Objektovo orientované programovanie

1. Prednáška

LS 2024/2025

Juraj Petrík

Čo tu budeme (snáď) robiť?

- OOP a Java
- Vývoj softvéru všeobecne a v kontexte OOP
- Testovanie
- Správa závislostí
- Ladenie
- Kolekcie
- Generickosť
- Vzory
- GUI
- Anotácie
- Dokumentácia
- Logovanie
- Paralelizmus
- IO
- Perzistencia
- Reflexia
- Metriky
- Iné OO jazyky
- Pozvané prednášky

Prečo ďalšia paradigma?

- Priblížiť ľudskému mysleniu v reálnom svete

- Napr. Java, Python, Scala, C++

- Pozor na paradigm shift (potrebný tréning, zvyk)

Procedural vs Object oriented

```c
#include <stdio.h>
#include <string.h>

struct Person {
    char name[50];
    int age;
};

void greet(struct Person p) {
    printf("Hello, %s! You are %d years old.\n",
        p.name, p.age);
}

int main() {
    struct Person person1;
    strcpy(person1.name, "Alice");
    person1.age = 25;

    greet(person1);
    return 0;
}
```

```java
class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void greet() {
        System.out.println("Hello, " + name + "! You
                are " + age + " years old.");
    }
}

public class Main {
    public static void main(String[] args) {
        Person person1 = new Person("Alice", 25);
        person1.greet();
    }
}
```

Ako [úspešne] absolvovať predmet?

- Semester (na pripustenie ku skúške 30b):
  - Test – 20b (8b minimum), ~8. týždeň semestra
  - Úlohy na cvičeniach (10b)
  - Semestrálny projekt (30b), info ~ 2. týždeň semestra:
    - Špecifikácia a architektúra (2b)
    - Checkpoint (5b)
    - Konečná implementácia (20b)
    - Prezentovanie (3b), ~10.-12. týždeň cvičení
  - Bonus 5b
- Skúška (40b, 10b minimum)
- Stupnica je štandardná (študijný poriadok STU)
- Akademická bezúhonnosť (študijný poriadok STU)

Rozvrh

Deň	8.00-8.50	9.00-9.50	10.00-10.50	11.00-11.50	12.00-12.50	13.00-13.50	14.00-14.50	15.00-15.50	16.00-16.50	17.00-17.50	18.00-18.50	19.00-19.50
Po		-2.01/c(CPUc) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. Považanová*		-2.01/c(CPUc) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. Považanová*								
Ut							-2.01/a(CPUa) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. Považanová*		-2.01/a(CPUa) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. Považanová*		-2.01/a(CPUa) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *P. Podhorský*	
							-2.01/b(CPUb) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *O. Udvardi*		-2.01/b(CPUb) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *O. Udvardi*			
St			-2.01/a(CPUa) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. A. Saleh*		-2.01/a(CPUa) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. A. Saleh*				-2.01/b(CPUb) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. L. Alqnatri*		-2.01/b(CPUb) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. L. Alqnatri*	
									-2.01/c(CPUc) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. Skyvová*		-2.01/c(CPUc) (BA-FIIT-FIIT) / * Objektovo-orientované programovanie *A. Skyvová*	
Št		-1.61 (Aula Magna) (BA-FIIT-FIIT) Objektovo-orientované programovanie *J. Petrík*										
Pi												

- Nahrádzanie cvičení
- https://fiit.jur1cek.eu/

Čo budeme potrebovať

- IDE (Intellij IDEA, Eclipse for JAVA, Netbeans)
  - Visual Studio Code
  - Notepad
- JDK 21 (LTS)
- Predinštalované v učebniach/laboch

Prečo objektovo orientované?

- Paralela s reálnym svetom
  - „Veci“ vnímame ako objekty
  - Ten istý typ objektu (trieda) sa môže nachádzať vo svete viac krát (objekt)
- Objekty
  - Majú nejaké vlastnosti (atribúty)
  - Vieme s nimi niečo robiť (metódy)

Java

- Veľa materiálov
- Ekosystém
- Silno typový jazyk
- “Write once, run everywhere” – Google “Saying that Java is nice because it works on all OS's is like saying”

- Nakoniec je to vlastne jedno ☺

Ako funguje Java?

