Predmet:
Základy objektovo-orientovaného programovania 2023/2024
Úvodné informácie

Ján Lang

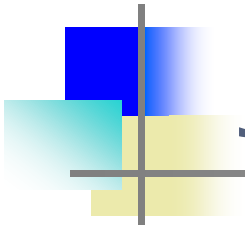
kanc. 4.34, jan.lang@stuba.sk, http://www2.fiit.stuba.sk/~lang/zoop/

Ústav informatiky, informačných systémov a softvérového inžinierstva

Fakulta informatiky a informačných technológií

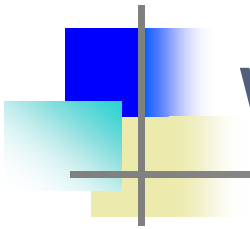
Slovenská technická univerzita v Bratislave

19. Septembra 2023



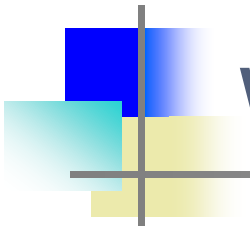
Ján Lang

- http://www2.fiit.stuba.sk/~lang/
- **kanc. 4.34**
- **jan.lang@stuba.sk**
- **Výučba:** ZOOP, OOP, MSOFT, MIP, TP, AOVS, ...
- **Výskum:** Object-oriented analysis and design, software modeling, software engineering, software development methodology, complex event processing, educational content engineering, applying software engineering principles in education and educational content modelling
- **Vedenie záverečných prác:** BP:20+, DP:30+



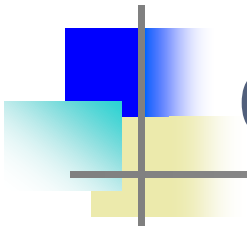
Web predmetu

http://www2.fiit.stuba.sk/~lang/zoop/



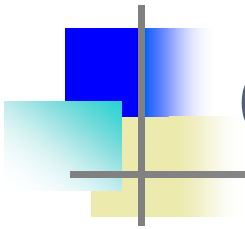
Web predmetu

http://www2.fiit.stuba.sk/~lang/zoop/
http://www2.fiit.stuba.sk/~lang/zoop/cvicenia
http://www2.fiit.stuba.sk/~lang/zoop/prednasky
http://www2.fiit.stuba.sk/~lang/zoop/podmienky



Cieľ predmetu

- Cieľom je naučiť sa základné princípy objektovo-orientovaného programovania. Osvojiť si spôsob objektovo-orientovaného myslenia
- Predmet sa zameriava na základy objektovo-orientovaného programovania: pojem objektu, triedy, agregácie a dedenia,...
- Programovací jazyk **Java je len prostriedkom** na dosiahnutie sledovaného cieľa



Charakteristika predmetu

- Dotácia hodín (prednáška/cvičenie): 2/2
- Ukončenie a kredity: skúška (**6 kreditov**)
- Rozsiahla praktická časť predpokladá zvládnutie radu **úloh na cvičeniach** a mimo nich v programovacom jazyku Java
- Seminárna časť sa venuje diskusii typických problémov pri tvorbe objektovo-orientovaných programov
- Vedomosti z predmetu Základy procedurálneho programovania, Procedurálne programovanie sú predpokladom
- **Prakticky zameraný** bakalársky predmet

Zložky hodnotenia predmetu

Priebežné hodnotenie predmetu - **65 bodov (D?)**

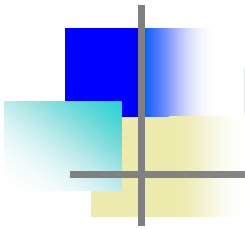
- predstavuje 65% celkového hodnotenia a pozostáva z týchto zložiek:

- riešenie úloh na cvičeniach - 10 bodov
- realizácia zadaní - 45 bodov
- semestrálny test - 10 bodov. V prípade, že výučba bude prebiehať online. Test nebude a body za test budú rozdelené za prácu na zadaniach nasledovne: Zadanie 2. +4 body, Zadanie 3. +6 bodov.

Záverečné hodnotenie - **35 bodov**

- je realizované písomnou skúškou, ktorá predstavuje 35% celkového hodnotenia

- skúška - 35 bodov



Úlohy na cvičeniach

Riešenie úloh na cvičeniach - 10 bodov

- Na cvičeniach budete precvičovať programovanie v jazyku Java riešením úloh, ktoré nadväzujú na odprednášaný obsah a budete pracovať na zadaniach
- Využite cvičenia na konzultovanie nejasností
- Cvičenia predstavujú len časť času, ktorý je potrebný na realizáciu úloh a zadaní
- Na úspešné riešenie úloh je potrebná príprava, ktorá pozostáva zo sledovania prednášok a štúdia literatúry
- Na úspešné zvládnutie zadaní je nevyhnutná dodatočná práca na zadaniach mimo cvičení
- Na základe pozorovanej aktivity študenta a výsledkov na cvičeniach učiteľ pridelí zodpovedajúci počet bodov
- Hodnotenie bude zverejnené v AIS-e do dvanásteho cvičenia

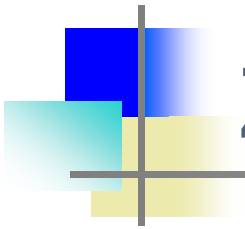
Úlohy na cvičeniach

- Od študenta sa očakáva aktívny prístup: aby získal body, študent sa musí sám hlásiť a preukázať svoju prácu
- Učiteľ môže akceptovať dodatočné preukázanie zvládnutia úloh a primerane to zobrať do úvahy pri hodnotení
- Odovzdanie zadaní v akceptovateľnom stave a v termíne je ďalšou podmienkou úspešného ukončenia predmetu
- Zvládnutie jednotlivých úloh a zadaní predstavuje cestu k pochopeniu základov objektovo-orientovaného programovania
- Priebežná práca počas semestra priamo prispieva k úspešnému zvládnutiu testov (semestrálneho testu a skúšky)
- Náplň cvičení bude postupne zverejňovaný v súlade s aktuálne prednášaným obsahom
- Cvičenia prebiehajú **prezenčne**

