, https://docs.oracle.com/javase/tutorial/reflect/index.html excluding *Arrays and Enumerated Types.*]
   d. **Annotations in Java**
      [*The Java Tutorial on the Reflection API*, https://docs.oracle.com/javase/tutorial/java/annotations/index.html
   e. **Software Components: the .NET framework by Microsoft**
      [Sections 15.1, 15.11, and 15.12 of **[COMP]**[1]]
   f. **Frameworks and Inversion of Control: Decoupling components; Dependency Injections; IoC Containers**

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

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

g. **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-21/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-21/DOCS/stl-tutorial-Weidl.pdf],
   e. **Generics and inheritance: invariance, covariance and contravariance in Java and other languages**
   [Covariance and Contravariance in C#, https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/covariance-contravariance/]
   [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-21/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]
   c. **Call by need in Haskell**

6. **Haskell**
   a. **Introduction to Haskell, Laziness, Basic and compounds types, Patterns and declarations, Function declarations**
   [Introduction to Haskell, by John C. Mitchell, http://pages.di.unipi.it/corradini/Didattica/AP-21/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-21/DOCS/Ch7.pdf]

d. **The Maybe constructor and composition of partial functions**

e. **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]
[Monads as Computations, https://wiki.haskell.org/Monads_as_computation]

f. **[Optional] The IO Monad**
https://en.wikibooks.org/wiki/Haskell/Understanding_monads/IO
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 the Rust programming language**
[RUST on Wikipedia: https://en.wikipedia.org/wiki/Rust_(programming_language)]
[Introduction to Rust, slides by Haozhong Zhang,
http://pages.di.unipi.it/corradini/Didattica/AP-21/DOCS/IntroToRUST.pptx]
   a. **Ownership and Borrowing**
   [Sections 4.1 and 4.2 of: https://doc.rust-lang.org/book/index.html]

9. **Scripting Languages and Python**
   a. **[Optional] Overview of Scripting Languages**
   [Scripting Languages, by Michael Scott,
   http://pages.di.unipi.it/corradini/Didattica/AP-21/DOCS/Scott-ch13.pdf]
   b. **Introduction to Python: Basic and Sequence Datatypes, Dictionaries, Control Structures, List Comprehension**
   [The Python Tutorial: till Section 4.5 and Section 5,
   http://docs.python.org/tutorial/]
   c. **Python: Function definition, Positional and keyword arguments of functions, Functional Programming in Python, Iterators and Generators, Using higher order functions: Decorators**
   [The Python Tutorial: Defining Functions, Sections 4.6 and 4.7,

https://docs.python.org/3.7/tutorial/controlflow.html - defining-functions]
[Primer on Python Decorators,
https://realpython.com/blog/python/primer-on-python-decorators/]

d. **Python: Classes and Instances, Single and Multiple Inheritance, Magic Methods for operator overloading, Modules definition and importing**
[The Python Tutorial: Sections 6 and 9, http://docs.python.org/tutorial/]

e. **The Global Interpreter Lock (GIL).**
[Inside the Python GIL, by David Beazley:
http://pages.di.unipi.it/corradini/Didattica/AP-21/DOCS/InsideThePythonGIL.pdf
]