1st semester	**2nd semester**	**3rd semester**		**4th semester**
Obligatory courses				
Ad Hoc Networks (2+2+1) Tamás LUKOVSZKI	Component-based software development (2+2+1) László Zsolt VARGA	Lab I-II.(8 kr.)		Lab III-IV. (8 kr.)
Data mining and information retrieval (2+2+1) András BENCZÚR	Interactive media design and development (2+2+1) Márta TURCSÁNYI-SZABÓ	Building distributed systems (2+2+1) Tamás KOZSIK	Advanced cryptography (2+2+2) Péter SZIKLAI	Thesis (20 kr.)
Type models (2+0+0) László Zsolt VARGA	Advanced functional programming (2+2+1) Zoltán HORVÁTH	Advanced Java programming (2+2+1) Tamás KOZSIK	Applied cryptography project seminar (2+2+2) Péter SZIKLAI	
Synthesis and verification (2+0+1) Tibor GREGORICS	Agile project management in informatics (2+2+1) Zoltán ISTENES	Analysis of distributed systems (2+2+1) Máté TEJFEL	Cryprography protocols (2+2+0) Attila KOVÁCS	
High assurance object oriented software engineering (2+2+1) Sándor SIKE	Formal semantics (2+0+1) Zoltán HORVÁTH	Design of distributed systems (2+2+1) Zoltán HORVÁTH	Cryptography and its applications (2+2+0) Attila KOVÁCS	
Web engineering (2+2+1) Zoltán ILLÉS	Formal methods in software development (2+2+1) Zoltán ISTENES			
Optional courses				
Optional course (2+2+1)	Optional course (2+0+0)	Optional course (2+2+0)		
Total credits				
30 credits	**30 credits**	**32 credits**		**28 credits**

Name of the course: Ad Hoc Networks			
Faculty member responsible for the course: Dr. Tamás Lukovszki, associate professor			
Responsible department: Faculty of Informatics, Department of Information Systems			
Total credits: 5			
Total hours: 5			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	
Topics: Ad hoc networks do not use any extra infrastructure. The nodes of the network use a wireless communication interface and communicate directly and provide the routing necessary to deliver messages over multiple hops. We discuss medium access, routing algorithms, and methods dealing with the mobility of participants. The topics of the course: Modeling networks, capacity of wireless networks, topology control, routing, distributed localization, energy, dilation, congestion, mobility models.			
Literature: H. Karl and A. Willig: Protocols and Architectures for Wireless Sensor Networks. Wiley, ISBN: 978-0-470-09510-2, 2005. Y. Wang: Topology Control for Wireless Sensor Networks. Book Chapter of Wireless Sensor Networks and Applications, Series: Signals and Communication Technology, edited by Li, Yingshu; Thai, My T.; Wu, Weili, Springer-Verlag, ISBN: 978-0-387-49591-0, 2008. XiuZhen Cheng, Xiao Huang, and Ding-Zhu Du (Editors): Ad Hoc Wireless Networking, Kluwer Academic Publishers, 2004. X.-Y. Li and Y. Wang: Wireless Sensor Networks and Computational Geometry. Book Chapter of Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, CRC Press, edited by Mohammad Ilyas et al., 2004. Ch. Scheideler: Overlay Networks for Wireless Systems. Book Chapter of Performance Analysis of Mobile and Ad Hoc Networks, Wireless Networks and Mobile Computing Series, Vol. 7, Nova Science Publishers, edited by Chansu Yu, 2007.			

Name of the course: Data mining and information retrieval			
Faculty member responsible for the course: Dr. András Benczúr, professor			
Responsible department: Faculty of Informatics, Department of Information Systems			
Total credits: 5			
Total hourse: 5			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	
Prerequisites: The course requires basic knowledge in calculus, probability theory, and linear algebra. Knowledge of graphs and basic algorithms is an advantage. **Topics:** The aim of the course is to provide a basic, but comprehensive introduction to data mining. By the end of the course, students will be able to build models, choose algorithms, implement and evaluate them.			