Úlohy na cvičeniach

Spôsob hodnotenia je nasledujúci:

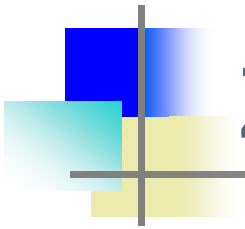
- pripravenosť na prácu, aktivita na cvičeniach a zvládnutie úloh na všetkých cvičeniach – 9–10 b
- pripravenosť na prácu, aktivita na cvičeniach a zvládnutie úloh väčšiny cvičení – 6–8 b
- ešte stále akceptovateľná aktivita na cvičeniach – 0–5 b



Zadania

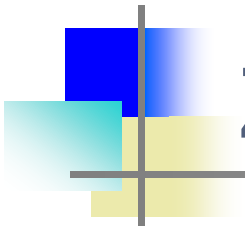
Realizácia zadaní spolu - 45 bodov

- Počas semestra budete vypracovávať zadania
- Na odovzdanie vypracovaných zadaní budú pripravené miesta odovzdávania v Akademickom informačnom systéme
- Termíny a spôsob odovzdania sú striktné a nebude možné dodatočné odovzdanie
- Na vypracovanie zadaní budete potrebovať aj čas mimo cvičení.
- Vypracovanie všetkých zadaní a ich odovzdanie najneskôr v stanovených termínoch je podmienkou absolvovania predmetu
- V prípade nesplnenia akejkoľvek podmienky akceptovania resp. neodovzdania predmet končí hodnotením FX
- Študent nemá nárok na opravné odovzdanie



Zadania

- Odovzdané zadanie musí byť vlastnou prácou študenta, ktorý ho odovzdáva
- Práca na zadaniach je individuálna
- Nie je povolené vytváranie skupín
- Výmena skúsenosti s kolegami je vítaná pokiaľ neprekračuje pravidlá akademickej korektnosti
- Neposkytujte však kolegom svoje zadania, ani ich žiadnym spôsobom nezverejňujte



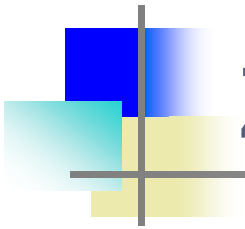
Zadania

Celkovo tri zadania počas výučby v zimnom semestri:

- Zadanie 1: Vypracovanie spresnenia rámcového zadania v zmysle určenia oblasti a jej pojmov, ktorým zodpovedajú abstrakcie rámcového zadania
- Očakávaný rozsah spresnenia je max. 500 slov
- Zadanie podlieha schváleniu a prípadným úpravám zo strany vyučujúceho na cvičeniach

Spôsob hodnotenia je nasledujúci:

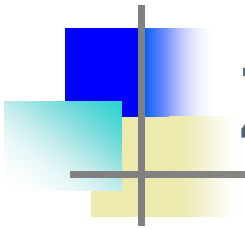
- prehľadný a jasný zámer – 5 b
- prevažne prehľadný a jasný zámer alebo neadekvátny alebo chýbajúci vlastný názov projektu – 3–4 b
- neprehľadný a nejasný zámer – 0–2 b



Zadania

Celkovo tri zadania počas výučby v zimnom semestri:

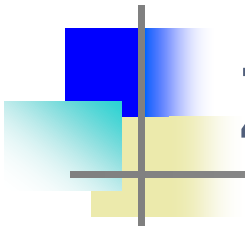
- Zadanie 2: Implementácia vlastnej konkretizácie rámcového zadania objektovo-orientovaným prístupom
- Očakávaný rozsah - uplatnenie všetkých dotknutých objektovo-orientovaných princípov relevantne progresu odprednášaného obsahu do termínu odovzdania tohto zadania, diagram vlastných tried a zoznam uplatnených objektovo-orientovaných princípov



Zadania

Zadanie 2, spôsob hodnotenia je nasledujúci:

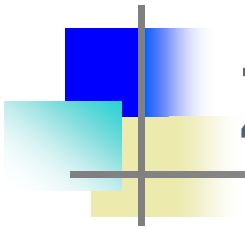
- Plne funkčný kód v jave a zároveň prehľadné a jasné uplatnenie dotknutých objektovo-orientovaných princípov deklarovaných zároveň v súbore pouziteOopPrincipy.pdf, korektný UML diagram tried – 14–15 b
- Funkčný kód v jave, prevažne prehľadné a jasné uplatnenie dotknutých objektovo-orientovaných princípov deklarovaných zároveň v súbore pouziteOopPrincipy.pdf, neadekvátny alebo chýbajúci UML diagram tried – 7–13 b
- Ešte stále akceptovateľný kód v jave, menej prehľadné až nejasne uplatnenie dotknutých objektovo-orientovaných princípov neadekvátne deklarovaných resp. nedeklarovaných v súbore pouziteOopPrincipy.pdf, neadekvátny resp. chýbajúci UML diagram tried – 0–6 b



Zadania

Celkovo tri zadania počas výučby v zimnom semestri:

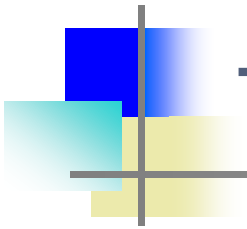
- Zadanie 3: Rozšírenie/doplnenie/aktualizácia implementácie vlastnej konkretizácie rámcového zadania objektovo-orientovaným prístupom, zapracovanie pripomienok z predchádzajúceho zadania
- Očakávaný rozsah - uplatnenie všetkých dotknutých objektovo-orientovaných princípov relevantne progresu odprednášaného obsahu do termínu odovzdania tohto zadania, aktualizácia diagramu vlastných tried a zoznamu uplatnených objektovo-orientovaných princípov



