Object Oriented Programming

Introduction (organization, topics, tasks)

2023/24

What is new?

```
class KeyValue
{
    private:
        int key;
        double value;

    public:
        KeyValue(int k, double v);
        int GetKey();
        double GetValue();
};
```

Why do we need OOP?

- OOP is more about software design than programming.
- We will focus on OOP techniques whose use is projected into well-organized code with measurable quality.
- This course is not about programming in C ++.
- We will use only minimum syntax and C ++ language constructions.

Lecture Outline

- Basic information
- About the subject
- Topics
- Example

Basic Information

Contact

- doc. Mgr. Miloš Kudělka, Ph.D.
- milos.kudelka@vsb.cz
- http://homel.vsb.cz/~kud007
- EA 439, +420 597 325 877

General requirements

- Lecture attendance is recommended.
- Seminars are mandatory.
- Homework can be required as a part of seminars.
- The required output skill is the ability to write a small program with object-oriented design.

Credit requirements

- Active participation in seminars (9 seminars with maximum 4 points + project with maximum 9 points; in total it is necessary to obtain at least 23 points out of 45 possible).

- Written test with lecture questions and simple program codes in C++ (at least 28 points out of 55 possible).

- For one seminar, a maximum of 4 points can be obtained in answers to lecture questions and for activity due solving examples (a maximum of 4 points per seminar).

- For the semester, one project will be evaluated (maximum 9 points).

About the Subject

Objectives

- What?
  - What do we understand by object-oriented principles and techniques? What they are and what do they mean?
- Why?
  - Why do we need object-oriented programming? What is the motivation? What is the purpose of object-oriented principles and what are different techniques?
- When?
  - When should we use different techniques and when not?
- How?
  - How to correctly understand object-oriented principles? How to properly use different techniques?

Sources

- Meyer, B. *Object-Oriented Software Construction*. Prentice Hall, 1997.

- Eckel B. *Thinking in C++*. Prentice Hall, 2000.

- Stroustrup, B. *The C++ Programming Language*. Addison-Wesley Professional 2013.

Topics

Programming Paradigms

- How and why programming languages evolve?
- How object-oriented programming (OOP) differs from other paradigms?
- What are the aspects of software quality?

- **A paradigm** is a distinct **set of concepts or thought patterns, including theories, research methods, postulates, and standards** for what constitutes legitimate contributions to a field. [Wikipedia]

Class and Objects

- What are classes and objects in OOP?
- The class is a static description.
- The object is a run-time representation that has:
  - state (data)
  - behavior (functions)

OOP Principles

- General principles
  - Information hiding
  - Composition
  - Message passing
  - General–special relationship
- Technical principles
  - Encapsulation
  - Polymorphism
  - Inheritance

Life-cycle of Objects

- How are objects created and destroyed?
- What are constructors and destructors?
- How does it work in different situations?

Information hiding

- What is a public and what is a private part of an object?
- Why is it important to hide information?
- What is a correct design of the public and the private parts of an object?

Inheritance

- Simple inheritance and the reasons for its use (polymorphism).
- Abstract class.
- Types of implementation hiding.
- Multiple and repeated inheritance. Issues.

And others...

- Design of object programs.
- Generic types.
- Exceptions.
- Object libraries.

Example

Class Declaration

```
#include <iostream>
using namespace std;

class KeyValue
{
    private:
        int key;
        double value;

    public:
        KeyValue(int k, double v);
        int GetKey();
        double GetValue();
};
```

Class Definition (implementation)

```
KeyValue::KeyValue(int k, double v)
{
    this->key = k;
    this->value = v;
}

int KeyValue::GetKey()
{
    return this->key;
}

double KeyValue::GetValue()
{
    return this->value;
}
```

Using the Class

```
int main()
{
    KeyValue kv1(1, 1.5);
    cout << kv1.GetValue() << endl;

    KeyValue *kv2 = new KeyValue(2, 2.5);
    cout << kv2->GetValue() << endl;
    delete kv2;

    getchar();
    return 0;
}
```

Seminar Assignments

- Design, declare and define simple classes and write the source code that works with objects of these classes.
- E-mail
- Person
- Stack