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 2026-2027

Programme Director: A-Prof. Volker Müller

Version: 01.10.2026

Introduction

The Bachelor in Applied Information Technology - Continuing Education Programme (BINFO-CEP) at the University of Luxembourg is a part-time lifelong learning programme leading to an academic Bachelor's degree. It strengthens participants' understanding of core IT concepts while integrating the latest industry trends, ensuring they gain up-to-date knowledge and practical skills for today's fast-evolving IT sector. Like its full-time counterpart, the Bachelor in Applied Information Technology (BINFO), BINFO-CEP emphasizes hands-on computer science techniques and tools essential for IT professionals. Delivered in collaboration with the *Lifelong-Learning Center* of the *Chambre des Salaries (CSL)*, the programme responds to the increasing demand for continuous education in Luxembourg's job market.

This two-year programme offers evening courses on weekdays, allowing professionals to pursue their studies alongside their careers. It combines project- and group-based learning, often through homework assignments, to provide practical experience and foster peer collaboration. The programme's primary goal is to refresh and expand students' perspectives on key areas of information technology. By exploring diverse topics, participants deepen their understanding, broaden their expertise, and build on the knowledge and skills they already possess.

Starting with the current academic year, the BINFO-CEP programme has been comprehensively renewed to reflect the rapid evolution of the discipline and the changing demands of the software and technology industry. In particular, the updated curriculum incorporates the latest developments and trends in Artificial Intelligence (AI), including modern AI methods, machine learning, generative AI, and the use of AI in software development and other areas of computing. These additions ensure that students acquire both a strong foundational understanding of Computer Science and the knowledge and practical skills needed to work effectively with emerging AI technologies. The renewed programme therefore provides a forward-looking education that prepares graduates for the increasingly AI-driven technological landscape.

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
TOTAL	**132**	**20**

Fourth Semester Programme

	Lectures (hrs)	ECTS
Mathématiques discrètes	20	4
Operating Systems	42	6
AI-Augmented Software Engineering	42	6
Algorithms and Data Structures 1	28	4
TOTAL	**132**	**20**

Fifth Semester Programme

	Lectures (hrs)	ECTS
Web Programming	42	6
Algorithms and Data Structures 2	28	4
Patterns in Software Design	42	6
Human-Computer Interaction and Systems Design	28	4
TOTAL	**140**	**20**

Sixth Semester Programme

	Lectures (hrs)	ECTS
Software Testing	28	4
Mobile and Extended Reality Development	42	6
Cloud-based Applications	28	4
Introduction to Artificial Intelligence	42	6
TOTAL	**140**	**20**

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:

- Construire et manipuler des expressions booléennes (y compris des tables de vérité).
- Traduire un énoncé en langage naturel en langage symbolique, et inversement.
- Utiliser les quantificateurs existentiels et universels ainsi que leurs négations.
- Manipuler des expressions algébriques simples relatives aux nombres réels.
- Représenter une droite à partir de l'une de ses équations ; déduire une équation d'une droite à partir d'une représentation.
- Calculer des droites de régression sur un petit jeu de données.
- Calculer des dérivées élémentaires.
- Manipuler les fonctions exponentielles et logarithmiques.
- Comparer les grandes classes de complexité.
- Appliquer le codage de Huffman ; calculer la longueur moyenne du code et sa borne inférieure théorique.

Contenu:

- Logique propositionnelle et logique des prédicats.
- Les nombres : manipulations et représentations.

- La droite : représentation, équation, régression.
- La notion de fonction et ses propriétés.
- Exponentielle, logarithme et classes de complexité.
- Théorie de l'information de Shannon et codage de Huffman.

Enseignant(s): Prof. Bruno Teheux

Prérequis: —

Langue: Français

Modalité enseignement: Cours et TD.

Modalités d'évaluation: Examen final écrit (100%).

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%) and 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 online 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 (30%), three practical laboratories in class (30%), and a final written exam (40%).

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.