Zadania

Zadanie 3, spôsob hodnotenia je nasledujúci:

- Plne funkčný kód v jave a zároveň prehľadné a jasne uplatnené dotknuté objektovo-orientované princípy deklarované zároveň v súbore pouziteOopPrincipy.pdf, korektný UML model tried – 19–25 b
- Funkčný kód v jave, prevažne prehľadné a jasné uplatnené dotknutých objektovo-orientovaných princípov deklarovaných zároveň v súbore pouziteOopPrincipy.pdf, neadekvátny alebo chýbajúci UML model tried – 9–18 b
- Ešte stále akceptovateľný kód v jave, menej prehľadné až nejasne uplatnenie dotknutých objektovo-orientovaných princípov neadekvátne deklarovaných resp. nedeklarovaných v súbore pouziteOopPrincipy.pdf, neadekvátny resp. chýbajúci UML model tried – 0–8 b



Test

Semestrálny test - 10 bodov

- Obsahuje uzavreté otázky (úlohy s výberom odpovede). Tento rok plánujem rozšírenie aj o otvorenú/produkčnú úlohu
- Semestrálny test. V prípade, že výučba bude prebiehať online. Test nebude. Plánovaný na 7. novembra 2023 o 16:00
- Opravný termín nie je
- Náhradný termín je možný len pri neprítomnosti z oprávnených dôvodov riadne registrovanej na študijnom oddelení

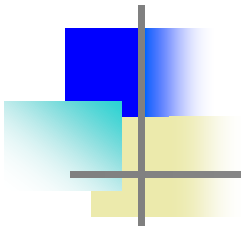
Skúška

Skúška - Záverečné hodnotenie - 35 bodov

- Písomnou formou, ktorá predstavuje 35% celkového hodnotenia
- Obsahuje uzavreté otázky (úlohy s výberom odpovede) ako aj otvorenú/produkčnú úlohu

Ukončenie predmetu

- Pre predmet platia univerzitné a fakultné podmienky absolvovania a hodnotenia predmetov
- Okrem toho študent musí realizovať **všetky zadania** v akceptovateľnej podobe a odovzdať ich najneskôr v stanovených termínoch
- Študent, ktorý sa dopustí plagiátorstva v projekte v hocijakom rozsahu, bude hodnotený známkou FX



Osnova predmetu

1. Štruktúrované prístupy k návrhu softvéru. Koncept abstraktného dátového typu
2. Štrukturálne koncepty objektovo-orientovaného prístupu: trieda, objekt Trieda ako prostriedok implementácie abstraktného dátového typu. Objekt ako inštancia triedy. Ich vlastnosti
3. Objektovo-orientované programovanie v jazyku Java. Integrované vývojové prostredie Eclipse pre Javu. Organizácia programových súborov a zdrojových súborov
4. Atribúty - deklarácia, typy, menné konvencie, použitie a modifikátory prístupu
5. Odkazy na objekty, referencovanie, priraďovanie objektových premenných, rekurzia, zreťazenie, agregácia
6. Metódy - deklarácia, parametre metód, primitívne typy, objektové typy, modifikátory prístupu

Osnova predmetu

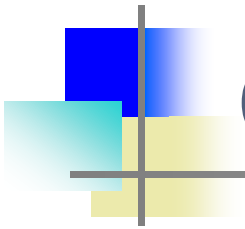
7. Zapuzdrenie. Atribúty a metódy triedy - statické. Bezparametrický konštruktor, parametrické konštruktory
8. Organizácia tried do balíkov, balíky, príslušnosť triedy k balíku, prístupové práva
9. Dedičnosť. Hierarchia tried
10. Preťažovanie a prekonávanie metód, polymorfizmus
11. Rozhrania, deklarácia a využitie. Implementácia viacerých rozhraní súčasne. Abstraktné triedy
12. Základné analytické postupy na vytváranie objektovo-orientovaného modelu

Odporúčaná literatúra, inf. zdroje

- Y. Liang. Introduction to Java Programming and Data Structures, Comprehensive Version. PEARSON Education Limited, 2021.
- Bart Baesens, Aimée Backiel, Seppe vanden Broucke. Java Programming. The Object-Oriented Approach. Wrox, 2015.
- Valentino Vranić. *Objektovo-orientované programovanie: Objekty, Java a aspekty*. Vydavateľstvo STU, 2008. (Errata)
- Bruce Eckel. *Thinking in Java.* 3rd edition, Prentice-Hall, 2002 HTML | PDF
- Bertrand Meyer. *Object-Oriented Software Construction.* Prentice Hall, 2nd edition, 1997
- Pavel Herout. Učebnice jazyka Java, Kopp, 2003
- Bogdan Kiszka. 1001 tipů a triků pro programování v jazyce Java, Computer Press, 2003
- James Keogh. Java bez předchozích znalostí, Computer Press, 2005

Odporúčaná literatúra, inf. zdroje

- Y. Liang. Introduction to Java Programming and Data Structures, Comprehensive Version. PEARSON Education Limited, 2021.
- Bart Baesens, Aimée Backiel, Seppe vanden Broucke. Java Programming. The Object-Oriented Approach. Wrox, 2015.
- Valentino Vranić. *Objektovo-orientované programovanie: Objekty, Java a aspekty*. Vydavateľstvo STU, 2008. (Errata)
- **Bruce Eckel. *Thinking in Java.* 3rd edition, Prentice-Hall, 2002 HTML | PDF**
- Bertrand Meyer. *Object-Oriented Software Construction.* Prentice Hall, 2nd edition, 1997
- Pavel Herout. Učebnice jazyka Java, Kopp, 2003
- Bogdan Kiszka. 1001 tipů a triků pro programování v jazyce Java, Computer Press, 2003
- James Keogh. Java bez předchozích znalostí, Computer Press, 2005



Odporúčaná literatúra, inf. zdroje

- Rudolf Pecinovský. Myslíme objektově v jazyku Java, Grada, 2008
- Kendal Simon.: Object Oriented Programming using Java. Simon Kendal & Ventus Publishing ApS, 2009. [Online] http://bookboon.com/
- Java™ Platform, Standard Edition 20 API Specification [Online] https://docs.oracle.com/en/java/javase/20/docs/api/index.html

Prednáška 1: Štruktúrované prístupy k návrhu softvéru. Koncept abstraktného dátového typu Štrukturálne koncepty objektovo-orientovaného prístupu: trieda, objekt Trieda ako prostriedok implementácie abstraktného dátového typu. Objekt ako inštancia triedy. Ich vlastnosti.

