NAME OF THE COURSE	Introduction to Software Engineering					
Code	PMID50	Year of study	UGU-3			
Course teacher	doc.dr. sc. Branko Žitko	Credits (ECTS)	5,0			
Associate teachers	Mensur Duraković	Type of instruction (number of hours)	L	S	E	F
			30		30	
Status of the course	obligatory	Percentage of application of e-learning				

COURSE DESCRIPTION	
Course objectives	Categorize and compare the life cycle of software development. Identify and describe the elements of the life cycle of software development. Make process models and other models that occur during the life cycle of software development. Describe the stages of specific activities in life cycle of software development. Measure the process of software and software development. Model, implement and test the object-oriented software.
Course enrolment requirements and entry competences required for the course	Requirements: Object Oriented Programming Competences: Procedural programming in Python
Learning outcomes expected at the level of the course (4 to 10 learning outcomes)	describe the process of software development measure software make UML model of object-oriented software write an object-oriented programs in Python test software
Course content broken down in detail by weekly class schedule (syllabus)	Week 1: Lectures: Introductory lecture: teachers, student obligations, elements of the monitoring, examination, evaluation, presentation of the course objectives, literature The life cycle of software: activities and documents in life cycle, life cycle models, linear model, prototype model, spiral model, incremental model, an iterative and incremental model, unified process Exercise: Defining the class and create an object in Python, attributes and methods, special methods Week2: Lectures: Process model of software development, data flow diagrams, Petri nets, software models, the object model, sequence diagram, use-case model, scripting, the control flow graph, state diagram Exercises: The static attributes and static methods, special methods Week3: Lectures: Project management software, process and project management, team management approach, the model of maturity, personal process, analysis of the acquired values, bug tracking, post mortem analysis Exercises: collection classes, special methods of collections class Week4: Lectures: Project planning, task structure breakdown, evaluation techniques and reviews of software, cost estimates, LOC estimation, COCOMO model, assessment of function points Exercises: Inheritance and polymorphism, superclass and subclasses, calling superclass method

	Week5: Lectures: Measuring software, measurement theory, relational systems of measurement, flatness, measuring scales, metrics in software, cyclic numbers, Halstead measure, Henry Kafuarov flow of information, process metrics and productivity Exercises: modules and applications with multiple files, packages Week6: Lectures: Colloquium Exercises: use-case model, action scripting, activity diagram Week7: Lectures: management and risk analysis, risk identification, risk assessment, risk exposure, decision tree risk, risk reduction, risk management plan, quality assurance software, formal inspections and checks, the reliability of software, statistics assurance Exercises: UML use-case diagram, scenario, activity diagram Week8: Lectures: Requirements, object model of requirements, data flow modeling, requirements dictionary, system diagram Exercises: UML class diagram, architecture modeling, UML modeling of attributes and methods, implementation of the attributes and methods in Python Week9: Lectures: Design, phases of design process, abstraction methods, measuring cohesion, measuring connections, monitoring requirements Exercises: UML modeling of relationships and inheritance, implementation of relations and inheritance in Python Week10: Lectures: Basics of software testing, test coverage criteria, inclusion, functional testing, test matrix, structural testing, test data flow, random testing, boundary testing Exercises: modeling user interface, the implementation of the user interface Week11: Lectures: Colloquium Exercises: modeling of the control interface, implementation of the control interface in Python Week12: Lectures: Object-oriented software development, identification of objects, identifying associations, identification of a multiplicity of associations Exercises: UML sequence diagram, mapping diagram of activities in sequence diagram Week13: Lectures: Traditional object-oriented metrics, metrics of object-oriented design, MOOD metrics Exercises: testing methods in Python Week14: Lectures: Object-oriented testing, MM testing, coverage of function pairs Exercises: Testing class in Python Week15: Lectures: Colloquium Labs: Testing module in Python

<table>
<tr><td>Format of instruction</td><td>☒ lectures
☐ seminars and workshops
☒ exercises
☐ on line in entirety
☐ partial e-learning
☐ field work</td><td>☐ independent assignments
☐ multimedia
☐ laboratory
☐ work with mentor
☐ homework assignments</td></tr>
<tr><td>Student responsibilities</td><td colspan="2">class attendance
active participation in the learning process
colloquiums
oral exam</td></tr>
<tr><td>Screening student work (name the proportion of ECTS credits for each activity so that the total number of ECTS credits is equal to the ECTS value of the course)</td><td colspan="2">
<table>
<tr><th>Name</th><th>Ects</th><th>Name</th><th>Ects</th><th>Name</th><th>Ects</th></tr>
<tr><td>Class attendance</td><td>2</td><td>Research</td><td></td><td>Experimental work</td><td></td></tr>
<tr><td>Oral exam</td><td>1</td><td>Report</td><td></td><td>Homework assignments</td><td></td></tr>
<tr><td>Seminar essay</td><td></td><td>Essay</td><td></td><td></td><td></td></tr>
<tr><td>Tests</td><td>1</td><td>Practical training</td><td>1</td><td></td><td></td></tr>
<tr><td>Written exam</td><td></td><td>Project</td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Grading and evaluating student work in class and at the final exam</td><td colspan="2">Activity of students in lectures and exercises (attendance, problem solving, general activity in the classroom) (25%).
If student has more than 50% in each colloquium than frees the oral exam.
Colloquium (50%)
Oral exam (25%)
The final grade is derived on the basis of all the above ratings.</td></tr>
<tr><td>Required literature (available in the library and via other media)</td><td colspan="2">
<table>
<tr><th>Title</th><th>Number of copies in the library</th><th>Availability via other media</th></tr>
<tr><td>Schaum's Outlines of Software Engineering, David A. Gustafson, McGraw-Hill, 2002, online</td><td>0</td><td>Web</td></tr>
</table>
</td></tr>
<tr><td>Optional literature (at the time of submission of study programme proposal)</td><td colspan="2">Software Engineering, Ian Sommerville, Addison-Wesley, 2011</td></tr>
<tr><td>Quality assurance methods that ensure the acquisition of exit competences</td><td colspan="2">talk with students
student evaluation using the anonymous survey
the success of students in the exam
self-assessment.</td></tr>
<tr><td>Other (as the proposer wishes to add)</td><td colspan="2"></td></tr>
</table>