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.
- Kurose, Ross: "Computer Networking: A Top-Down Approach", public powerpoint slides, version 9 (June 2025): https://gaia.cs.umass.edu/kurose_ross/ppt.php

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:

- Convertir des entiers dans une base quelconque.
- Manipuler les notations et opérations ensemblistes de base.
- Démontrer une égalité ou une inclusion d'ensembles à l'aide d'un raisonnement formel simple.
- Appliquer les principes de base du dénombrement (principe additif, principe multiplicatif).
- Distinguer et calculer arrangements, permutations, combinaisons et partitions.
- Appliquer le principe d'inclusion-exclusion.
- Résoudre des problèmes de dénombrement simples issus de contextes informatiques.
- Écrire une relation de récurrence à partir d'un problème donné.
- Résoudre des récurrences simples.
- Démontrer une propriété par récurrence, en identifiant correctement l'hypothèse d'induction et l'étape d'hérédité.
- Appliquer l'algorithme d'Euclide étendu.
- Effectuer des calculs dans $\mathbb{Z}/n\mathbb{Z}$.
- Résoudre des congruences linéaires simples.
- Utiliser le petit théorème de Fermat pour le calcul modulaire.

- Appliquer les fonctions de chiffrement et de déchiffrement RSA sur des exemples fictifs.
- Représenter un ordre partiel à l'aide d'un diagramme de Hasse et en identifier les éléments minimaux, maximaux, comparables et incomparables.
- Appliquer les algorithmes de tri.

Contenu:

- Ensembles et constructions d'ensembles.
- Dénombrement.
- Récursion et récurrence.
- Principes d'arithmétique modulaire.
- Chiffrement RSA.
- Ordres et tri.

Enseignant(s): Prof. Bruno Teheux

Prérequis: —

Langue: Français

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

Modalités d'évaluation: Examen final écrit (100%).

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 (40%) and final exam (60%).
- 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. AI-Augmented Software Engineering [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 the B-Method to formally specify requirements.
- Learn the role of AI assistants to write formal specifications using the B-Method.

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

- Know the notion of state machines.
- Know how to use state machines to specify requirements.

- Know what it means that a state machine is consistent and how to determine it.
- Know and apply the B-Method to specify state machines.
- Know how to use an AI assistant to specify requirements using the B-Method.

Description:

- Introduction to Software Engineering and the role of the Formal Methods.
- State machines
- The B-Method to specify requirements
- Proof obligations

Lecturer(s): Dr. Alfredo Capozucca

Prerequisites: Knowledge on set theory and first order logic.

Language: English

Teaching modality: Interleaved lectures and practical work.

Evaluation: The final grade of all students is based on a final exam (100%).

Literature:

- Lecture notes on B (available on the Moodle's course's page).
- Schneider, Steve. The B-Method: an Introduction. New York NY: Palgrave Macmillan, 2001. Print.

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:

- **First time students:** Active participation in all exercise sessions (20%) and final exam (80%).
- **Repeating students:** Final exam (100%).

Literature:

- "Introduction to Algorithms (4th edition)", T.Cormen, C.Leiserson, R.Rivest & C.Stein, MIT Press, 2022, ISBN 978-0-262-04630-5, 3rd edition (2009) online available at https://www.cs.mcgill.ca/~akroit/math/compsci/Cormen%20Introduction%20to%20Algorithms.pdf
- "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 and Go.
- Overview on Web Application Security and Web Application Performance.

Lecturer(s): Prof. Volker Müller

Prerequisites: *Introduction to Programming*

Language: English

Evaluation:

- **First-time students:**
  - Two practical homework assignments (15% each).
  - Final written exam (70%).
- **Repeaters:** 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 course *Algorithms and Data Structures 1* and certain algorithms are revisited and examined in more detail. Students shall learn to analyze, select, and implement appropriate algorithms and abstract data types for solving computational problems. Particular emphasis is placed on trees, graphs, algorithmic complexity, and general algorithm design techniques.

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

- Explaining and analyzing common algorithms for trees and graphs.
- Evaluating the time and space complexity of algorithms using asymptotic notation.
- Selecting appropriate data structures and algorithms for concrete problems.
- Using abstract data types and standard data structures provided by Java.
- Implementing and applying graph traversal and tree algorithms.
- Designing algorithms using divide-and-conquer and backtracking techniques.
- Comparing alternative algorithmic solutions in terms of correctness and efficiency.