Ján Lang

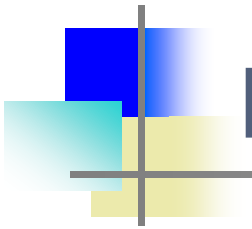
kanc. 4.34, jan.lang@stuba.sk, http://www2.fiit.stuba.sk/~lang/zoop/

Ústav informatiky, informačných systémov a softvérového inžinierstva

Fakulta informatiky a informačných technológií

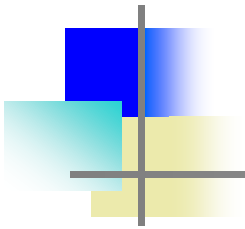
Slovenská technická univerzita v Bratislave

19. Septembra 2023



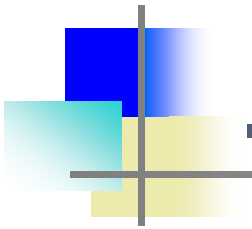
Programovanie???

...prečo práve objektovo-orientovaným spôsobom?



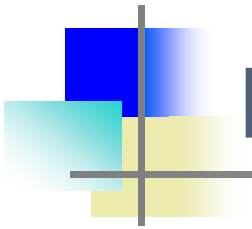
Motivácia

- Nárast zložitosti programových produktov
- Obmedzená ak vôbec nejaká modularizácia
- V minulosti sémantická medzera medzi strojovým jazykom a vyššími programovacími jazykmi
- Narastajúce požiadavky na kvalitu programov
- Prekonanie problémov - vhodná paradigma programovania (procedurálne, objektovo-orientované, funkcionálne, logické programovanie a pod.)
- K zmene paradigmy - prevládajúceho vedeckého názoru v danej oblasti - dochádza zlomom, teda revolúciou, nie postupne - evolúciou (Thomas Kuhn)



...???

Naprogramujte aplikáciu na evidenciu charakteristík našich študentov.
...v čom by ste to kódili?



Procedurálne programovanie

...existuje tu nejaká nadväznosť?

Procedurálne programovanie

Štruktúry

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Čo je to štruktúra?

- Štruktúra je heterogénny dátový typ, teda typ, ktorý je zvnútra zložený z prvkov rôznych typov a navonok je možné k nemu pristupovať ako k jednému objektu.

```
struct {
    položka_1;
    ...
    položka_n;
}
```

dá sa definovať piatimi rôznymi spôsobmi

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Definícia štruktúry (1)

- základný spôsob
  - štruktúra nie je pomenovaná,
  - nedá sa ďalej v programe použiť
  - dajú sa použiť len definované premenné

```
struct {
     int vyska;
     float vaha;
} peter,pavol,jan;
```

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Definícia štruktúry (2)

- modifikácia základného spôsobu
  - štruktúra je pomenovaná,
  - dá sa využiť aj ďalej v programe

```
struct miery {
     int vyska;
     float vaha;
} peter,pavol,jan;
```

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Definícia štruktúry (3)

- podobne ako predchádzajúci spôsob
  - definícia štruktúry je oddelená od definície premenných (ktoré sa môžu definovať viackrát)

```
struct miery {
     int vyska;
     float vaha;
};
struct miery pavol;
struct miery jan,peter;
```

Definícia štruktúry (4)

- definícia nového typu (typedef)
  - štruktúra nie je pomenovaná, pomenovaný je typ
  - typ sa dá použiť na definíciu premenných, pretypovanie...

```
typedef struct {
    int vyska;
    float vaha;
} MIERY;
MIERY pavol, jan, peter;
```

nebolo použité "**struct**"

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Definícia štruktúry (5)

- modifikácia predchádzajúceho spôsobu
  - štruktúry aj typ je pomenovaná (v tomto prípade to nie je potrebné, ale nutné ak štruktúra odkazuje sama na seba

```
typedef struct miery {
    int vyska;
    float vaha;
} MIERY;
MIERY pavol, jan, peter;
```

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

odporúča sa pomenovať typ aj štruktúru rovnako, odlíšiť ich len veľkosťou písma

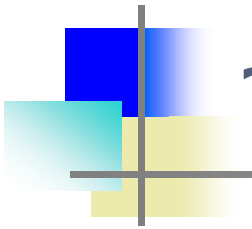
Používanie štruktúr

- odporúča sa používať definície typu (4) a (5)
  - je to prehľadnejšie ("**struct**" sa použije len raz)

```
typedef struct {
    int vyska;
    float vaha;
} MIERY;
MIERY pavol, jan, peter;
```

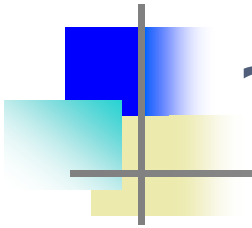
```
typedef struct miery {
    int vyska;
    float vaha;
} MIERY;
MIERY pavol, jan, peter;
```

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková



???

Aké typy môže obsahovať štruktúra???