Detailed Program and Class Schedule:

1. Motivations for data mining. Examples of application domains. Methodology of knowledge discovery in databases (KDD) and data mining (DM). Formulation of main problems of data mining.
2. Understanding data: preparation and exploration. Sampling.
3. Basics of classification. Concepts of training and prediction. Decision trees.
4. Models and algorithms for classification: k-NN, naïve-Bayes. Measuring quality and comparison of classification models.
5. Introduction to the WEKA data mining software. Classification with WEKA.
6. More models and algorithms for classification: neural networks, linear separation methods, support vector machine (SVM).
7. Basics of cluster analysis. Type of variables, measuring similarity and distances. Partitioning clustering algorithms, *k*-means, *k*-medoids.
8. Introduction to frequent itemset mining. The APRIORI algorithm. Applications for finding association rules.
9. Advanced classification methods: Bagging, boosting, AdaBoost.
10. Support Vector Machine. Kernel methods, graph kernels. Protein function prediction.
11. Dimensionality reduction by spectral methods, singular value decomposition, low-rank approximation.
12. Search engines, web information retrieval, PageRank and beyond.

Literature:

Pang-Ning Tan, Michael Steinbach, Vipin Kumar: Introduction to Data Mining, Addison-Wesley, 2006.
Jiawei Han és Micheline Kamber: Data Mining: Concepts and Techniques, 2^{nd} ed., Morgan Kaufmann Publishers, 2006.
T. Hastie, R. Tibshirani, J. H. Friedman: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer-Verlag, 2001.

Name of the course: Type models

Faculty member responsible for the course: Dr. László Zsolt Varga, associate professor

Responsible department: Faculty of Informatics, Department of Software Technology And Methodology

Total credits: 2

Total hours: 2

Type of the course	lecture	practice	consultation
Hours per week	2		
Type of testing	exam		

Topics:

Object oriented programming and development. Abstract data type. The algebraic theory of abstract type. Type specification methods. Analyzis of type specifications. Concrete data type. Datatype class morphism. Information hiding. Reusing, inheritance, aggregation. Synchronization interface of data types.

Literature:

Krisztof R. Apt, Ernst-Rüdiger Olderog: Verification of Sequential andConcurrent Program (Springer-Verlag, 1997, ISBN 0-387-94896-1)
Williem-Paul de Roever et al.: Concurrency Verification (CambridgeUniversity Press, 2001, ISBN 0-521-80608-9)

Recommended literature:

Ehrig H. Mahrch B.: Fundamentals of Algebraic Specification 2, Module Specification and Constraints (Springer-Verlag, 1990, ISBN-0-387-51799-5)
Jacques Loeckx, Hans-Dieter Ehrich, Markus Wolf: Specification of Abstract Data Types (John Wiley and Sons, ISBN0-471-95067-X, 1996.6)
Annabelle Mclver, Caroll Morgan ed. Programming Methodology, Monographs in computer

science (Springer-Verlag, 2003, ISBN-0-387-95349-3)

Name of the course: Synthesis and verification			
Faculty member responsible for the course: Dr. Tibor Gregorics, associate professor			
Responsible department: Faculty of Informatics, Department of Software Technology And Methodology			
Total credits: 3			
Total hours: 3			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2		1
Type of testing	exam		
Topics: Introduction. Basic notions of programs. The syntax and semantics of nondeterministic programs. The basic notion of correct programs. The Floyd method for proving partial correctness of flow charts programs. Examples. The Floyd method for proving total correctness of flow charts programs: partial correctness + termination. Examples. Formal definition of Floyd methods. Examples. Partial and total correctness of structured programs by Hoare methods. Examples. The Hoare's methods are corrects and sounds. Theorems. The basic notions of parallel and concurrent programs. Hardware architectures and software architectures. Strategies of implementations and languages tools. Examples. Process interactions. The correctness of concurrent programs. Behavioural analysis of concurrent programs using shared variables: busy waiting. Examples. Behavioural analysis of concurrent programs using semaphores, monitors, resources, remote procedure calls, message passing. Examples. Owicki's and Gries's method for proving the partial correctness of parallel programs. Examples. Owicki's and Gries's method for proving the total correctness of parallel programs. Free from deadlock and starvation, problems of termination. Examples. Formal derivation of weakly and strongly correct concurrent programs. Examples. A method for solving synchronization problems of concurrent programs. The implementation methods of derived abstract programs using semaphores: changing variables, split binary semaphores, passing the baton. Examples. Contracts and proofs. Contracts as a way of modeling a collection of agents. Summary.			
Literature: Kröger, F.: Temporal Logic of Programs (Springer-Verlag, 1987) McIver, A., Morgan, C.: Programming Methodology (Springer-Verlag, 2003)			

