

University of Luxembourg

Faculty of Science, Technology and Medicine

**Bachelor in Applied Information  
Technology  
Continuing Education Programme  
(BINFO-CEP)**

<https://binfo-cep.uni.lu>

Programme

Academic Year 2024-2025

Programme Director: A-Prof. Volker Müller

Version: 14.10.2024






## INTRODUCTION

The "Bachelor in Applied Information Technology - Continuing Education Programme" (BINFO-CEP) at the University of Luxembourg is a part-time lifelong learning programme that leads to an academic Bachelor's degree. BINFO-CEP aims to enhance participants' understanding of fundamental IT concepts while incorporating the latest industry trends, ensuring they acquire up-to-date knowledge and skills essential in the fast-evolving IT sector. Like its full-time counterpart, the "[Bachelor in Applied Information Technology](#)", BINFO-CEP emphasizes practical computer science techniques and tools crucial for IT professionals. This programme is offered in collaboration with the [Lifelong-Learning Center](#) of the *Chambre des Salaries (CSL)*, addressing the growing demand for continuous education in the Luxembourgish job market.






This two-year programme offers evening courses on weekdays, enabling working professionals to continue their studies while maintaining their daytime careers. It features project- and group-based learning activities, often completed as homework, giving students practical experience and opportunities for peer collaboration. The primary goal is to provide students with a refreshed and expanded perspective on key aspects of information technology. By exploring various topics, participants can deepen their understanding and broaden their expertise. The programme also recognizes and builds upon the students' existing knowledge and skills.

**Note:** Applicants must have at least six years of proven professional experience in the IT field to qualify for the programme. Alternatively, they can apply with three years of professional experience combined with a Bac+2 degree (equivalent to an associate degree). This professional experience is recognized and credited with 100 ECTS (European Credit Transfer System) credits, effectively covering the first two semesters of a traditional Bachelor's programme.






## THIRD SEMESTER PROGRAMME

	Lectures (hrs)	ECTS
Mathématiques générales 	20	4
Introduction to Programming 	42	6
Databases 	42	6
Networks  	28	4
<b>TOTAL</b>	<b>132</b>	<b>20</b>






## FOURTH SEMESTER PROGRAMME

	Lectures (hrs)	ECTS
Mathématiques discrètes 	20	4
Operating Systems 	42	6
Analyse et conception des logiciels 1 	42	6
Algorithms and Data Structures 1  	28	4
<b>TOTAL</b>	<b>132</b>	<b>20</b>

## FIFTH SEMESTER PROGRAMME

	Lectures (hrs)	ECTS
Web Programming 	42	6
Algorithms and Data Structures 2 	28	4
Analyse et conception des logiciels 2  	42	6
GUI Programming 	28	4
<b>TOTAL</b>	<b>140</b>	<b>20</b>

## SIXTH SEMESTER PROGRAMME

	Lectures (hrs)	ECTS
Software Testing  	28	4
Mobile Application Development 	42	6
Java for Enterprise Applications 	38	6
Introduction to Blockchains 	24	4
<b>TOTAL</b>	<b>132</b>	<b>20</b>

Flag signs show the primary and possibly the secondary language used in a course.

ECTS = Number of credits in the European Credit Transfer System.

**Remarks:**

- One "hour" shown in the table corresponds to one teaching unit (*unité d'enseignement*) of 45 minutes. The given number of hours is the maximal number of hours for a course, which can change due to the academic calendar for a specific semester (e.g. official national holidays, number of weeks with courses for the given semester).
- Note that the number of effective hours a student is expected to work for a course to work is in general larger than the numbers given above, since all courses include additional personal / group work and practical projects.

## COURSE DETAILS FOR SEMESTER 3

### 1. Mathématiques générales [4 ECTS]

**Objectif:** Connaître et maîtriser les bases de l'analyse réelle en une variable, en particulier savoir manipuler les fonctions usuelles.

**Learning Outcomes:** Après avoir réussi ce cours, les étudiants sont capables de:

- déterminer si une suite ou une série converge;
- manipuler les fonctions usuelles;
- déterminer les dérivées des fonctions standard;
- résoudre des problèmes mathématiques de base.

