Object-oriented programming

Rafał Witkowski

2021

Mikołaj K

Poza tym nie wiem czy słyszeliście państwo o takiej technice jak programowanie obiektowe - chodzi z grubsza o to, ze obiekty którymi otacza się ludzi mają wpływ na nasze mózgi tak żebyśmy byli zniewoleni.

Programowanie obiektowe nawet wykorzystuje swoją specjalną technikę która nazywa się Jawa - przez to ludzie myślą ze symulacja na która patrzą to tak naprawdę jawa czyli jest prawdziwa

Przed chwilą Lubię to! Więcej

Introduction

- Goal for the lecture:

  - Introduction to object-oriented programming,
  - Introduction to the use of patterns.
  - Introduction to UML (Unified Modelling Language),
  - Skills acquired during the classes will allow to create simple object-oriented software

- Prerequisites for a lecture listener:

  - Knowledge of some object-oriented programming languages (C++, Java, C#) and experience in programming in one language,

  with procedures and function

Why object-oriented programming is not easy?

- Knowledge of syntax is not sufficient to effectively program using object-oriented languages (C++, Java, C#) - "Having a hammer does not make us immediately architect”.
- Programming in these languages becomes effective in case of correct "object-oriented thinking". Therefore it is necessary before starting the programming, carrying out an analysis (object-oriented analysis and design: OOA/D).
- Critical and basic skills of an OOA/D is the ability to correspond to individual responsability, deciding how the responsability should interact with each other, defining what should do what.

Why object-oriented programming?

- Real World Modelling:

Object-oriented programming means building in the program models of objects from the real world. The programmer's work consists of the revitalisation of these objects and making them to start cooperate with each other.

Why object-oriented programming?

- Code reusablity:

Once written and saved, the code can be used several times. The following is implemented it is thanks to inheritance (methods from the superclass are available in classes derivatives).

Why object-oriented programming?

- Extensibility:

Thanks to polymorphism, implementation of new classes does not require to modify parts of the code already done. Polymorphic methods are responsible for carrying out activities related to a given class. If new classes are introduced, appropriate polymorphic methods will be introduced.

Why object-oriented programming?

- Improved error detection:

Thanks to the connection of methods to objects, it is not possible to make errors of incorrect function calling with unauthorized arguments (methods operate on data related to the object). Thus, the number of errors of incorrect use of the function is minimized.

Why object-oriented programming?

- Simplicity in programming the user interfaces:

Object-oriented programming directly responds to the needs of creating "window" user interests. Windows can be seen as objects, consisting of other objects (buttons, menu items, etc.).

Object-oriented analysis

- The goal of object-oriented analysis is to provide answers to the question: how should the system work?
- Tasks carried out during the analysis:
  - creation of a logic system model describing the way of implementation by the system of set requirements (the logical model omits most of the implementation details),
  - to define a basic "dictionary" of domain knowledge in order to facilitate the analysis and subsequent development of applications.
- This is achieved by finding and describing the objects, understood as concepts
- Created models are saved with the use of notations defined in modelling languages (e.g. UML).

Object-oriented designing

- The aim of object-oriented designing is to answer the question: how to implement the system?
- Tasks carried out during the design process:
  - development of a detailed description,
  - definition of objects used in the program.
- This is achieved by defining the principles of objects cooperation in such a way that it meets the requirements (defining attributes, methods, etc.).
- The structure of the created software should as far as possible preserve the general structure of the model created in the previous phase.
- In the design phase the same notation is used as in the analysis phase.

Object-oriented programming

- Implementation (encoding) of the software project in the selected environment:
  - use of the object-oriented language,
  - reuse of already existing object libraries,
  - use of tools for quick application development,
  - use of code generators.
- Take steps to develop reliable software:
  - avoidance of dangerous techniques,
  - the need-to-know principle (you know only you need to know),
  - the use of type-checking compilers.

UML – object-oriented models

- What is a model?
  - A model is a simplification of reality.
- Why do we model?
  - We build models to better understand the system we create.

UML – object-oriented models

- In what modelling helps us in:
  - in the visualization of the system as it is, or as it should be,
  - in the specification of the structure or behaviour of the system,
  - serves as a template during application development,
  - documents the results of the work carried out.
- Why do we need formal models?
  - A unified language will facilitate communication and allow to describe the created model in a uniform way.

UML – what is it?

- Unified Modelling Language
- A unified language for the object-oriented modelling.
- UML provides the system designer a tool for visualizing, specifying, constructing and documenting concepts and mechanisms that make up the solution of the problem.

UML characteristic

- UML is developed by the following persons: Grady Booch, James Rumbaugh and Ivar Jacobson,
- successor of other object-oriented methodologies such as OMT (Rumbaugh), OOA/OOD (Coad, Yourdon), OOAD (Booch), Objectory (Jacobson),
- a combination of theory and practice: Rational Software Corporation.

Patterns

- „Design templates” are a written collection of advanced experience of object-oriented program designers.
- Design patterns are written in a codified way that allows to describe a programming problem and its solution.
- The most famous design patterns (23 patterns) were developed by the so-called Gang of Four (Gamma, Helm, Johnson and Vlissides):
  - Adapter,
  - Factory,
  - Singleton,
  - Strategy,
  - – ...

Bibliography

1. Craig Larman, Applying UML and Patterns An Introduction to Object-Oriented Analysis and Design and the Unified Process, Prentice Hall, 2002
2. Rebecca Wirfs-Brock, Alan McKean, Object Design – Roles, Responsibilities and Collaborations, Addison Wesley, 2003
3. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns Elements of Reusable Object-Oriented Software, Addison Wesley, 1995
4. Richard C. Lee, William M. Tepfenhart, UML and C++ A practical guide to object-oriented development, Prentice Hall, 1997
5. Bruce Eckel, Thinking in Java