Name of the course: High assurance object oriented software engineering			
Faculty member responsible for the course: Dr. Sándor Sike, associate professor			
Responsible department: Faculty of Informatics, Department of Software Technology And Methodology			
Total credits: 5			
Total hours: 5			
Type of the course	**lecture**	**practice**	**consultation**

Hours per week	2	2	1
Type of testing	exam	practice	
Topics: Lifecycle of software development. Models for software development. Object oriented analysis and design by UML diagrams. Architecture. Patterns (OO desing patterns, architectural patterns, concurrent patterns, anti patterns, refactoring patterns). Patterns in software development.			
Literature: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns - Elements of Reusable Object-Oriented Software (Addison-Wesley Longman, Inc. 1995) I. Sommerville: Software Engineering (Addison-Wesley 8th edition, 2007) J. Warmer, A. Kleppe: The Object Constaint Language, Precise Modeling with UML (Addison-Wesley, 1999) Mark Grand: Patterns in Java Vol. 1-2 (John Wiley & Sons, 1999)			

Name of the course: Web engineering			
Faculty member responsible for the course: Dr. Zoltán Illés, associate professor			
Responsible department: Faculty of Informatics, Department of Media & Educational Informatics			
Total credits: 5			
Total hours: 5			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	
Topics: This curriculum introduces the students with the modern, state-of-the-art client and server side web technologies, methodologies of web engineering, the programming and design patterns, especially with the web service oriented architectures. By the end of the course the student has a global overview of the up-to-date web trends and technologies, and, with the help of them, is able to develop a web application and web information systems. Introduction to Web Technologies and Web Engineering: specialties, characteristics, categories of web applications. Web Architectures: multi-tier, data-centric architectures, Requirement Analysis of Web Applications Specialties of Large Enterprise and Small and Medium Enterprise Web Applications Development Process of Web Applications Model-Based Web Application Design and Development, WebML			

Testing, Quality Management. Design of Web 2.0 és Enterprise 2.0 Applications Web Business Models Web project management Design of Mobile Web Applications Semantic Web Applications, integration to Web Information Systems Web Application Models, Cloud computing Service Oriented Architectures, Web Information Systems
Literature: Kappel, G., Pröll, B., Reich, S., Retschitzegger W. (Eds.): Web Engineering: The Discipline of Systematic Development. John Wiley & Sons Inc., Chichester (2006). Mendes, E., Mosley, N. (Eds.): Web engineering. Springer-Verlag, Berlin (2005). Murugesan, S., Deshpande, Y. (Eds.): Web Engineering: Managing Diversity and Complexity of Web Application Development. LNCS 2016, Springer-Verlag, Berlin (2001).

Name of the course: Component-Based Software Development			
Faculty member responsible for the course: Dr. László Zsolt Varga, associate professor			
Responsible department: Faculty of Informatics, Department of Software Technology And Methodology			
Total credits: 5			
Total hours: 5			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	
Topics: Component-Based Software Development Introduction, basic notions. The notion of software development model. The UML as a primary model-based notation for all analysis and design activities. The notions of meta model and verified components. The tasks and types of the component models. The interface description language (IDL). The notion and roles of the middleware. The short overview of CORBA, COM/DCOM and JavaBeans/EJB. The notions and types of software architectures. The relationships of reference models, architectural patterns, reference architectures and software architectures. The main characteristics of the J2EE/EJB architecture. Principles of the KobrA Method. Context realization defines the environment of the target system by enterprise or business process model, by usage model, by structural model and by interaction or behavioural model. The main descriptive artifacts in a component specification are a functional model, a behavioral model and a structural model The main descriptive artifacts in a component realization are an interaction model, a behavioural or algorithmic model and a structural model. COTS component: a commercial off-the shelf component is a ready-to-use physical component from a third party that can be incorporated into an application. SYNTHESIS: a tool for creating correct system from COTS components. The new version of the SYNTHESIS tool. Propositional Temporal Logic: sytax and semantics. Axiomatization of Propositional Temporal Logic. First-Order Temporal Logic and its Semantics. Temporal semantics of concurrent programs. A			