Description:

- Review of algorithm analysis and asymptotic complexity.
- Trees and tree algorithms:
  - Tree traversal.
  - Binary search trees.
  - Balanced trees and related structures.

- Graphs and graph algorithms:
  - Graph representations.
  - Breadth-first search (BFS).
  - Depth-first search (DFS).
  - Shortest-path algorithms.
  - Minimum spanning trees.

- Algorithm design techniques:
  - Divide and conquer.
  - Backtracking.

- Selection and practical use of appropriate data structures and algorithms in Java.

Lecturer(s): Dr. Enea Mancellari

Prerequisites: Basic knowledge of programming in Java and successful completion of *Algorithms and Data Structures 1*, or equivalent knowledge.

Language: English

Teaching modality: Lectures, practical sessions partly as homework.

Evaluation:

- **First time students:** 40% midterm exam and 60% final exam.
- **Repeating students:** 100% final exam.

Literature:

- "Introduction to Algorithms (4th edition)", T.Cormen, C.Leiserson, R.Rivest & C.Stein, MIT Press, 2022, ISBN 978-0-262-04630-5, 3rd edition (2009) online available at https://www.cs.mcgill.ca/~akroit/math/compsci/Cormen%20Introduction%20to%20Algorithms.pdf
- D. S. Myers, Data Structures and Algorithms in Java: A Project-Based Approach, Cambridge University Press, 2024. Open Access.

Additional up-to-date resources, lecture materials, and programming exercises will be provided through the Moodle course platform.

Notes: Students are expected to regularly complete the assigned exercises and homework in preparation for the practical sessions and examinations.

11. Patterns in Software Design [6 ECTS]

Objectives: The course introduces students to the concepts and best practices of software engineering centered on Design Patterns. Several individual Design Patterns and some of their possible combinations will be discussed in detail from both theoretical and practical points of view, further familiarizing students with the underlying ideas. The course focuses on the Java programming language in examples and for implementation work.

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

- apply the best practices of software engineering centered on Design Patterns.
- use individual design patterns and their combination in software development.
- explain the underlying ideas of specific design patterns.

Description: The course is structured into two components: lectures and practical computer sessions. Each lesson introduces two or three design patterns and is followed next week by an exercise session implementing these concepts.

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

Prerequisites:

- Course *Introduction to Programming* (Java programming skills).
- Course *AI-Augmented Software Engineering*.

Language: English, French

Teaching modality: Interleaved sequence of lectures and supervised practical sessions. Students must participate in all practical sessions.

Evaluation:

- **First time students:** Continuous assessment during the practical sessions (50% of the final grade) and written examination (50%).
- **Repeating students:** written examination (100%).

Literature:

- "Design patterns: elements of reusable object-oriented software", E. Gamma, R. Helm, R. Johnson & J. Vissides, Addison-Wesley, 1994, ISBN 0-201-63361-2.
- "Design Patterns en Java", Laurent Debrauwer, ENI.
- "Design Patterns in Java", Steven J. Metsker & William C. Wake, Addison-Wesley, 2006, ISBN 978-0321629944.
- "Design patterns for dummies", Steve Holzner, Wiley Publishing, 2006, ISBN 978-0471798545.
- "Head First Design Patterns", Eric Freeman & Elisabeth Freeman, O'Reilly, 2021, ISBN 978-1492078005.

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. Human-Computer Interaction and Systems Design [4 ECTS]

Objectives: The course provides students with the theoretical and practical foundations of human-computer interaction, user-centred interface design,

and interactive systems implementation.

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

- Analyse user needs and design interactive systems with a user-centred approach, taking fundamental usability, accessibility, and interaction-design principles into account;
- Design and realize graphical user interfaces using different toolkits;
- Apply event-driven programming techniques in interactive systems;
- Use relevant design patterns, including Observer and Model-View-Controller, when developing interactive systems.

Description: The course covers the fundamentals of human-computer interaction, user-centred systems design, and graphical user interface programming. This includes user needs and usability considerations, user interface elements, layout management, interaction techniques, event-driven programming, and related design patterns such as Observer and Model-View-Controller. The various concepts are discussed and illustrated through practical examples and implemented using different GUI toolkits such as Java Swing, JavaFX, or Qt.