**Contenu:**

- Suites et limites

- Fonctions réelles à une variable
- Fonctions usuelles
- Dérivation
- Intégration et primitives

**Enseignant(s):** Prof. Bruno Teheux

**Prérequis:** —

**Langue:** Français

**Modalité enseignement:** Cours et TD.

**Modalités d'évaluation:** Examen partiel écrit (40%) et examen final écrit (60%).

**Ouvrage de référence:** Les références seront données pendant le cours et sur la page Moodle.

**Notes:** Il est obligatoire d'assister aux cours et faire les exercices demandés d'une séance à l'autre.

## 2. Introduction to Programming [6 ECTS]

**Objectives:** This course provides a thorough introduction to the Java programming language.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- design and realize Java applications of average complexity.
- explain basic principles underlying the Java programming language.
- apply in practice the concept of interfaces, interface implementation, and inheritance in Java programming.

**Description:** Students will learn to design and implement Java applications of average complexity. Topics covered:

- primitive types,
- variables,
- expressions,
- control flow,
- classes and objects,
- encapsulation and access control,
- inheritance and polymorphism,
- interfaces and abstract classes,
- exception handling,
- introduction to generics.

**Lecturer(s):** Prof. Steffen Rothkugel

**Prerequisites:** —

**Language:** English

**Teaching modality:** Combination of lectures and supervised practical lab sessions.

**Evaluation:** Winter semester:

- First session students: midterm exam (40%) + final exam (60%)
- Resitting students: final exam (100%)

**Summer semester:** final exam (100%)

**Literature:**

- "The Java Language Specification, Java SE Edition", James Gosling et al, Addison-Wesley, ISBN 978-0133260229, available online at: <http://docs.oracle.com/javase/specs/>

- Head First Java, 2nd edition, Kathy Sierra et al., O'Reilly Media, ISBN 978-0596009205
- Head First Object-Oriented Analysis and Design, Brett McLaughlin et al., O'Reilly Media, ISBN 978-0596008673

### 3. Databases [6 ECTS]

**Objectives:** Relational databases are the default architecture to manage, query, and analyze large volumes of (structured) data. This course introduces common concepts and structures necessary for the design and implementation of database management systems and their usage in practical applications. In addition to studying relational databases with a practical focus on learning SQL over open-source databases such as MariaDB/PostgreSQL, this course also introduces large-scale data management techniques over in-memory computing platforms such as SparkSQL.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- develop an Entity-Relationship model for a concrete data modeling problem;
- formulate queries of average complexity in SQL;
- assimilate practical SQL experience on open-source database systems;
- explain the basic concepts used in databases;
- explain the differences of NoSQL databases compared with relational databases and their relevance for "Big Data" applications.

**Description:** The course starts by introducing the basic techniques to model both entities and their relationships and object-oriented representations in a relational database schema with respective normal forms. Next, students learn to formulate relational queries in the structured query language (SQL)



and implement database constraints and triggers via PL/SQL and stored procedures. Moreover, we will have a look at advanced data-warehousing techniques for extracting, loading, and transforming (ETL) data, as well as for on-line analytical processing (OLAP) and online transaction processing (OLTP). Finally, an outlook complements the topics onto recent trends in NoSQL databases, which are highly relevant for implementing "Big Data" applications based on the Apache Hadoop, SparkSQL platforms. The course proposes a practical assignment, in which students implement a data-warehousing application by using the various platforms:

- Entity-Relationship Model (ERM) and Object Definition Language (ODL)
- Relational Data Model, Schema Design and Normal Forms, Relational Algebra
- Structured Query Language (SQL), Constraints and Triggers, Stored Procedures in PL/SQL, Embedded SQL and JDBC
- Data Warehousing, Extract-Transform-Load (ETL), Online Analytical Processing (OLAP) and Online Transaction Processing (OLTP)
- MapReduce Principle for Distributed Data Management using Apache Hadoop
- NoSQL Databases using SparkSQL

**Lecturer(s):** Dr. Ovidiu-Cristian Marcu

**Prerequisites:** —

**Language:** English

**Teaching modality:** Lectures, exercise sessions and practical homework.

**Evaluation:** One project assignment (70%) and a final written exam (30%).

**Literature:** Relevant literature and practical exercises will be announced in class and made available on the Moodle course management platform.

## 4. Networks [4 ECTS]

**Objectives:** The objective of the course is to give an introduction to TCP/IP networks. The course will provide an introduction to current network architectures, address modern application level protocols (SIP/http), routing protocols and security.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- elaborate on current network architectures.
- explain modern application level protocols used in networking.
- apply the learned background on networking in program development of medium complexity.