correctness proof method: the sometime system. The tableau-based methods for synthesizing correct concurrent programs. Model checking as an automatic technique for verifying finite state concurrent systems Component verification by model checking
Literature: Bass, L., Clements P., Kazman R.: Software Architecture in Practice (Addison-Wesley, 2003) Clarke, E. M. Jr., Grumberg, O., Peled, D. A.: Model Checking (The MIT Press, 1999) Gross, H-G.: Component-based Software Testing with UML (Springer-Verlag, 2005) **Recommended literature:** Kröger, F.: Temporal Logic of Programs (Springer-Verlag, 1987) McIver, A., Morgan, C.: Programming Methodology (Springer-Verlag, 2005) Meyer, G.B.: Object-Oriented Software Construction, Second edition (Prentice Hall, 1997)

Name of the course: Interactive Media Design and Development			
Faculty member responsible for the course: Márta Turcsányi-Szabó, associate professor			
Responsible department: Faculty of Informatics, Department of Media & Educational Informatics			
Total credits: 5			
Total hours: 5			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	
Topics: The course introduces Human–Computer Interaction (HCI) involving the study, planning, and design of the interaction between people (users) and computers. Its aim is to understand the theoretical basics of Perception, Multimedia design, Information Visualization, Interaction Design, the Virtual Continum, Serious Games, Tangible, Collaborative, Location-based, and Gesture-based technologies, etc.) and recent innovations in these areas. Activities involve the exploration of emerging interactive technologies designed for demonstration, education, entertainment, navigation, narrative, support …etc. purposes and their variety of creative applications in different disciplines and user interest groups. Students from different disciplines form groups to design and implement a specified innovative project that could well serve the basis of an industrial entrepreneurship.			
Literature: C. Ware. Information Visualization - Perception for Design. (ed 3) 536 pp. Morgan Kaufmann. 2012. ISBN 978-0-12-381464-7 Ed. Ioannis Deliyannis, Interactive Multimedia, ISBN 978-953-51-0224-3, Hard cover, 312 pages, Publisher: InTech, Chapters published March 07, 2012 under CC BY 3.0 license OpenAccess: http://www.intechopen.com/books/interactive-multimedia Lester Madden, Professional Augmented Reality Browsers for Smartphones: Programming for Junaio, Layar and Wikitude (Wrox Programmer to Programmer) ISBN-13: 978-1119992813 L. Annetta and S. C. Bronack, (eds.), Serious Educational Game Assessment: Practical Methods and Models for Educational Games, Simulations and Virtual Worlds, 1–18. © 2011 Sense Publishers. ISBN: 978-94-6091-327-3 (paperback) The Functional Art: An Introduction to Information Graphics and Visualization (Peachpit/Pearson			

Education, 2012): http://www.thefunctionalart.com/ ISBN-13: 978-0321834737

Ed. Xin- Xing Tang, Virtual Reality - Human Computer Interaction, ISBN 978-953-51-0721-7, Hard cover, 306 pages, Publisher: InTech, Chapters published September 05, 2012 under CC BY 3.0 license, OpenAccess: http://www.intechopen.com/books/virtual-reality-human-computer-interaction

Recommended literature:
The Encyclopedia of Human-Computer Interaction, 2nd Ed. At: http://www.interaction-design.org/books/hci.html

Journal of Virtual World Research: http://jvwresearch.org/

Horizon Reports: http://www.nmc.org/horizon-project

Papers submitted to conferences:
- Museums and the Web: http://www.museumsandtheweb.com/
- CHI: http://chi2013.acm.org/
- iED: http://europe.immersiveeducation.org/events/ied-europe-summit-2012
- DIS: http://www.dis2012.org/
- ISMAR: http://ismar2011.vgtc.org/

Name of the course: Advanced Functional Programming

Faculty member responsible for the course: Dr. Zoltán Horváth, professor

Responsible department: Faculty of Informatics, Department of Programming Languages And Compilers

Total credits: 5

Total hours: 5

Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	

Topics:
Algebraic types, type classes.

Higher-order types, existential types.

Uniqueness typing.

Dynamics, generic programming.

Purely functional data structures.