- JVM vie optimalizovať kód počas behu
- Áno, Javoviny vedia alokovať veľa pamäte
- Rýchlosť (veľmi zovšeobecnené): C++ > Java > Python

Čo musí minimálne mať programovací jazyk aby bol použiteľný?

- „Pamäť“
- Vetvenie
- Cyklenie

Hello world

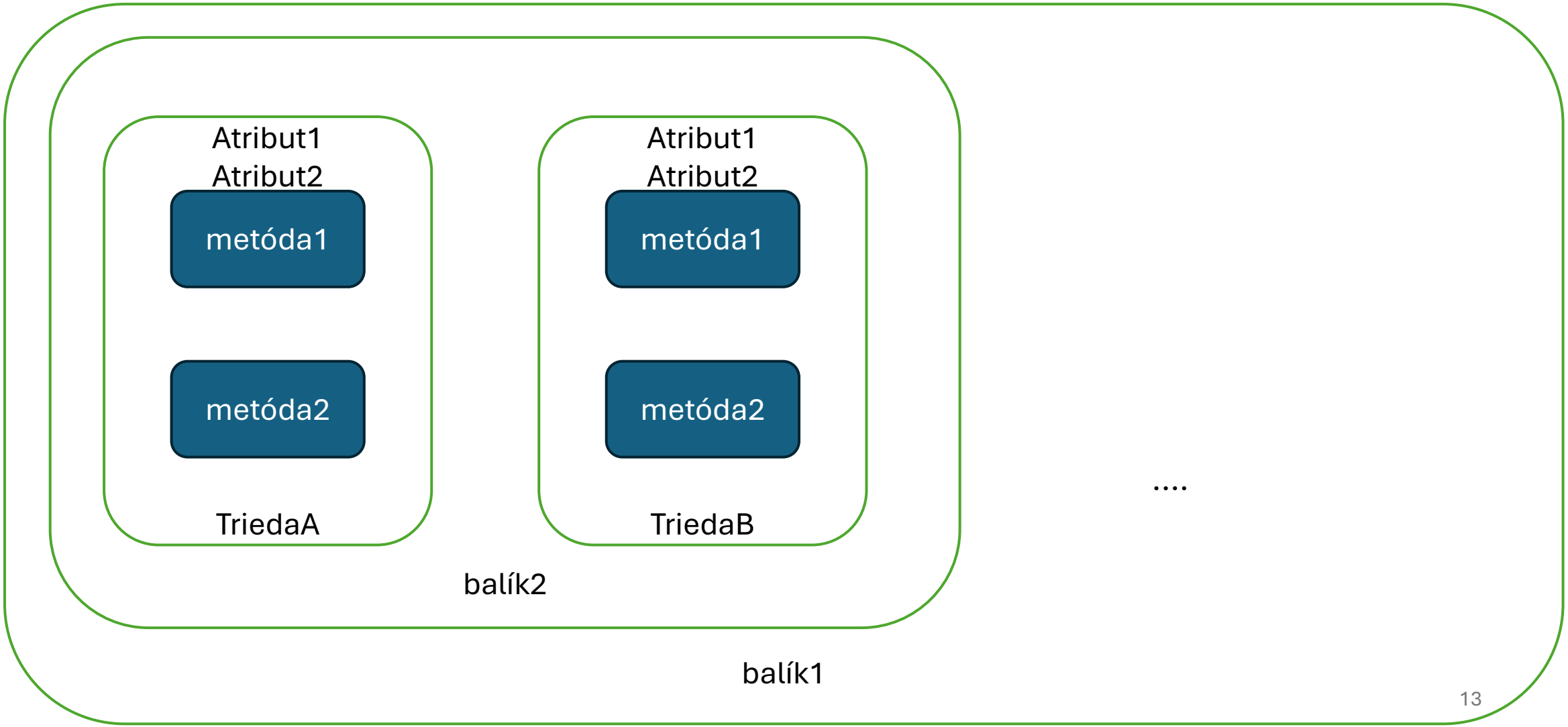
```
package sk.stuba.fiit;

public class Main {
    public static void main(String[] args) {
        System.out.printf("Hello and welcome!");

        for (int i = 1; i <= 5; i++) {
            System.out.println("i = " + i);
        }
    }
}
```

balík (sk -> stuba -> fiit)

trieda

metóda (main je špeciálna)