**Description:** The course is aligned with the standard computer networking course offered in major US universities and defined by Prof Kurose and Prof. Ross. (<http://www-net.cs.umass.edu/kurose-ross-ppt-6e/>):

- Introduction
- The Application Layer
- The Transport Layer
- The Network Layer
- The Link Layer
- Wireless and Mobile Networks
- Multimedia Networking
- Security
- Network Management

This course will use the same support material and slides.

**Lecturer(s):** Prof. Radu State

**Prerequisites:** The course assumes basic programming skills.

**Language:** English, French

**Teaching modality:** The class will be held in weekly sessions of 4 teaching units. Some of the classes will be practicals, while others will be lectures.

The written material of the course is in English, the teaching language is French.

**Evaluation:** The evaluation is based on a mid-term evaluation (practical exam) and a written final exam. The practical exam counts for 40% of the final mark.

**Literature:** Kurose, Ross: "Computer Networking: A Top-Down Approach", (6th Edition), Pearson; 6th edition (March 5, 2012). ISBN-13: 978-0132856201.

**Notes:** Students must participate to all practicals.

## COURSE DETAILS FOR SEMESTER 4

### 5. Mathématiques discrètes [4 ECTS]

**Objectif:** S'initier aux mathématiques discrètes et maîtriser les bases de la logique.

**Learning Outcomes:** Après avoir réussi ce cours, les étudiants sont capables de:

- d'appliquer les règles de logique élémentaire;
- d'utiliser les ensembles et les relations binaires;
- de résoudre des petits problèmes de dénombrement;
- de résoudre certains problèmes élémentaires de théorie des graphes;
- d'appliquer un raisonnement par récurrence;
- expliquer les principes de base des mathématiques discrètes.

**Contenu:**

- Logique élémentaire
- Dénombrement
- Techniques de preuves (récurrence, absurde)
- Selon l'avancement: arithmétique et graphes

**Enseignant(s):** Prof. Bruno Teheux

**Prérequis:** —

**Langue:** Français

**Modalité enseignement:** Cours. Il est obligatoire d'assister aux cours et faire les exercices demandés d'une séance à l'autre.

**Modalités d'évaluation:** Examen partiel écrit (40%) et examen final écrit (60%).

**Ouvrage de référence:** Les références seront données pendant le cours et sur la page Moodle.

## 6. Operating Systems [6 ECTS]

**Objectives:** The course introduces the concepts of operating systems together with the abstractions provided for developers.

**Learning Outcomes:** After successful completion of the course, students are capable to:

- identify and explain the responsibilities of an operating system;
- compare and evaluate the properties of different operating systems;
- designate the abstractions that operating systems provide;
- properly use those abstractions from within applications;
- handle heterogeneity appropriately.

**Description:** Operating systems represent sophisticated runtime platforms for all types of software. Their primary purpose is to mediate between applications and different kinds of resources. The internal workings of modern operating systems as well as the abstractions they provide for application programmers will be introduced throughout this course. Practical experiments will illustrate the theoretical concepts in the context of the Windows and Linux operating systems. Topics covered include:

- Memory management;
- Processes and threads;
- Scheduling;

- Synchronisation.

**Lecturer(s):** Prof. Steffen Rothkugel

**Prerequisites:** —

**Language:** English

**Teaching modality:** Lectures, supervised practical labs, homework.

**Evaluation:** Summer semester:

- First session students: midterm exam (30%) + final exam (70%)
- Resitting students: final exam (100%)

**Winter semester:** final exam (100%)

**Literature:**

- Andrew S. Tanenbaum: "Modern Operating Systems". 3rd Edition, Pearson Education, ISBN 978-0138134594.
- William Stallings: "Operating Systems". 6th Edition, Pearson Education, ISBN 978-0136033370.
- Abraham Silberschatz et al.: "Operating System Concepts". 8th Edition, Wiley & Sons, ISBN 978-0470128725.
- Additional material will be announced during the lecture.

## 7. Analyse et conception des logiciels 1 [6 ECTS]

**Objectives:** Provide to the student conceptual knowledge about software engineering in the large and focus on requirements analysis and specification in more details.

Learn and apply both a requirements analysis method and the UML modelling notation when performing such a software development phase. Both

the method and the notation are learnt by doing (both in-class and home-work) exercises.

