University of Luxembourg

Faculty of Science, Technology and Communication

Bachelor in Applied Information Technology

Continuing Education Programme (BINFO-CEP)

https://binfo-cep.uni.lu

Programme

Academic Year 2019-2020

Programme Director: A-Prof. Volker Müller

Version: September 1, 2019

Introduction

The "Bachelor in Applied Information Technology - Continuing Education Programme" at the University of Luxembourg (BINFO-CEP) is a part-time study programme leading to an academic Bachelor degree which offers to employees the possibility to extend their knowledge on the core elements and on current trends of information technology. Similar to its full-time equivalent programme, the "Bachelor in Applied Information Technology", the BINFO-CEP focuses on applied techniques and tools that are important for professional activities in the IT domain. The BINFO-CEP is organized in cooperation with the Lifelong-Learning Center of the *Chambre des Salaries (CSL)* to satisfy the growing demands of the Luxembourgish market for continuous education offered to its employees.

The objective of the programme which lasts over 2 years is to provide to the students a renewed and extended view on different core aspects of IT together with explanations on current trends and hot topics in computer science. A typical student in the programme has some background in information technology, either through some previous study, or based on professional experience. The programme builds upon this expertise and extends it with additional information about recent developments in information technology.

For an easier combination of the professional life with this educational activity, all courses are organized on evenings of week days, but additional homework and project work are expected from all students. Most of the courses are highly depending on either individual or group projects to strengthen the practical experience gained in the courses.

Note: The main requirement to enter the programme is proven professional experience in an IT-related domain of at least 6 years, or 3 years together with a Bac+2 degree. This professional experience is acknowledged with a total of 100 ECTS (credits in the "European Credit Transfer System") and replaces the first two semesters of courses in a typical bachelor programme. As a consequence, the programme description for the BINFO-CEP starts only with semester 3.

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
Analyse et conception des logiciels 1	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
Analyse et conception des logiciels 2	42	6
GUI Programming	28	4
TOTAL	**140**	**20**

Sixth Semester Programme

	Lectures (hrs)	ECTS
Software Testing	20	4
Mobile Application Development	42	6
Java for Enterprise Applications	42	6
Introduction to Blockchains	28	4
TOTAL	**132**	**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:

- 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): Dr. 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: Midterm exam (40%), final exam (60%).

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 data. This course provides an introduction into common problems related with databases and usage of databases in practical applications.

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;
- explain the basic ideas used in distributed 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 as well as 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 to 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). The topics are complemented by an outlook onto recent trends in NoSQL databases and key-value stores, which are highly relevant for the implementation of "Big Data" applications, based on the Apache Hadoop, HIVE and HBase platforms. The course is organized with 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), Business-Intelligence Algorithms and Applications
- MapReduce Principle for Distributed Data Management using Apache Hadoop
- NoSQL Databases and Key-Value Stores using Apache HIVE and HBase

Lecturer(s): Prof. Martin Theobald

Prerequisites: —

Language: English

Teaching modality: Lectures, exercise sessions and practical homework.

Evaluation: One graded assignment (50%) and a final written exam (50%).

Literature: Relevant literature 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): Dr. 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: Midterm exam (30%), written final exam (70%).

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]

Objectif: Provide to the student conceptual knowledge about software engineering in the large, and focus on requirements analysis and software design in more details.

An individual project is at the core of the course during which the students have to deliver an update of the analysis and design deliverables corresponding to a new version of a concrete existing software.

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

- perform a requirements analysis and a software design in a practical software engineering project;
- the MESSIR academic requirements analysis method;
- apply the UML modeling notation;
- use a software engineering environment for a software engineering project following an agile process;

- share practical experiences of an agile software development process.

Contenu:

Part 1

- Introduction
- Course objectives and material
- iCrash presentation with its variants

Part 2

Tools Introduction:

- Eclipse
- LaTeX
- Design Document
- Template
- UML design tools

iCrash Design and Implementation:

- iCrash H5: presentation and demo
- Design overview
- Implementation overview

Part 3

- Analysis concepts
- Work on student's project

Part 4

- Design concepts
- Work on student's project

Part 5	• Analysis concepts • Work on student's project
Part 6	• Work on student's project
Part 7	• Analysis concepts • Work on student's project
Part 8	• Design concepts • Work on student's project
Part 9	• Project presentations

Enseignant(s): Dr. Alfredo Capozucca

Prérequis: —

Langue: Français, Anglais

Modalité enseignement: Lectures and directed work.

