Programming II

Introduction (organization, topics, tasks)

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

Requirements

- Lecture attendance is recommended.
- Seminars are mandatory.
- Homework can be required as a part of seminars.

Requirements

- How to finish:
  - Active participation on seminars.
  - Answers for questions (20 - 40 points).
  - Written test (31 - 60 points).

About the Subject

Objectives

- What?
  - What do we understand object-oriented principles and techniques? What they are and what they mean?
- Why?
  - Why do we need object-oriented programming? What is a 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 relation
- 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. Problems.

More...

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

Example

Class Declaration

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

class KeyValue {
private:
    int key;
    float value;

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

Class Implementation (definition)

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

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

Using the Class

```
int main() {
    KeyValue c11(1, 1.5);
    cout << c11.GetValue() << endl;

    KeyValue *c12 = new KeyValue(2, 2.5);
    cout << c12->GetValue() << endl;
    delete c12;

    getchar();
    return 0;
}
```