**Learning Outcomes:** At the end of the course students are expected to attain the following learning outcomes :

- Know and apply requirements elicitation using “Use Cases”,
- Know and apply a subset of the UML notation to specify parts of the specification,
- Know and apply the OCL language to specify requirements.

**Description:**

1. Course presentation and general introduction, Part I: Use-Case and Use-Case Instances
2. Part I: Use-Case and Use-Case Instances (cnt'd)
3. Part II: Environment and Concept Model
4. Part II: Environment and Concept Model (cnt'd)
5. Evaluation 1
6. Part III: Operation Model
7. Part III: Operation Model (cnt'd)
8. Part III: Operation Model (cnt'd)
9. Part III: Operation Model (cnt'd)
10. Evaluation 2

**Lecturer(s):** Dr. Alfredo Capozucca

**Prerequisites:** —

**Language:** English

**Teaching modality:** Lectures and practical work.

**Evaluation:** The final grade of the student is based on the average of the evaluation 1 and 2: i.e. final grade = average (grade eval 1, grade eval 2).

**Resit:** a student resitting to the exam will do an exam which counts for 100% of the final grade.

**Notes:** Mandatory presence at each session.

## 8. Algorithms and Data Structures 1 [4 ECTS]

**Objectives:** The course is mainly intended for deepening students' knowledge of essential linear data structures: array, linked list (dynamic array), associative array, hash table. The implementation of such data structures will be discussed in detail, further familiarizing students with the underlying ideas. The course focuses on the Java programming language in examples and for implementation work in the supervised exercise sessions that complement all and each of the lectures.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- explain the concepts, properties and usage of standard linear data structures;
- implement basic operators (like insert, search, delete) on these data structures with the Java programming language.

**Description:**

- Arrays: basic operators, dichotomic search, iterative sorting algorithms.
- Linked lists: basic operators, variants.
- Associative arrays: basic operators, dictionaries.



- Hash tables: hash functions, basic operators, solutions to collision and overflow problems.

**Lecturer(s):** Prof. Denis Zampunieris

**Prerequisites:** Basic knowledge of imperative programming in Java.

**Language:** English, French

**Teaching modality:** Interleaved sequence of lectures (12 teaching units) and supervised exercise sessions (12 teaching units)

**Evaluation:** Active participation in all exercise sessions (20%) and final exam (80%).

**Literature:**

- "Introduction to Algorithms (3rd edition)", T.Cormen, C.Leiserson, R.Rivest & C.Stein, MIT Press, 2009
- "Algorithmique - Entrenez-vous et améliorez votre pratique de la programmation (exemples en Java)", L.Debrauwer, Editions ENI, 2008

**Notes:** Students must participate to all exercise sessions.

## COURSE DETAILS FOR SEMESTER 5

### 9. Web Programming [6 ECTS]

**Objectives:** The course provides with many practical examples information about server- and client-side programming languages and related frameworks popular in Web application development.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- use the PHP programming language to develop server-side components of web applications of medium complexity.
- explain some basic concepts commonly used in PHP frameworks.
- apply JavaScript for writing client-side scripts of medium complexity.
- explain current trends and techniques for the development of web applications.

**Description:** The course provides a thorough introduction to the programming languages PHP and JavaScript used for web application development. Popular frameworks for both languages and support tools are explained with a practical approach. Active student involvement through several practical projects is essential to succeed in this course. More specifically, the course covers the following topics:

- Short introduction into container based web development with **Docker**.
- Short introduction into HTML and CSS and related frameworks.
- Introduction to **PHP 8.x**: regular expressions, PHP and MariaDB.
- PHP-related tools for web development: **Composer**.
- The PHP based web development framework **Symfony**.
- Introduction to the **JavaScript** language.
- Dynamic HTML: Document Object Model (**DOM**) and JavaScript.

- **Ajax** and related technologies like **JSON**.
- New JavaScript APIs and tools for web development: Web sockets, Web storage, **npm** & **yarn**.
- JavaScript on the server-side: **Node.js**
- **RESTful Web services** and introduction to **GraphQL**.
- Common ideas found in popular JavaScript frameworks.
- Reactive programming with **React.js**
- Short Introduction to **Web Assembly** using Rust.
- Overview on Web Application Security and Web Application Performance.

**Lecturer(s):** Prof. Volker Müller

**Prerequisites:** Introduction to Programming

**Language:** English

**Evaluation:**

- **First-time students:**
  - Four practical homework assignments (20% each).
  - Final written exam (20%).
- **Repeaters:** only final written exam (100 %).