Modalités d'évaluation: Three sprints are evaluated. For each sprint the analysis and design deliverables are provided and presented using screencast videos. All 6 videos have equal weight. In addition, the student can obtain bonus points by providing the implementation of the software next version, if he/she decides to provide such a deliverable.

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 consists of two parts, each covering a popular programming language in web development. Many practical exercises are shown in class and made available to students for self-study. These examples are all provided in a virtualized environment built with the Docker framework:

- Short introduction to Docker
- Basics of HTML5 and CSS3
- Introduction to **PHP**:
  - PHP Basics
  - PHP and relational Databases (MySQL)
  - PHP and NoSQL Databases (MongoDB)
  - PHP-based frameworks (Symfony)
- Introduction to **JavaScript (JS)**:
  - JS Basics
  - JS and the Document Object Model
  - JS and Ajax

  - JS framework JQuery
  - Introduction to JS for the server-side: NodeJS

Lecturer(s): Prof. Volker Müller

Prerequisites: Introduction to Programming

Language: English

Evaluation:

- Three best out of four practical assignments (20% each).
- Final written exam (40%).

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 part about algorithms. Certain algorithms are revisited and examined in more detail. Students will learn to use abstract types in practice 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. Raphaël Frank

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 oeuvre 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, Mr. Sandro Reis

Prérequis:

- "Introduction to Programming" (Java)
- "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 intermédiaire des connaissances par un examen partiel en travaux pratiques (30% de la note globale) et un examen écrit final (70% 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
- 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
- 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
- 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:

1. **Part I: Software Testing principles and practice**
   - (1) 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.
   - (2) Unit, (3) integration and (4) system testing are the three remaining modules, going in depth into the issues and techniques specific to each of these life cycle stages.

2. **Part II. Towards 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/).

Lecturer(s): Dr. Michail Papadakis, Mr. Dietmar Gehring

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 combination of two written exams (Part I: 60-70% and Part II: 30-40%). The practical exams may be taken into account for the final grade.

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: Practical midterm exam 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.
- "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.
- Online documentation and developer resources.

Notes: Mandatory presence at each session.

15. Java for Enterprise Applications [6 ECTS]

Objectives: The students will learn how Java EE (Java 2 Enterprise Edition) provides a component-based approach to the design, development, as-

sembly, and deployment of enterprise applications. The different APIs of Java EE will be examined in theory and practice, with special focus on the value of Web services for realizing service-oriented architectures. The students will also get practical experience with Wildfly, a free Java application server (closely related with the popular, but commercial JBoss application server).

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

- explain the main ideas of the Java EE framework.
- develop web front ends with the Java Server Faces framework (JSF).
- explain the ideas of the most important APIs provided, especially on Persistence and Enterprise Beans.
- develop Java EE-based applications with the application server Wildfly.
- implement SOAP-based and RESTful web services with the provided respective Java EE APIs.
- extend their knowledge on Java EE by autonomously reading the Java EE specification and other documentation available.

Description:

- Introduction to the main ideas of Java EE (Java 2 Enterprise Edition).
- The web layer of Java EE:
  - Basic concepts of Java Servlets,
  - Java Server Pages (JSP),
  - JSTL and JSP custom tags,
  - Java Server Faces (JSF) and managed beans.
- Enterprise Java Beans.
- The Java Persistence API.
- Important Java EE APIs like XML processing, JSON support, Web Sockets.
- The Java Message Service (JMS) and Message-driven Beans.

- Introduction to Java EE Security.

Lecturer(s): Prof. Volker Müller

Prerequisites: Java programming course.

Language: English

Teaching modality: Lectures and practical sessions. The practical sessions will be done within a Docker container-based environment using the free Wildfly application server. The last practical session ends with a graded student project using Java EE technology.

Evaluation: The final grade will be determined by the practical project (50%) and a final exam (50%).

Literature: The course mainly uses the various Java EE component specifications available on the web. A detailed list of references will be made available in the course website.

Notes: This course is closely related with the course "Web Programming" of 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.