|  |
| --- |
| **WSB University Branch/Department of Jaworzno** |
| **Field of study: Computer Science** |
| **Subject: Fundamentals of object-oriented programming** |
| **Educational profile: practical** |
| **Level of education: undergraduate studies** |
| **Number of hours** **per semester** | 1 | 2 | 3 | 4 |
| I | II | III | IV | V | VI | VII |
| **Full-time studies**(w/w/lab/pr/e)\* |  | **16w / 20lab / 25Pr** |  |  |  |  |  |
| **Part-time studies**(w/æw/lab/pr/e) |  | **12w / 16lab / 25Pr** |  |  |  |  |  |
| **LANGUAGE OF INSTRUCTION** | Polish |
| **LECTURER** | Dr.-Ing. Dawid Aleksander, Prof. AWSB, M.Sc. Popławski Krzysztof |
| **FORM OF ACTIVITIES** | Lecture, laboratory, project, consultation |
| **SUBJECT OBJECTIVES** | The aim of the course is to learn object-oriented thinking, analysis, design and programming. The aim of the course is to provide knowledge of object-oriented programming using the Java platform. It includes the presentation of basic concepts and issues present in object-oriented programming. The lecture introduces the concepts of class and object and issues related to encapsulation, inheritance and polymorphism. The basics of object-oriented design are discussed. Formalisms for writing object-oriented designs and programs are introduced. The lab focuses on writing small programs to illustrate further issues presented in the lecture and teaches you to describe your solutions in the accepted design notation. |
| **Reference to learning outcomes** | **Description of learning outcomes** | **Means of verification of the effect****learning** |
| **Directional effect** | **PRK** |
| **NEWS** |
| INF\_W02 | P6S\_WG | The student knows and understands the concepts and principles of the object-oriented approach to software development | Written examination |
| INF\_W02 | P6S\_WG | The student knows and understands the object-oriented elements and mechanisms used in programming languages. | Written examination |
| INF\_W05 | P6S\_WG | The student knows and understands the techniques and tools involved in analysing and designing software in an object-oriented approach  | Written examination |

|  |  |  |  |
| --- | --- | --- | --- |
| INF\_W02 | P6S\_WG | The student knows the basic syntax of at least one of the object-oriented programming languages (C++, Java, C#). | Individual consultations of the instructor with students performing specific tasks. Conducting a colloquium on the syntax and semantics of one of the object-oriented programming languages. Checking the degree of independent completion of the credit project - questions concerning the source code of the software. |
| INF\_W02 | P6S\_WG | Has knowledge of the mechanisms that allow object-oriented programming using the chosen programming language. | Individual consultations of the instructor with students performing specific tasks. Conducting a colloquium on the syntax and semantics of one of the object-oriented programming languages. Checking the degree of independent completion of the credit project - questions concerning the source code of the software. |
| **SKILLS** |
| INF\_U06 | P6S\_UW  | The student is able to write a program using the object-oriented approach  | Analytical tasks  |
| INF\_U10 | P6U\_UW | Students will be able to use appropriately selected programming environments in the process of designing and verifying the operation of computer applications. | Analytical tasks |
| INF\_U10INF\_U14 | P6S\_UW  | The student is able to design and implement an information system using object-oriented techniques | Analytical tasks  |
| INF\_U13 | P6S\_UW  | Students will be able to critically analyse how software (including software consisting of multiple components and processes, also distributed) works, and be able to identify and formulatespecification of simple computer programming tasks. | Project tasks |
| INF\_U03 | P6S\_UW | The student makes skilful use ofCan make use of available sources on programming libraries and language constructs in developing object-oriented programmes. Can make use of ready-made basic methods and classes and use them in the programmes he/she writes | At each stage of the problem-solving process, the tutor checks the concrete results and directs further work. |

|  |  |  |  |
| --- | --- | --- | --- |
| INF\_U02INF\_U10 | P6S\_UW | Creates simple programs in the selected language (C++, C#, Java) independently using the basics of object-oriented methodology. Can also design, implement and describe created software in a team of 2-3 people. | At each stage of the problem-solving process, the tutor checks the concrete results and directs further work. |
| INF\_U10 | P6S\_UW | The student is able to install a basic working environment depending on the chosen object-oriented language (C++, C#, Java). He/she also has the ability to comment the code using the chosen IT tool of the project he/she creates. Compiles and runs programmes. | Checking of the various stages of the student's work and acceptance of the final result - the final credit project. |
| **SOCIAL COMPETENCES** |
| INF\_K01 | P6U\_K | The student is prepared to critically evaluate his/her knowledge and perceived content concerning the achievements of computer science, to recognise the importance of knowledge in solving cognitive and practical problems, and to seek expert advice when encountering difficulties in solving a problem independently. | Project assignments, classroom observation |
| **Student workload (in teaching hours 1h =45 minutes)\*\***  |
| **Stationary**attendance at lectures = 16participation in exercises = 20preparation for exercise = 29lecture preparation = 15exam preparation = 10implementation of project tasks = 25e-learning =Pass/examination = 4other (consultation) = 6**TOTAL: 125h****Number of ECTS credits: 5****including in practical classes: 3** | **Part-time**attendance at lectures = 12participation in exercises = 16preparation for exercise = 30lecture preparation = 22exam preparation = 10implementation of project tasks = 25e-learning =Pass/examination = 4other (consultation) = 6**TOTAL: 125h****Number of ECTS credits: 5****including in practical classes: 3** |
| **PREREQUISITES** | Fundamentals of computer programming. Ability to think abstractly. Knowledge of the basics of computer science and computer use. Basic algorithmics (data structures and simple algorithms) and syntax basics of a structured (e.g. C, Pascal) or object-oriented language (e.g. C++, C#, Java). |
| **SUBJECT CONTENT**(broken down into face-to-face and e-learning classes) | Content delivered in a face-to-face format: Lecture: * Modelling using UML.
* Debugging - eliminating errors in an object-oriented program.
* Basic elements of the Java language
* Operators and expressions
* Methods and encapsulation, polymorphism
* Classes and objects, encapsulation, inheritance.
* Input/output operations, data formatting.
* Exceptions and error handling.
* Multi-threaded programmes.
* Design patterns and their implementation in Java.
* Key concepts and principles of the object-oriented approach in different programming languages: class, object, constructor, destructor; creation, deletion, initialisation of objects; operator overloading; inheritance; virtual functions.
* Linking, aggregation and composition of objects.
* Class composition versus role delegation. Implementation issues and methods. Use of interfaces, abstract classes.
* Abstraction, static and dynamic polymorphism, data encapsulation, types of...
* Analysis and object-oriented design