**Literature:** The used literature consists mainly of genuine specifications for the given technologies and online tutorials. More detailed information will be made available on the Moodle course website.

## 10. Algorithms and Data Structures 2 [4 ECTS]

**Objectives:** The course is the continuation of the first course about algorithms. Certain algorithms are revisited and examined in more detail. Students shall learn to use abstract data types in Java and choose the right algorithms for concrete problems.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- explain in detail common algorithms for trees and graphs;
- apply abstract types in the Java programming language providing common data structures;
- construct own algorithms by applying "divide and conquer" or "backtracking" techniques.

**Description:**

- Algorithms for trees and graphs.
- Utilisation of abstract types (lists, queues, ...) in Java.
- Useful approaches for algorithm construction:
  - Divide and conquer
  - Backtracking

**Lecturer(s):** Dr. Stefan Hommes

**Prerequisites:** Basic knowledge of programming in Java, course "Algorithms and Data Structures 1".

**Language:** English

**Teaching modality:** Lectures, practical sessions partly as homework.

**Evaluation:** 40% midterm exam / 60% final exam.

**Literature:** Relevant literature will be announced in class and made available on the Moodle course platform.

**Notes:** Students are obliged to prepare the given homeworks.

## 11. Analyse et conception des logiciels 2 [6 ECTS]

**Objectif:** Introduire les concepts, les méthodes de génie logiciel et les bonnes pratiques de programmation sous-jacentes aux « design patterns » (modèles de conception). Analyser plus en détail et mettre en œuvre concrètement en Java, une quinzaine de design patterns sélectionnés dans les trois grandes catégories classiques : les patterns de création qui donnent des solutions aux problèmes liés à l'instanciation des classes, les patterns structurels qui donnent des solutions aux problèmes liés à l'architecture d'un logiciel en classes, et enfin les patterns de comportements qui donnent des solutions algorithmiques aux problèmes de communication et de synchronisation entre objets pendant l'exécution du logiciel. Explorer quelques exemples classiques de composition de plusieurs design patterns dans une application.

**Learning Outcomes:** Après avoir réussi ce cours, les étudiants sont capables de:

- mettre en œuvre les « design patterns » en Java.
- expliquer les concepts et les méthodes de génie logiciel.

**Contenu:** Le cours est structuré en deux composantes: des cours magistraux et des séances de travaux pratiques sur ordinateur. Chaque leçon introduit deux ou trois design patterns et est suivie la semaine suivante par une séance d'exercices mettant en œuvre ces concepts.

**Enseignant(s):** Prof. Denis Zampunieris, Dr. Sandro Reis

**Prérequis:**

- Course "Introduction to Programming" (programming with Java knowledge),
- Course "Analyse et conception des logiciels 1".

**Langue:** Français, Anglais

**Modalité enseignement:**

- Cours magistraux en français avec supports en anglais
- Travaux pratiques en français / anglais / allemand / luxembourgeois avec supports en anglais.

**Modalités d'évaluation:** Un contrôle continu des connaissances lors de chaque séance de travaux pratiques (50 % de la note globale) et un examen final (50 % de la note globale).

**Ouvrage de référence:**

- "Design patterns: elements of reusable object-oriented software", E. Gamma, R. Helm, R. Johnson & J. Vissides, Addison-Wesley.
- "Design Patterns en Java", Laurent Debrauwer, ENI.
- "Design Patterns in Java", Steven J. Metsker & William C. Wake, Addison-Wesley.
- "Design patterns for dummies", Steve Holzner, Wiley Publishing.
- "Head First Design Patterns", Eric Freeman & Elisabeth Freeman, O'Reilly.

**Notes:** Les étudiants devront obligatoirement participer aux séances de TPs, toute absence non justifiée et/ou non valable empêchera l'étudiant(e) de se présenter à l'examen final.

## 12. GUI Programming [4 ECTS]

**Objectives:** Providing students with the theoretical and practical foundations of graphical user interface design and programming.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- design and realize graphical user interfaces.
- apply the learned techniques with the various explained toolkits (Java Swing, JavaFX, Qt).
- explain how event-driven programming is used in GUI programming.

**Description:** The course covers the fundamentals of graphical user interface design and programming. This includes user interface elements, layout management, event-driven programming, as well as related design patterns such as Observer and Model-View-Controller. The various concepts will be discussed and illustrated using practical examples based on different GUI toolkits such as Java Swing, JavaFX, or Qt.

