Object oriented programming - course description

General information

Course name	Object oriented programming
Course ID	13.2-WF-FizP-OP-S17
Faculty	Faculty of Physics and Astronomy
Field of study	Physics
Education profile	academic
Level of studies	First-cycle Erasmus programme
Beginning semester	winter term 2017/2018

Course information

Semester	3
ECTS credits to win	6
Course type	obligatory
Teaching language	english
Author of syllabus	• dr Marcin Kośmider

Classes forms

The class form	Hours per semester (full-time)	Hours per week (full-time)	Hours per semester (part-time)	Hours per week (part-time)	Form of assignment
Lecture	30	2	-	-	Exam
Laboratory	30	2	-	-	Credit with grade

Aim of the course

The aim of this course is to introduce the Object Oriented Programming techniques required to develope and create modern applications related to the „every day" and science problems. This is an active course where students solve realistic problems from beggining. Students learn how to analyse problem in the object oriented way and how to implement code according to the standards.

Prerequisites

The efficient use of the Linux system (both in the terminal and in the graphical environment), knowledge of the basics of programming including procedural programming.

Scope

1. Introduction

- object and procedural programming

- class, object and methods

- constructor and destructor

- encapsulation

- pointers

- operators overloading

- friend function

2. Using standard class

- IO operations

- short introduction to the STL containers and algorithms

3. Pointers

- objects and dynamic memory allocation

- copy constructor

- destructor

- „intelligent" pointers

4. Inheritance, polymorphism and code reuse

- inheritance

- virtual and abstract classes and methods

- interfaces

- polymorphism

- the idea of „code reuse”

5. Clean code

- name standards

- header files

- namespaces

- makefile

- code comment and documentation

- version control systems

6. Templates in C++

7. Exception

8. Object oriented modelling and programming

- defining and analysing problem and model creation

- UML diagrams

- coding UML diagrams in C++

9. Design patterns

- the idea

- creational patterns

- structural patterns

- behaviour patterns

10. Frameworks

- the idea

- Qt as as sample

Teaching methods

Lecture:

Convencional lecture, work with problems, discusiion, workshop

Laboratory:

Laboratory exercise, project, work in group, presentation, work with documentation, independed work, brain storm

Learning outcomes and methods of theirs verification

Outcome description	Outcome symbols	Methods of verification	The class form
Student can compile and run program. For a given physical problem student can analyse and interpret computational resuts and verify the correctness of a written application		• a discussion • a project • an exam - oral, descriptive, test and other • an ongoing monitoring during classes	• Lecture • Laboratory
Student knows how to search, find and use modern tools and informations that can be used to solve given problem		• a discussion • a project • an exam - oral, descriptive, test and other • an ongoing monitoring during classes	• Lecture • Laboratory

Outcome description	Outcome symbols	Methods of verification	The class form
Student can define problem and explain the problem posed by breaking it into elementary problems, describes and analyses it in the object oriented way, indicating the models, objects and relations between them.		• a discussion • a project • an exam - oral, descriptive, test and other • an ongoing monitoring during classes	• Lecture • Laboratory
The student can apply his knowledge of programming and object modeling and available tools to present ways to solve the considered problem in physics or related fields in the form of the source code of the program.		• a discussion • a project • an exam - oral, descriptive, test and other • an ongoing monitoring during classes	• Lecture • Laboratory
Student can create and present report from given problem		• a discussion • a project • an exam - oral, descriptive, test and other • an ongoing monitoring during classes	• Laboratory
Student know laboratory statute and BHP rules		• a discussion • a project • an ongoing monitoring during classes	• Laboratory
The student is able to cooperate in a group, feels responsible for the tasks entrusted to him, is open to new concepts and ideas.		• a discussion • a project • an ongoing monitoring during classes	• Lecture • Laboratory

Assignment conditions

Lecture:
A practical exam consisting in solving a given problem (chosen from the list of problems). Final evaluation is subject to problem analysis, presentation of problem solving algorithms, source code as well as evaluation and verification of obtained results

Laboratory:
The final grade consists of: average marks obtained during laboratories with activity and short tests to check learning progress (50% of final grade), semester project assessment (50% of final grade). The condition for passing the semester project is its implementation, preparation and delivery of the project report and its presentation within the prescribed period. Before taking the exam the student must get a pass from the exercises.

Final grade: weighted average of exam grades (60%) and exercises (40%).

Recommended reading

[1] Bruce Eckel, Thinking in C++ Edycja Polska, Helion Gliwice, 2002.
[2] Bruce Eckel, Thinking in C++ Edycja Polska, Tom 2, Helion Gliwice, 2004.
[3] Steve Holzner, Design patterns for dummies, Willey Publishing Ing. Indianapolis 2006.

Further reading

[1] Internet

Notes

The lecture should take place in a room with Internet access. Computer laboratories should take place in groups enabling independent work at the computer of every student and not more than 12 people.

Modified by dr hab. Maria Przybylska, prof. UZ (last modification: 06-07-2018 21:52)