Parallel and distributed programming.

Combinators, combinator libraries.

Monadic programming.

Interactive programs, Functional Reactive Programming.

Embedded domain-specific languages.

Literature:
Koopman, P., Plasmeijer, R., van Eekelen, M., Smetsers, S. *Functional Programming in Clean,* 2002.
Plasmeijer, R., van Eekelen, M., von Groningen, J. *Clean Language Report 2.2*, December 2011.
Hudak, P. *The Haskell School of Expression, 1st Edition.* Cambridge University Press, February

2000.
Gibbons, J., de Moor, O. *The Fun of Programming (Cornerstones of Computing).* Palgrave Macmillan, June 2005.
Hutton, G. *Programming in Haskell.* Cambridge University Press, 2007.
Thompson, S. *Haskell: The Craft of Functional Programming, 3rd Edition.* Addison-Wesley, June 2011.
Marlow, S. *Parallel and Concurrent Programming in Haskell.* Proc. of the 4th Central European Functional Programming School, CEFP 2011, Budapest, Hungary, June 2011, Revised Selected Papers.
Recommended literature:
O'Sullivan, B., Stewart, D., Goerzen, J. *Real World Haskell, 1st Edition.* O'Reilly Media, November 2008.
Lipovaca, M. *Learn You a Haskell for Great Good! – A Beginner's Guide, 1st Edition.* No Starch Press, April 2011.

Name of the course: Agile project management in informatics			
Faculty member responsible for the course: Dr. Zoltán Istenes, associate professor			
Responsible department: Faculty of Informatics, Department of Software Technology And Methodology			
Total credits: 5			
Total hours: 5			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	

Topics:
Classical Project Management Basic
Agile Introduction
Scrum overview
Roles, responsibilities
Scrum events, meetings, environment
Artifacts, documents
Product backlog, Sprint backlog
Sprint planning, running, review, retrospective
Estimation, Velocity
Backlog grooming
Release planning
Different burndown charts and agile metrics
Scrum of Scrums
Scrum simulation
Areas of application, scaling Scrum
Distributed projects
Agile transformation steps
Common pitfalls
Other topics
Lean and agile?
Kanban
Kanban vs Scrum
Exercises
Requirement engineering basics and refresher

User stories - introduction User stories - fine tuning Exercises

Name of the course: Formal semantics			
Faculty member responsible for the course: Dr. Zoltán Horváth, professor			
Responsible department: Faculty of Informatics, Department of Programming Languages And Compilers			
Total credits: 3			
Total hours: 3			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2		1
Type of testing	exam		

Topics:
Introduction: motivation, approaches to semantics definitions
Translational semantics, attribute grammars and their applications
Denotational and operational semantics of expressions
Natural semantics of imperative statements
Structural operational semantics of imperative statements
Semantics of abort, nondeterministic and parallel execution
Denotational semantics of imperative statements
Domain and fixed point theory
Semantics of functional language elements
Modeling blocks and procedures
Modeling exceptions
Full abstraction

Classification of semantic descriptions. Attribute grammars.
Translational semantics using attribute grammars.
Two-level grammar defining the syntax and semantics of programming languages.
The semantics of arithmetic expressions. Semantics of binary numbers. Principles of compositional definitions and structural induction. Properties of the semantics: free variables, substitutions.
Denotational semantics. Direct style semantics. Requirements ont he fixed point.
Fixed point theory, definitions, theorems.
Direct style semantics: existence.
Extensions of While language with blocks declaring local variables and procedures. Examples.
Continuation style semantics for While language.
Operational semantics. Natural semantics. The derivation tree. The notion of semantically equivalence. Examples.
Structural operational semantics and its properties. An equivalence result: For every statement S of While language we have S_{ns}[S] = S_{sos}[S].
Extensions of While language: abortion, non-determinism, parallelism. Operational semantics of a collection of agents that work within the limits set by the contract. Examples.
For every statement S of While language the structural operational semantics and the direct style denotational semantics are equivalent. Theorems.
The operational approach: Vienna Definition Language (VDL).
Summary.