**Lecturer(s):** Dr. Jean Botev

**Prerequisites:**

- Introduction to Programming
- Operating Systems

**Language:** English

**Teaching modality:** The course comprises a combination of lectures, supervised practical lab sessions (TD/TP) and homework.

**Evaluation:** Practical midterm exam accounting for 40% of the overall mark, plus written final exam.

**Literature:**

- The Definitive Guide to Java Swing, John Zukowski, Apress, ISBN: 978-1-59059-447-6 (Print), 978-1-4302-0033-8 (Online), DOI: 10.1007/978-1-4302-0033-8, available via [findit.lu](https://findit.lu)
- Pro JavaFX 2 - A Definitive Guide to Rich Clients with Java Technology, James L. Weaver, Weiqi Gao, Stephen Chin, Dean Iverson, Johan Vos, Apress, ISBN: 978-1-4302-6872-7 (Print), 978-1-4302-6873-4 (Online) DOI: 10.1007/978-1-4302-6873-4, available via [findit.lu](https://findit.lu)

- Foundations of Qt Development, Johan Thelin, Apress, ISBN: 978-1-59059-831-3 (Print), 978-1-4302-0251-6 (Online), DOI: 10.1007/978-1-4302-0251-6, available via [findit.lu](https://findit.lu)
- Online documentation of the different toolkits.

**Notes:** Mandatory presence at each session.



## COURSE DETAILS FOR SEMESTER 6

### 13. Software Testing [4 ECTS]

**Objectives:** The objective of the course is to present the three classical stages for software testing, namely unit, integration and system testing. The course is focusing on OO Java programming, as a basis for unit testing. The concepts, methods and techniques seen during the course are illustrated by recent testing frameworks widely used in the industry.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- Explain the different stages in testing.
- Use popular testing frameworks in Java-based application development.

**Description:** The course will be divided into 2 parts:

#### **Part I. Certification: Certified Tester - Foundation Level (CTFL)**

- Nowadays, software testing is identified as a key role in a company that requires specific knowledge on which we can get a certification.
- The goal of this part is to go through the ISTQB (International Software Testing Qualifications Board) standard and prepare the first level of qualification.
- The student will then be free to go through the official ISTQB certification exam to be certified (<http://www.istqb.org/>..)

#### **Part II: Software Testing principles and practice**

- Software testing : presents the overall view of the testing process, the definitions and the main issues related to the software life-cycle. Some classical testing techniques are presented.
- Unit, integration and system testing are the three remaining modules, going in depth into the issues and techniques specific to each of these

life cycle stages.

**Lecturer(s):** Prof. Michail Papadakis

**Prerequisites:** —

**Language:** English, French

**Teaching modality:** Students must deliver all exercises and homeworks, and participate to all practical sessions.

**Evaluation:** The grading will be based on a written exam composed of two parts (Part I: 30% and Part II: 70%).

**Literature:**

- "Introduction to Software Testing" - Paul Ammann and Jeff Offutt - ISBN-13: 9780521880381 – Cambridge Press – 2008.
- "Foundations of Software Testing" - Aditya Mathur - Addison-Wesley Professional – 2007
- "Software testing techniques" - B. Beizer - Van Nostrand Reinhold 1992
- "Le test des logiciels" - S. Xanthakis et Co - Editions Hermes - 2000

**Notes:** The course includes the preparation for the CTFL certification. This preparation counts for 12 hours, included in the total of 37 hours for CM.

## 14. Mobile Application Development [6 ECTS]

**Objectives:** Providing students with the theoretical and practical foundations of mobile application development.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- develop mobile apps for Android of medium complexity.
- explain the theoretical background of mobile app development.

**Description:** The course covers the fundamentals of mobile application development. This includes different development models also for wearables, including intents and activities with lifecycle and state management, data and background processing, as well as UX design and publishing aspects. The various concepts will be discussed and illustrated using practical examples based on the Android platform and Java/Kotlin.

**Lecturer(s):** Dr. Jean Botev

**Prerequisites:** Course "GUI Programming".

**Language:** English

**Teaching modality:** The course comprises a combination of lectures, supervised practical lab sessions (TD/TP) and homework.