Metódy, triedy, balíčky



Naming conventions

- Balíky: *edu.cmu.cs.bovik.cheese*
- Triedy: *class ImageSprite;*
- Rozhrania: *interface RasterDelegate;*
- Metódy: *getBackground();*
- Premenné: *float myWidth;*
- Konštanty: *static final int MIN_WIDTH = 4;*

- https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html

Formátovanie kódu

- Áno
- Ako?
  - Konzistentne
    - V tíme
    - V čase

Premenné

- Primitive vs Object reference (https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html)
- Garbage Collector

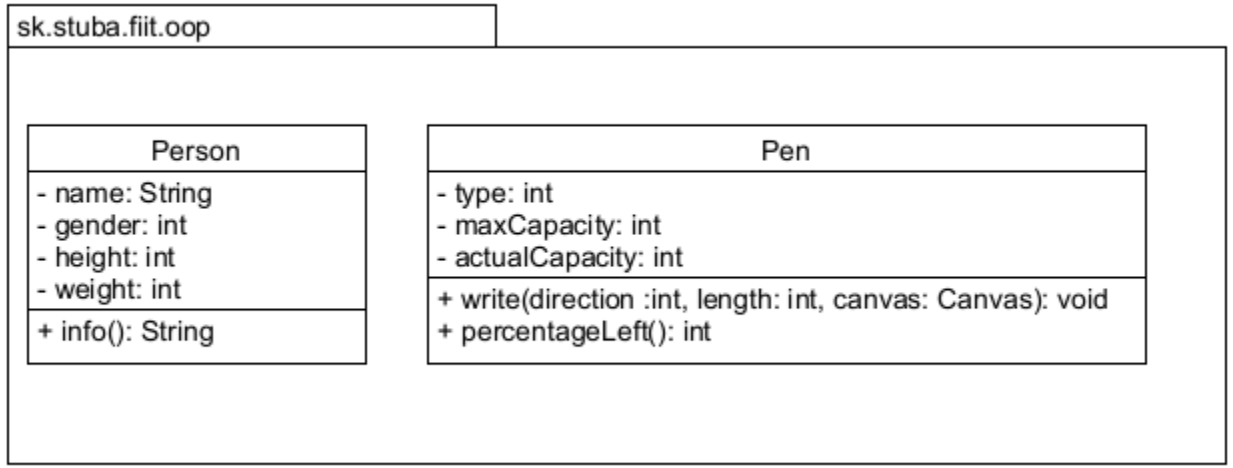
Trieda

- Predloha (blueprint) pre objekty
- Atribúty
- Metódy

- Vnorené triedy

(UML)