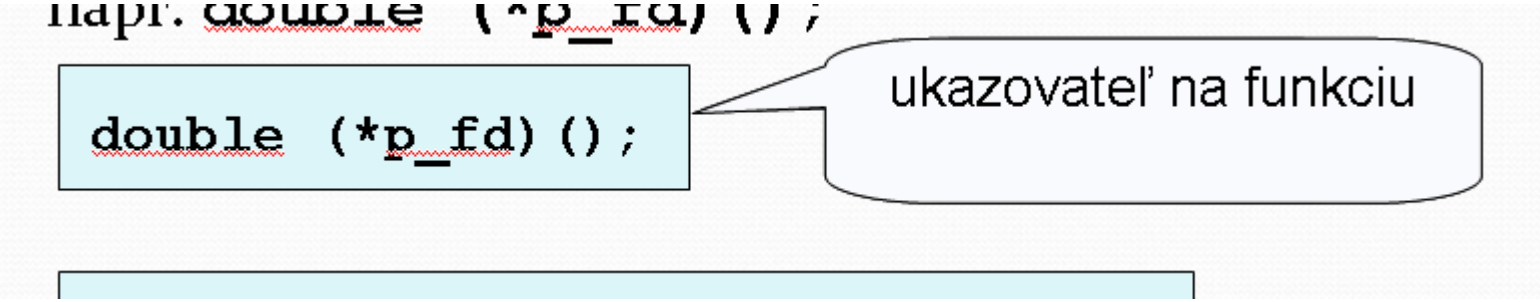


???

Štruktúry môžu obsahovať len údaje???

Integrovaná funkcionalita?

- Avšak máme ukazovatele na funkcie



Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Ukazovatele na funkcie

- Funkcia môže vrátiť ukazovateľ na typ:
  - **`FILE *fopen(...)`** vracia smerník na typ **`FILE`**
- Definovanie premennej ako ukazovateľ na funkciu: napr. **`double (*p_fd)();`**

```
double (*p_fd)();
```

ukazovateľ na funkciu

```
double scitaj(double x, double y)
p_fd = scitaj;
```

`p_fd` má adresu funkcie **`scitaj()`**

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Príklad ukazovateľa na funkciu

funkcia na výpočet hodnôt polynómov (napr. $x^2 + 3$, $x + 8$) pre zadanú hornú, dolnú hranicu a krok
- najprv pomocné funkcie pre polynómy

```
double pol1(double x)
{
    return  (x * x + 3);
}
```

```
double pol2(double x)
{
    return  (x + 8);
}
```

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Príklad ukazovateľa na funkciu

funkcia **`vypis()`** na vypísanie tabuľky

```
void vypis(double d, double h, double k,
double (*p_f)()) {
     double x;
     for(x=d; x<=h; x+=k)
          printf("%lf, %lf \n", x, (*p_f)(x));
}
```

volanie
vo funkcii main():

```
vypis(-1.0, 1.0, 0.1, pol1);
vypis(-2.0, 2.0, 0.05, pol2);
```

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Príklady definícií

```
int i;          -
float *y;       -
double *z();    -

int (*v)();     -

int *(*v)();    -
```

???

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

Príklady definícií

```
int i;
```
- **i** je typu **int**

```
float *y;
double *z();

int (*v)();

int *(*v)();
```

Príklady definícií

```
int i;
```
- **i** je typu **int**

```
float *y;
```
- **y** je ukazovateľ na typ **float**

```
double *z();
```

```
int (*v)();
```

```
int *(*v)();
```

Príklady definícií

```
int i;
```
- **i** je typu **int**

```
float *y;
```
- **y** je ukazovateľ na typ **float**

```
double *z();
```
- **z** je funkcia vracajúca ukazovateľ na **double**

```
int (*v)();
```

```
int *(*v)();
```

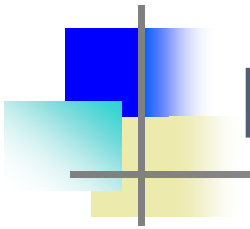
Príklady definícií

`int i;`	- **i** je typu **int**
`float *y;`	- **y** je ukazovateľ na typ **float**
`double *z();`	- **z** je funkcia vracajúca ukazovateľ na **double**
`int (*v)();`	- v je ukazovateľ na funkciu vracajúcu **int**
`int *(*v)();`	

Príklady definícií

`int i;`	- **`i`** je typu **`int`**
`float *y;`	- **`y`** je ukazovateľ na typ **`float`**
`double *z();`	- **`z`** je funkcia vracajúca ukazovateľ na **`double`**
`int (*v)();`	- v je ukazovateľ na funkciu vracajúcu **`int`**
`int *(*v)();`	- v je ukazovateľ na funkciu vracajúcu ukazovateľ na **`int`**

Zdroj: Prednášky z predmetu Procedurálne programovanie, Anna Bou Ezzeddine, Gabriela Kosková

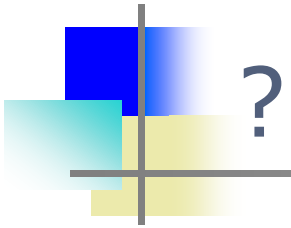


Resumé

Aj keď sa nástup objektovo-orientovaného programovania často vníma ako zlom paradigmy, korene objektovo-orientovaného programovania sú v skutočnosti v procedurálnom programovaní

Objektovo-orientované programovanie

- OOP, programovanie pomocou **objektov**
- Tvrdenie, že svet je svetom objektov, ktoré medzi sebou svojim spôsobom komunikujú (verbálne, neverbálne,...)
- Objekty môžeme pomyselne zoskupiť resp. rozdeliť do skupín - **tried**, ktoré majú spoločné vlastnosti
- Takéto skupiny označujeme ako **triedy**. Napr.:
  - **Trieda** 1.A je zoskupením žiakov, ktoré sa pravidelne stretáva, sleduje spoločný cieľ, študuje ten istý predmet/predmety,...
  - Skupina zdieľajúca ten istý ekonomický alebo sociálny status (z dôb temna napr. Robotnícka **trieda**, a pod.)
  - Z biológie základné taxonomické kategórie ríša, kmeň, **trieda**, rad, čeľaď, rod, druh,...
  - Vzor, najlepší svojho druhu, v kategórii ...to je **trieda**!
  - angl. **Class**