Literature:
Hanne Riis Nielson and Flemming Nielson: Semantics with Applications - A Formal Introduction (John Wiley & Sons, 1992)
Kenneth Slonneger and Barry L. Kurtz: Formal Syntax and Semantics of Programming Languages (Addison Wesley Longman, 1995)
Glynn Winskel: The Formal Semantics of Programming Languages - An Introduction (Foundations of Computing Series, MIT Press, 1993)
John C. Reynolds: Theories of Programming Languages (Cambridge University Press, 1998)

Name of the course: Formal methods in software development

Faculty member responsible for the course: Dr. Zoltán Istenes, associate professor

Responsible department: Faculty of Informatics, Department of Software Technology And Methodology

Total credits: 5

Total hours: 5

Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	

Topics:
Characteristics of the formal methods, their types and their place in the software development. Formal specification methods. Specification languages and systems. Theorem proving systems. Formal development. Basics of the B-method, main characteristics, use on simple problems. Specification, refinement, implementation, poof of correctness. Description of data, description of methods, architecture, components. Code generation. Event-B.
Tools: AtelierB, Click’n’prove, ProB, B4free, Rodin.

Literature:
Jean-Raymond Abrial:The B Book - Assigning Programs to Meanings (Cambridge University Press, 1996)
Wordsworth J.: Software Engineering with B (Addison-Wesley, 1996)

Name of the course: Building distributed applications

Faculty member responsible for the course: Dr. Tamás Kozsik, associate professor

Responsible department: Faculty of Informatics, Department of Programming Languages And Compilers

Total credits: 5

Total hours: 5

Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	

Topics:
The course presents some important application domains for distributed programming, with special regard to present software industry challenges and scientific computations. After the completion of the course the students will not only understand the theoretical issues of distributed computing, but they will also be capable of designing and implementing distributed applications in general, and distributed object systems in particular. They will also learn

common technologies used in the software industry. The following topics will be addressed (related technologies that can be used for illustration purposes are in parentheses). Multi-tier application model: Modularization of large software systems, optimal use of distributed architectures in the design of the components (with respect to efficiency and high availability). Transactional applications backed by information systems. (Java EE, JDBC, JPA, JTA) Remote Procedure Call: (Java RMI, EJB) Message-based communication: (JMS, PVM/MPI) Web-programming: Web-applications (Java servlet, JSP, JSF) , web-services (JAX-WS) Component lookup: (JNDI, Jini). Code mobility: (Java applet) Grid systems: fulfilling high computational requirements. Aspect-oriented programming: Used in the implementation of the above technologies. (AspectJ)
Literature: Jendrock, E., Ball, J., Carson, D., Evans, I., Fordin, S., Haase, K.: The Java EE 5 Tutorial, Third Edition (Addison-Wesley, 2007) http://java.sun.com/javaee/5/docs/tutorial/doc/ Foster, I.: The Grid: Blueprint for a New Computing Infrastructure, 2nd Edition (Morgan Kaufmann, 2004)

Name of the course: Advanced Java Programming			
Faculty member responsible for the course: Dr. Tamás Kozsik, associate professor			
Responsible department: Faculty of Informatics, Department of Programming Languages And Compilers			
Total credits: 5			
Total hours: 5			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	
Topics: The purpose of the course is to acquire knowledge on, and enhance competence in, Java Standard Edition, beyond the fundamental language concepts and standard libraries. - Generic definitions - Annotations - Reflection - Multithreading - Memory management, garbage collection - Input-output, serialization - Database management and persistence: JDBC and the fundamentals of JPA - Network programming: TCP and UDP; HTTP - Program design principles and best practices - Exceptions, assertions - Logging and testing			
Literature: James Gosling, Bill Joy, Guy Steele, Gilad Bracha. The Java Language			

Specification, Third Edition. Addison-Wesley, 2005. ISBN 0-321-24678-0 Linda DeMichiel, Michael Keith. JSR 220: Enterprise JavaBeans, Version 3.0, Java Persistence API. Sun Microsystems, Inc., 2006.