Lab:* Introduction to object-oriented programming
* The concept of class, encapsulation
* Fundamentals of object-oriented programming (C++, C#, Java)
* Dynamic memory management in C++ - the new and delete operators
* Access: public, protected and private to fields and methods
* the keyword "this" - its meaning and use
* Static methods and fields
* Fixed values, fixed objects and fixed object methods
* 9 Creation and destruction of objects - constructors and destructors (for C++ language)
* Object storage methods: arrays, template classes (C++), collections (Java)
* Selected elements of libraries
* Overloading and obfuscation of methods
* Inheritance and polymorphism
* 14. interfaces (Java language)
 |
| **LITERATURE** **COMPULSORY** | * E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns. Elements of reusable object-oriented software , Helion , 2021.
* J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley, The Java Language Specification, Java SE 8 Edition, Addison-Wesley Professional, 2014.
* C++ Language. A compendium of knowledge, Stroustrup B., Helion 2014
* Grębosz J.: Symfonia C++: Programowanie w języku C++ zorientowane obiektowo, vol. 1, "Edition 2000" Oficyna Kallimach Publishing House, Kraków 2010
* Grębosz J.: Symfonia C++: Programowanie w języku C++ zorientowane obiektowo, vol. 2, "Edition 2000" Oficyna Kallimach Publishing House, Kraków 2010
* Thinking in Java, Eckel B., Helion 2006
* Thinking in C++, vol.1, Eckel B., Helion 2004
 |
| **LITERATURE** **SUPPLEMENTARY**(including min. 2 items in English; book publications or articles) | * <https://docs.oracle.com/en/java/>
* <https://docs.oracle.com/javase/tutorial/>
* G. Booch, J. Rumbaugh, I. Jacobson, UML user's guide, Wydawnictwa Naukowo Techniczne, 2002.
* Craig Larman, UML and design patterns. Object-oriented analysis and design and the iterative model of application development. Wydanie III, Helion 2011
* Big Java: Early Objects, C. S. Horstman, 7th Edition, Wiley 2019
* Effective Java, J. Bloch, 3rd Edition, Addison-Wesley Professional 2017
* Josuttis N. M.: C++. The standard library, Helion 2014
* Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14, S. Meyers, O'Reilly Media 2014
* Prata S.: C++ language. Szkoła programowania, Helion 2012
* Bertrand Meyer, Object-Oriented Programming, Helion 2005
* Matt Weisfeld, Object-oriented thinking in programming. 4th ed. Helion 2014.
* Java. A programmer's compendium., H. Schildt, 11th ed., Helion 2020
* Java. A guide for beginners, H. Schildt, 11th ed., Helion 2015
* Java Fundamentals, Cay S. Horstmann, Gary Cornell, 9th ed., Helion 2014
 |
| **TEACHING METHODS**(broken down into face-to-face and e-learning classes) | In direct form:* Multimedia lecture with presentation of many examples in different programming languages.
* Discussion of the topic of the laboratory activity using a blackboard and multimedia projector
* Presentation of the programming technique for the problem discussed at the beginning of the class using a multimedia projector
* Supervise students' independent work and give them guidance in developing the program code created and discussed during the first part of the meeting
* Project tasks, practical examples, case studies
 |
| **LEARNING AIDS** | Onlinewsb platform, MS Teams application, multimedia projector, auxiliary Jetbrains |
| **PROJECT**(insofar as it is carried out as part of a course module) | Project objective: To learn fundamental ways of modelling real-world problems and implementing them using object-oriented programming techniques.Project theme: Application built on the basis of classes and the mechanism of inheritance and polymorphism; e.g. a system for handling staff, a document database, etc. Project form: Interactive computer program with its source code and basic documentation in the form of a class diagram. |
| **FORM AND CONDITIONS OF PASSING**(broken down into face-to-face and e-learning classes) | A prerequisite for obtaining credit is obtaining a positive grade in all forms of assessment provided for in the course syllabus, taking into account the quantitative assessment criteria defined in the Framework System of Student Assessment at the WSB Academy.* participation in laboratory activities,
* Colloquium in the form of a test,
* creation, presentation and defence of own project prepared in groups.

Independent performance on a computer of a task requiring knowledge of the object-oriented approach to software development in any object-oriented language. |

*\* W - lecture, ćw - exercise, lab - laboratory, pro - project, e-learning*