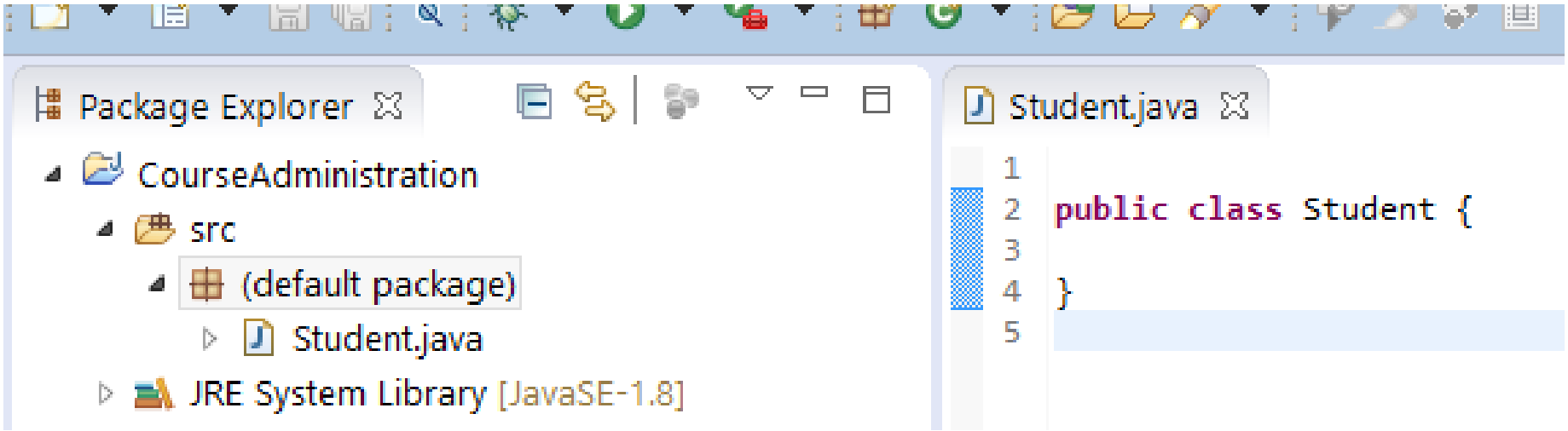


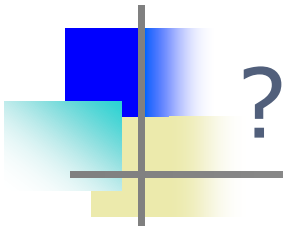
?

Môžeme študentov modelovať ako triedu???

Skúsme toto:

- Eclipse - File - New - Java Project - Project name(CourseAdministrattion)
- Package Explorer/CourseAdministrattion/src – New – Class(Student)





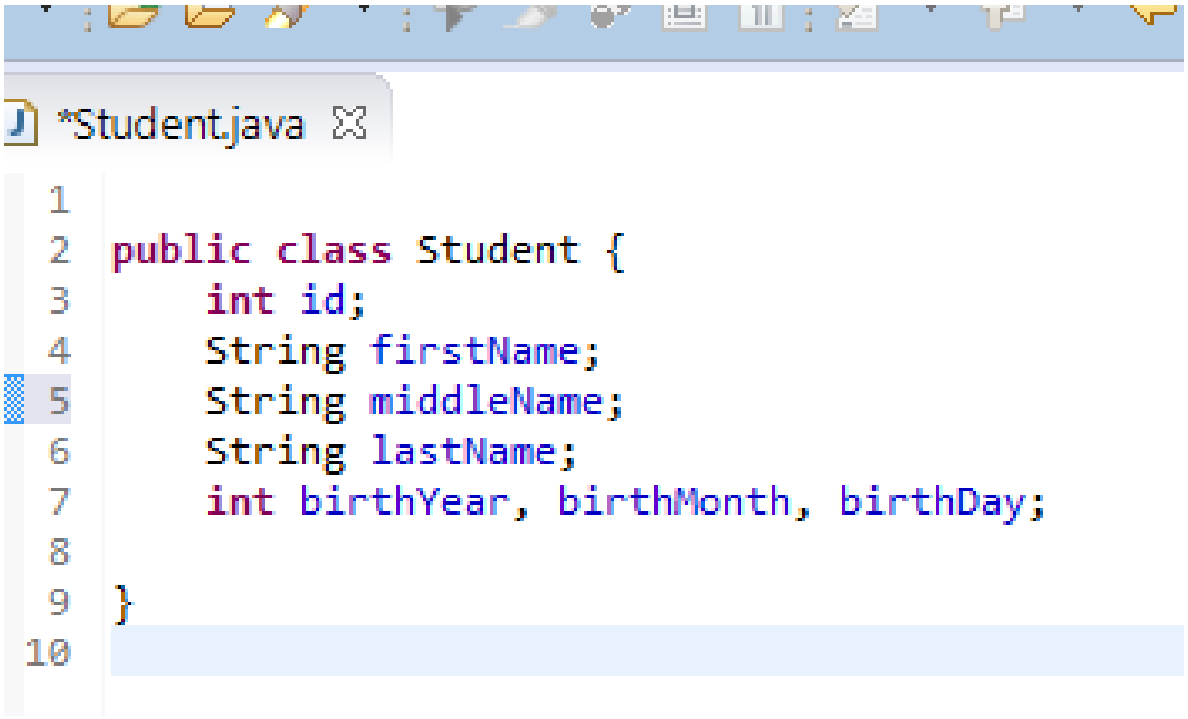
?