- Unified Modeling Language



Trieda vs Objekt

- Trieda = Predloha na tvorbu objektov
- Objekt = inštancia triedy, keyword new

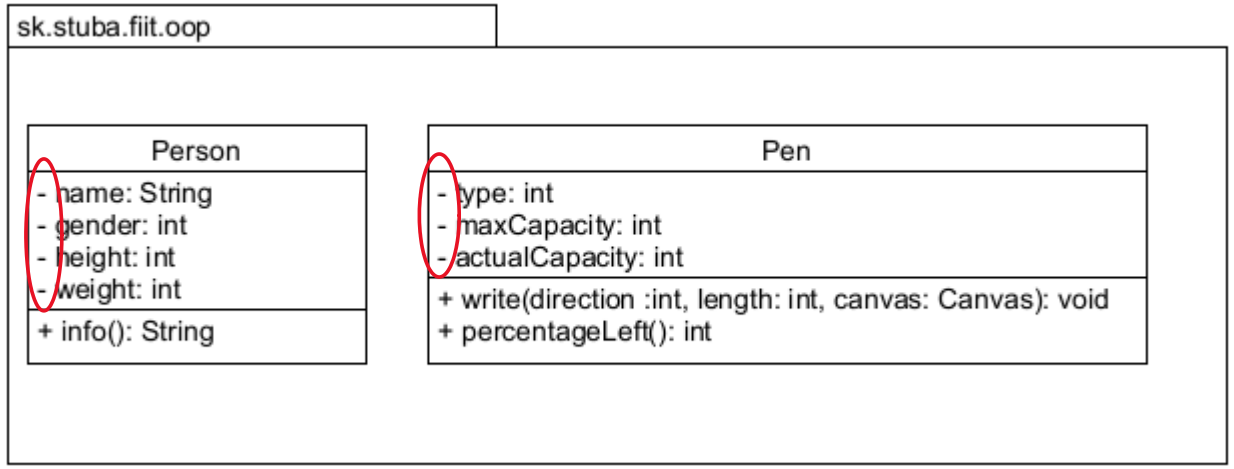
- Jablko (trieda), moje jablko (objekt), tvoje jablko (objekt)

Základné prvky OOP

- Enkapsulácia (zapuzdrenie)
- Dedičnosť
- Polymorfizmus
- Abstrakcia

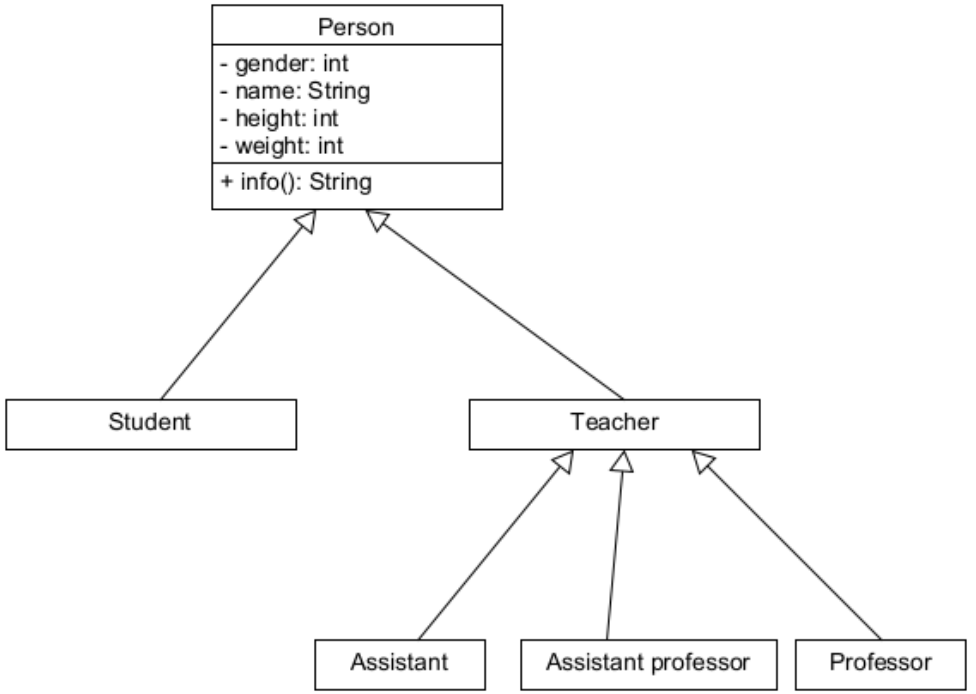
Enkapsulácia (zapuzdrenie)

- Nemanipulujeme priamo s dátami, ale pomocou metód
- V Jave napr. keyword private
- Mutátory. ale nie len to



Dedičnosť

- Umožňuje „odvodzovanie“ tried z iných tried
- Vzniká strom
- V Jave keyword extends



Polymorfizmus

- Compile time
  - Method overloading – rovnaké meno metód, rôzne parametre
- Runtime
  - Method overriding – pri dedení
  - Virtual functions

Abstrakcia

- “Skrývanie“ internej implementácie
- Abstrakcia je na úrovni návrhu (design), enkapsulácia na úrovni implementácie
- V Jave napr. Abstract class a Interface