Name of the course: Analysis of Distributed Systems			
Faculty member responsible for the course: Dr. Máté Tejfel, assistant professor			
Responsible department: Faculty of Informatics, Department of Programming Languages And Compilers			
Total credits: 5			
Total hours: 5			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	1
Type of testing	exam	practice	
Topics: The goal of the subject is to give an overview for the student about how can we explain the parallel behaviour by algebraic methods and Petri-nets, and how work applications based on that models in practice. The basic concepts of the course are processes, computational processes, parallelism, operations of processes, compositions of processes and properties of processes (liveness, deadlock-free, etc.). The theory of Petri-nets is explored more partially with many modelling example. The behavioural and structural properties, methods of analysis, famed subclasses and relationships between these subclasses are investigated. We define theorems about liveness, safetyness and reachability and present transformation, which preserve these properties. The course introduces the Petri-boxes, a special class of Petri-nets, which help us to model the program structures (sequences, branches and loops). Some tools for simulation and analysis of Petri-nets are also investigated. The second part of the course introduces the theory of algebraic models through a given example. The properties of the models, the methods of descriptions of processes and the possible compositions are examined. The denotational, operational and axiomatic semantics of the model is given and the relationships of these different descriptions are investigated. Teaching methods: There will be lectures introducing the formal specification and properties of Petri nets and algebraic models and exercises where the students will create concrete examples. There will be also programming exercises where the students can use the learned methods.			
Literature: Murata, T.: Petri Nets, Properties, Analysis and Applications (Proc. of the IEEE. Vol. 77., no. 4, ASpr 1989, 541-580) Best, E., Devillers, R., Koutny, M.: Petri Net Algebra (Springer 2001) Hennessy M.: Algebraic Theory of Processes (MIT, 1989) Hoare, C.A.R.: Communicating Sequential Processes (Prentice-Hall, 1985)			

Name of the course: Design of Distributed Systems			
Faculty member responsible for the course: Dr. Zoltán Horváth, professor			
Responsible department: Faculty of Informatics, Department of Programming Languages And Compilers			
Total credits: 5			
Total hours: 5			
Type of the course	**lecture**	**practice**	**consultation**

Hours per week	2	2	1
Type of testing	exam	practice	

Topics:
Students will be able to express and verify the properties of the distributed programs using formal methods, apply different ways to create advanced compositions of simple programs, and solutions for interesting and difficult problems in a distributed way.
Dining/drinking philosophers, formal specification of distributed problems, properties of distributed systems, safety and progress properties of distributed programs, verification of safety critical properties, program compositions from components with proven properties, computing the value of an associative function, message channels, pipelined networks programming exercises where the students apply the learned methods in the practice.

Literature:
Misra, J.: A discipline of multiprogramming: programming theory for distributed applications (Springer, 2001)
K. Mani Chandy and Jayadev Misra: Parallel Program Design: A Foundation (Addison-Wesley, Reading, MA, Reading, Mass., 1988)
Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers (Addison-Wesley 2002)
Schmidt, D., C. et al.: Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects (Wiley & Sons, 2000)

Name of the course: Advanced cryptography

Faculty member responsible for the course: Dr. Péter Sziklai, associate professor

Responsible department: Faculty of Science, Institute of Mathematics, Department of Computer Science

Total credits: 6

Total hours: 6

Type of the course	lecture	practice	consultation
Hours per week	2	2	2
Type of testing	exam	practice	

Topics:
The course have two main goals: discovering the mathematical background beyond several cryptographic constructions and introducing novel cryptographic primitives using interesting results from various topics of mathematics or computer science. For the first part, we present the necessary exact definitions, precise assumptions and rigorous proofs of security. For the second part, we present recent results, methods and its connections to cryptographic problems from finite fields to linear algebra.

Perfect and computational security, proofs by reduction, security definitions, pseudorandomness, message authentication codes, collision-resistant hash functions, one-way functions, cryptographic hardness assumptions, primality testing, factoring and computing discrete logarithms, arithmetics in finite fields and its applications, elliptic curve based cryptography, lattice based constructions, secure multiparty computation, secret sharing problems, applications for e-commerce.