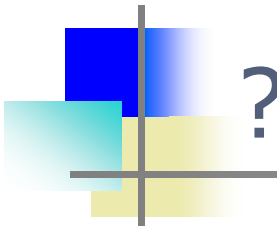
V akom stave sa nachádza náš študent???

Máme už jeho objekt???

Koncept študenta

- Definíciou študenta
- Vytvorme premenné reprezentujúce jeho identifikačné číslo, meno, priezvisko, a dátum narodenia (zatiaľ len v kombinácii troch celých čísel bez použitia triedy Date z java API 8 špecifikácie)

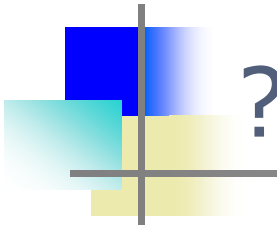




?

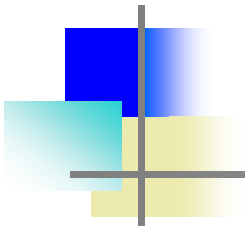
V akom stave sa nachádza náš študent???

Máme už jeho objekt???



?

Čo vlastne máme???



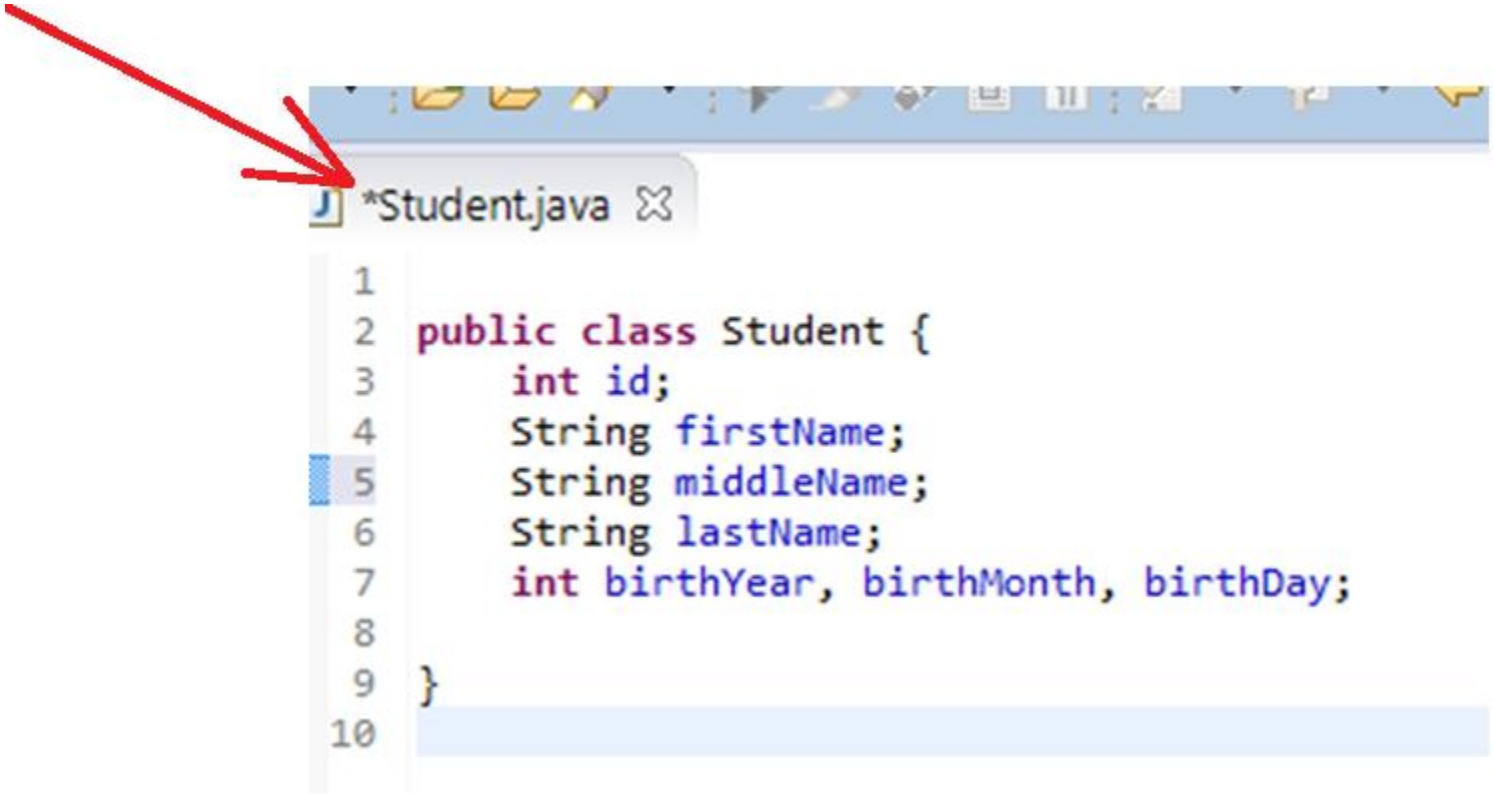
Máme

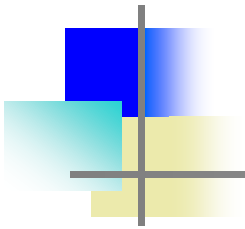
- Zdrojový súbor triedy študent
- Uložený v .java súbore (súbor má mať rovnomenné pomenovanie ako samotná trieda)
- Definovaný deklarovaním siedmych premenných
- Fyzicky prítomný v súborovom systéme. Na mieste označovanom ako Workspace (na ktoré sa Eclipse štandardne pýta pri jeho spustení)

- ...on je vlastne neuložený!!!