**Evaluation:** Programming project accounting for 40% of the overall mark, plus written final exam (60%).

**Literature:**

- "Pro Android 5", Dave MacLean, Satya Komatineni, Grant Allen, Apress, ISBN: 978-1430246800 (Print), 978-1-4302-4681-7 (Online) DOI: 10.1007/978-1-4302-4681-7, available via [findit.lu](https://findit.lu).
- "Android Studio 3.0 Development Essentials", Neil Smyth, CreateSpace, ISBN: 978-1977540096 (Print).
- "Pro Android Wearables", Wallace Jackson, Apress, ISBN: 978-1430265504 (Print), 978-1-4302-6551-1 (Online) DOI: 10.1007/978-1-4302-6551-, available via [findit.lu](https://findit.lu).
- Online documentation and developer resources.

**Notes:** Mandatory presence at each session.

## 15. Java for Enterprise Applications [6 ECTS]

**Objectives:** The students will learn how the Jakarta EE framework provides a component-based approach to the design, development, assembly, and deployment of enterprise applications for the Java programming language. The students will get hands-on experience with Wildfly, a free Java application server (closely related with the popular, but commercial JBoss application server). In addition, students will learn about SpringBoot as a popular alternative framework for building Enterprise level applications with Java.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- Explain the main ideas of the Jakarta EE framework.
- Develop web front ends with Java Server Faces (JSF).
- Explain the ideas of the most important APIs included in Jakarta EE, especially on Persistence and Enterprise Beans.
- Develop Jakarta EE-based applications with the application server Wildfly.
- Implement SOAP-based and RESTful web services with Jakarta EE.
- Use SpringBoot for developing REST web services.
- Extend their knowledge on Jakarta EE and SpringBoot by autonomously reading the Jakarta EE and SpringBoot specifications and other documentation available.

### Description:

- Introduction to the main ideas of Jakarta EE.
- The web layer of Jakarta EE:
  - Basic concepts of Java Servlets,
  - Java Server Faces (JSF), named beans and important related topics,

- Component libraries for JSF.
- Enterprise Java Beans.
- The Java Persistence API.
- Several important Jakarta EE APIs like XML processing, JSON support, Web Sockets.
- The Java Message Service (JMS) and Message-driven Beans.
- Building SOAP and RESTful Web Services with Jakarta EE.
- Introduction to the main ideas of Jakarta EE Security.
- SpringBoot as alternative framework to Jakarta EE:
  - Building RESTful web services with SpringBoot.
  - SpringBoot with MongoDB.
  - SpringBoot and GraphQL.
  - Develop web applications with a MVC framework and SpringBoot Reactive.
  - Spring Native and Quarkus.

**Lecturer(s):** Prof. Volker Müller

**Prerequisites:** Java programming course.

**Language:** English

**Teaching modality:** Lectures with many practical examples and homework. All practical sessions will be done within a Docker container-based environment using the free Wildfly application server.

**Evaluation:**

- **First-time students:** The final grade will be determined with three practical either weekly or bi-weekly homework exercises (25% each). In addition, there will be a final written exam (25 %).
- **Repeating students:** only final written exam required (100 %).

**Literature:** The course mainly uses the various Jakarta EE API specifications available on the web and genuine documentation on SpringBoot. A detailed list of references will be made available on the course website.

**Notes:** This course is related with the course "Web Programming" in the fifth semester.

## 16. Introduction to Blockchains [4 ECTS]

**Objectives:** This course aims at providing a hands-on introduction to blockchain, distributed ledger and crypto-currencies.

**Learning Outcomes:** After successful completion of this course, students are capable to:

- explain the main ideas of blockchains;
- reflect on the opportunities, but also the limits of blockchain technology;
- write simple blockchain applications using the Ethereum platform;
- understand the basic theory on consensus protocols;
- develop basic monitoring tools for other blockchains (Ripple);
- understand the economic issues and incentives of a decentralized application.

**Description:** The class will provide an introduction to the underlying technology, use cases and provide hands-on programming experience with the Ripple API, Ethereum smart contract programming and Hyperledger SDK. The class will also cover the bitcoin crypto-currency, detail the underlying consensus mechanism (PoW) and introduce the other consensus algorithms (Proof of Stake, Federated Proof of Stake and PBFT).

**Lecturer(s):** Prof. Radu State

**Prerequisites:** —

**Language:** English

**Evaluation:** Evaluation will be done based on intermediate midterm exam (30% of the grade) and a final project (70%).

**Literature:** Genuine literature will be provided during the course.