Literature:
Berlekamp, E.R.: Algebraic Coding Theory. McGraw Hill, 1968
Huffman, W.C. –Pless, V.: Fundamentals of Error-Correcting Codes. Cambridge University Press, 2003

van Lint, J.H.: Introduction to coding theory. Springer Verlag, 1982
McWilliams, F.J. – Sloane, M.J.A.: The theory of error-correcting codes. North-Holland, 1977
Roman, S.: Coding and information theory. Springer Verlag, 1992
Beutelspacher, A.: Cryptology. The Mathematical Association of America, 1994
Brassard, G.: Modern cryptology. Springer Verlag, 1988
van Tilborg: An introduction to cryptology. Kluiver Academic Publisher, 1988

Name of the course: Applied cryptography project seminar			
Faculty member responsible for the course: Dr. Péter Sziklai, associate professor			
Responsible department: Faculty of Science, Institute of Mathematics, Department of Computer Science			
Total credits: 6			
Total hours: 6			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	2
Type of testing	exam	practice	

Topics:
The objective of the course is to develop and strengthen the ability to complete miniprojects, working in small groups (3 persons approx.). The practical aspects of the learned cryptographical solutions is emphasized, as well as focused team work concentrated on modeling and solving a security problem originated in a real, practical situation.

Literature:
Berlekamp, E.R.: Algebraic Coding Theory. McGraw Hill, 1968
Huffman, W.C. –Pless, V.: Fundamentals of Error-Correcting Codes. Cambridge University Press, 2003
van Lint, J.H.: Introduction to coding theory. Springer Verlag, 1982
McWilliams, F.J. – Sloane, M.J.A.: The theory of error-correcting codes. North-Holland, 1977
Roman, S.: Coding and information theory. Springer Verlag, 1992
Beutelspacher, A.: Cryptology. The Mathematical Association of America, 1994
Brassard, G.: Modern cryptology. Springer Verlag, 1988
van Tilborg: An introduction to cryptology. Kluiver Academic Publisher, 1988

Name of the course: Cryptographic protocols			
Faculty member responsible for the course: Dr. Attila Kovács, associate professor			
Responsible department: Faculty of Informatics, Department of Computer Algebra			
Total credits: 4			
Total hours: 4			
Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	
Type of testing	exam	practice	

Topics:

This course gives an overview of the basic building blocks used to engineer cryptographic protocols, and discusses in details the operation of mainstream cryptographic protocols used in wired and wireless computer networks. In particular, TLS and IPsec are covered, as well as security protocols in WiFi networks. We also study protocols used in emerging wireless networks, such as wireless sensor networks and RFID systems.

Basic concepts and crypto primitives
Basic concepts and crypto primitives (cont'd)
Block encryption modes
Message authentication and authenticated encryption
Key exchange protocols
Random number generation
Verification of key exchange protocols with ProVerif
Public Key Infrastructres
TLS
WiFi security
IPsec
Security protocols for wireless sensor networks
Secure routing and wormhole detection
RFID security and privacy

Literature:

G. Schaefer, Security in Fixed and Wireless Networks, Wiley, 2004.
J. Edney and W. A. Arbaugh, Real 802.11 Security: WiFi Protected Access and 802.11i, Addison-Wesley, 2003.
L. Buttyán, JP. Hubaux, Security and Cooperation in Wireless Networks, Cambridge University Press, 2008.
J. Lopez and Z-H. Zhou (eds), Wireless Sensor Network Security, IOS Press, 2008.
A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

Name of the course: Cryptography and its applications
Faculty member responsible for the course: Dr. Attila Kovács, associate professor
Responsible department: Faculty of Informatics, Department of Computer Algebra
Total credits: 4
Total hours: 4

Type of the course	**lecture**	**practice**	**consultation**
Hours per week	2	2	
Type of testing	exam	practice	

Topics:

Paradigms of provable security algorithmic reduction, algorithmic indistinguishability, simulatability.
Universal composability (UC) security framework:
The underlying computational model: PPT interactive Turing machines.
The model of protocol execution.
Protocol emulation. Special approaches: black box adversary, dummy adversary.
Ideal functionality and ideal protocol. Defining the security protocols: emulation of the ideal functionality.
Hybrid protocol.
The UC theorem.
UC with joint state (JUC).
Definition of ideal functionalities for cryptographic tasks.

Examples: UC-realization, composition, technique of hybrid protocols.
Literature: [1] R.Canetti: Universally Composable Security: A New Paradigm for Cryptographic Protocols, IACR Archieve, 2005 [2] M.Backes, B. Pfitzmann, M.Waidner: A Universally Composable Cryptographic Library, IACR Archieve, 2003 [3] O.Goldreich: Foundations of Cryptography, Cambridge Press, 2004