Lecturer(s): Dr. Jean Botev

Prerequisites:

- *Introduction to Programming*
- *Operating Systems*

Language: English

Evaluation:

- **First-time students:** practical project (30%) and written final examination (70%).
- **Repeating students:** written final examination (100%).

Literature:

- Designing the User Interface: Strategies for Effective Human-Computer Interaction, Ben Shneiderman et al., Pearson, 2017, ISBN: 978-1292153919
- Java: The Complete Reference, Herbert Schildt, McGraw-Hill Education, 2018, ISBN: 978-9387432291
- Foundations of Qt Development, Johan Thelin, Apress, 2007, ISBN: 978-1590598313
- Additional references and online documentation will be provided during the course.

Course Details for Semester 6

13. Software Testing [4 ECTS]

Objectives: The objective of the course is to provide students with the theoretical and practical foundations of software testing and engage students with the recent testing frameworks that are widely used in industry.

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

- Explain the different stages of software testing.
- Being able to test an application under test using test adequacy criteria.
- Use popular testing frameworks in Java-based application development.

Description: Software testing is an expensive activity as it can consume 50% or even 60% of the total cost of the software life cycle. To reduce the software testing cost, an excessive amount of effort has been put towards objective criteria that can measure, manage, and overall reduce testing cost. To evaluate that a piece of software has been thoroughly tested, a collection of requirements is selected and checked whether they have been successfully executed with test cases. This course will present and detail the key concepts of software testing and quality assurance. It will demonstrate the related theory and will focus on software development. The course will cover practical testing techniques, applied both during software development and software maintenance.

Lecturer(s): Prof. Yves le Traon, Prof. Michail Papadakis

Prerequisites: —

Language: English, French

Teaching modality: Students must participate to all theoretical and practical sessions.

Evaluation: The grading will be based on a written exam (100%).

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 – 2013, ISBN 978-9332517660.
- "Software testing techniques" - B. Beizer - Van Nostrand Reinhold, 1992, ISBN 978-1850328803.
- "Le test des logiciels" - S. Xanthakis et Co, Editions Hermes, 2000, ISBN 978-2746200838.

14. Mobile and Extended Reality Development [6 ECTS]

Objectives: The course provides students with the theoretical and practical foundations of mobile and extended reality (XR) application development.

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

- Design and develop Android applications of medium complexity;
- Apply core principles of Android development, including application components, lifecycles, state management, data handling, and background processing;
- Create user interfaces and interactive user experiences for mobile and XR applications;
- Integrate XR features using Android development tools and frameworks;
- Evaluate technical and user-experience considerations when developing and deploying mobile and XR applications.

Description: The course covers the fundamentals of mobile and extended reality (XR) application development in the Android ecosystem. It addresses Android application architectures and development models, including wearable applications; activities, intents, application lifecycles, and state management; data handling and background processing; as well as user-interface design, interaction techniques, and publishing aspects.

The course further introduces the foundations of XR development for Android. Topics include spatial user interfaces, tracking, 3D content, device-specific constraints, and interaction in immersive environments. The various concepts are discussed and illustrated through practical examples using Android Studio, Java, Android Views and layouts, ARCore, and Android XR.

Lecturer(s): Dr. Jean Botev

Prerequisites: *Human-Computer Interaction and Systems Design*

Language: English

Evaluation:

- **First-time students:** practical project (30%) and written final examination (70%).
- **Repeating students:** written final examination (100%).

Literature:

- Android Programming: The Big Nerd Ranch Guide, Bill Phillips, Chris Stewart, and Kristin Marsicano, Big Nerd Ranch Guides, 2017, ISBN: 978-0134706054.
- Head First Android Development: A Brain-Friendly Guide, Dawn Griffiths and David Griffiths, O'Reilly Media, 2017, ISBN: 978-1491974056.
- Official Android Developers documentation and developer resources for Android, Wear OS, ARCore, and Android XR.
- Additional references and online documentation will be provided during the course.

15. Cloud-based Applications [4 ECTS]

Objectives: The course provides an introduction to both the theoretical foundations and practical applications concerning the broad area of "Cloud Computing".

The course covers both the practical development and deployment of cloud-based computing applications as well as current tools and respective application-programming interfaces (APIs) in the context of the Apache Hadoop and Spark ecosystems (cloud-based "big data" processing).

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

- Explain both the theoretical foundations and the practical usage of current cloud-computing architectures.
- Explain concepts on virtualization and application deployment into the cloud.
- Describe how different concepts concerning the modeling and management of large data collections are implemented on top of these architectures.
- Prove first-hand experience about usage of the introduced tools in AWS / Google Cloud.

Description: The course will cover two important aspects of cloud-based applications, compute-centric and data-centric applications, in particular the following topics:

- Cloud fundamentals and Cloud application architecture.
- Developing and using REST APIs - Spring Boot.
- Containers - Docker.
- Container orchestration in the cloud with Kubernetes.
- Examples for cloud providers: AWS, Google Cloud, RedHat OpenShift.
- Infrastructure as Code: Terraform.

- Distributed file systems - GFS & HDFS.
- Distributed computing principles - MapReduce.
- MongoDB and Redis as distributed NoSQL databases.
- Spark: distributed resilient data objects (RDDs), overview of streaming and machine-learning extensions.
- Event-driven applications - Kafka.
- Introduction to High Performance Computing (HPC).

Lecturer(s): Prof. Volker Müller

Prerequisites: —

Language: English

Evaluation:

- **First time students:** 2 graded exercises (15% each) and final written exam (70%).
- **Repeating students:** final written exam (100%).

Literature: The course will rely on genuine documentation available on the web, which will be made available on the Moodle course page.

16. Introduction to Artificial Intelligence [6 ECTS]

Objectives:

- Understand the foundational concepts, history, and terminology of artificial intelligence and machine learning, and their relevance to organizations and careers.
- Identify and evaluate strategic and business considerations involved in planning and adopting AI/ML solutions, including how to frame a business problem for an AI/ML approach.

- Understand the data management pipeline underlying AI/ML systems, including data collection, preparation, and quality.
- Describe how AI/ML models are trained, evaluated, and how neural networks function.
- Understand the practices involved in monitoring, maintaining, and governing deployed AI/ML models.
- Gain a working understanding of generative AI/ML, its architectures, capabilities, and applications.

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

- Apply core AI/ML concepts and vocabulary confidently in professional settings, bridging the gap between technical teams and business or non-technical stakeholders.
- Critically assess the opportunities and risks of adopting AI/ML solutions within an organization, to inform real-world strategic decisions.
- Translate a genuine business problem into a well-framed AI/ML use case — a skill directly transferable to current job responsibilities.
- Judge the data readiness and data management practices needed to support a reliable AI/ML initiative.
- Engage meaningfully with technical teams and vendors on model training, evaluation, and monitoring, without needing to write the code.
- Evaluate the capabilities and limitations of generative AI for practical adoption in the workplace.

Description: This course provides an introduction to Artificial Intelligence (AI) and Machine Learning (ML) for students without an extensive technical background who wish to develop practical skills for AI-driven careers. Following the course textbook, the course covers AI/ML fundamentals, strategic and business considerations for AI/ML development, the framing of business problems for AI/ML solutions, data management, model training, neural networks, model monitoring and maintenance, and generative AI. Lectures

combine conceptual foundations with real-world case studies and applied exercises.

This course is designed to pay off immediately for working professionals. Because the outcomes above emphasize applied literacy over abstract theory, what is learned can be put to use at work right away — in evaluating an AI vendor proposal, framing a digitalization project, or contributing to an organization's AI strategy. For professionals returning to study without interrupting their career, the course offers a structured, credentialed way to catch up with a fast-moving field, strengthens their standing in an AI-driven job market, and equips them to participate confidently in AI adoption and governance decisions within their own organization. Case studies and applied exercises are chosen to connect directly to real organizational contexts, so participants can relate course content to the problems they already face at work.

Lecturer(s): Prof. Radu State, Dr. Tatiana Petrova

Prerequisites: —

Language: English

Evaluation:

- First-time students: 2 graded exercises (15% each) and a final written exam (70%).
- Repeating students: final written exam (100%).

Literature: R. Kelly Rainer, *Introduction to Artificial Intelligence and Machine Learning*, 1st Edition, Wiley, 2025 (ISBN 9781394344710 / eBook 9781394344666).

Additional literature will be provided during the course.