...neuložený zdrojový súbor

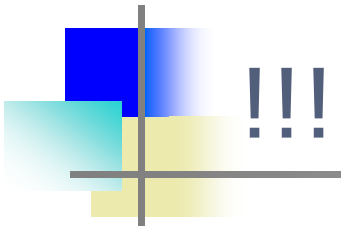
- Toto je v IDE Eclipse indikované príznakom *





Máme

- Definíciu triedy začatú, kľúčovým slovom **class**
- Nasledovanú názvom triedy (názov zvyčajne začína veľkým písmenom – v UpperCamelCase formáte)
- Telo triedy ohraničené { } (podobné ako pri štruktúrach...)
- V tele triedy atribúty - premenné (dáta)
- Definíciu premenných začínajúcu typom, názvom (jeden alebo viac) končiac bodkočiarkou (podobne ako každý príkaz v jazyku C)
- Názov premennej v lowerCamelCase formáte pre rozlíšenie medzi názvom triedy a názvom premennej - konvencia



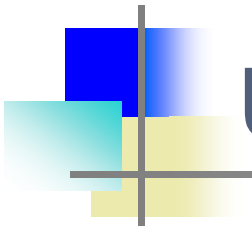
!!!

Čo ešte stále nemáme je - objekt!

Programovanie pomocou **objektov**

- Objekt = **inštancia** príslušnej triedy. Vlastné objekty označujeme ako inštancie príslušnej triedy
  - Napr. počítač s ktorým pracujem je objekt, inštanciou triedy napr. Počítač
  - Inštancia je konkrétnou realizáciou svojej triedy
    - Má **svoju** identitu
    - Má **svoj** stav
    - Má **svoje** správanie[1]
  - „**Svoje**" nakoľko vie existovať ako entita
  - Z latinského *ent-, ens* existujúca vec[2]

[1] G. Booch. Object-Oriented Analysis and Design with Applications. Addison-Wesley, 1994.
[2] http://www.merriam-webster.com/dictionary



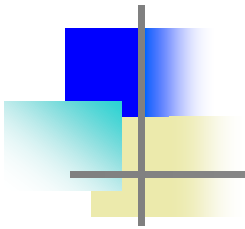
Úloha

Doplňme chýbajúce správanie

Koncept študenta

Student.java

```
1
2  public class Student {
3      int id;
4      String firstName;
5      String middleName;
6      String lastName;
7      int birthYear, birthMonth, birthDay;
8
9      boolean isBirthDay() {
10         //vráti true ak má narodeniny
11         return false;
12     }
13
14     int numberOfFriends() {
15         //vráti počet priateľov
16         return 0;
17     }
18
19     void giveWarning() {
20         //mal by sa viac snažiť
21     }
22 }
23
```



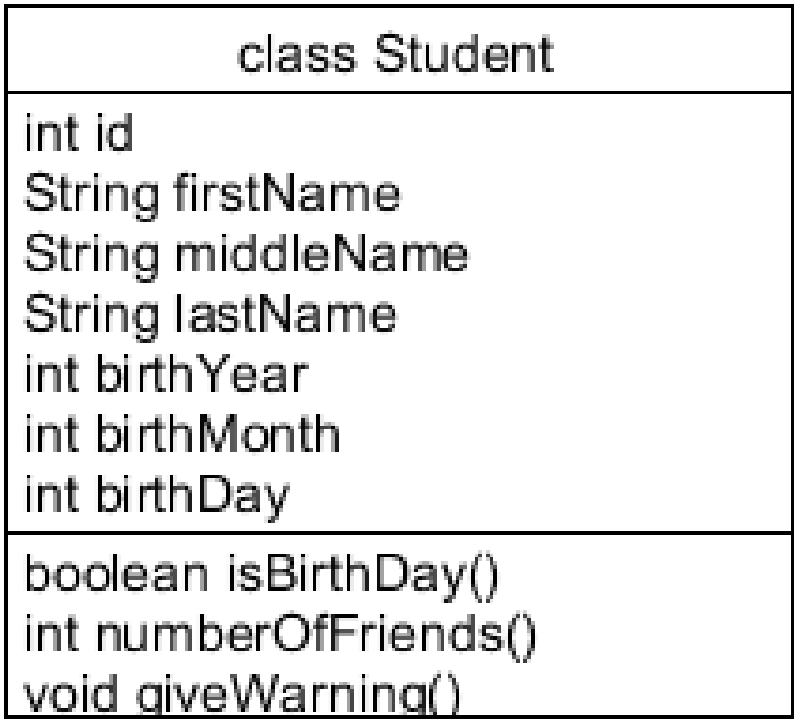
Máme

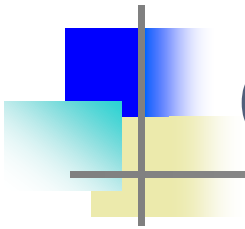
- Koncepciu máme doplnenú o tri **metódy**
- isBirthDay, numberOfFriends, giveWarning
- Metóda môže začínať návratovou hodnotou (v našom prípade - boolean, int, void - kedy sa neočakáva návratová hodnota)
- Telo metód zatiaľ bezvýznamne prázdne
- Možnosť ďalej pridávať definície metód prípadne kód do tiel metód
- Uplatnenú štábnu kultúru: najprv premenné potom metódy (upratený kód)
- Prototyp triedy “blueprint“

- ...stále nemáme inštanciu, objekt študenta

Máme

- Reprezentácia triedy v diagrame tried



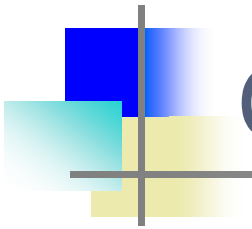


Objekt

- Pojem **objekt** a **inštancia** sú synonymá.
- Pojmu objekt sa dáva väčšia prednosť ak hovoríme všeobecne o objektoch
- Pojmu inštancia dávame prednosť vtedy ak chceme zdôrazniť do akej triedy objektov patrí (najmä ak ju vtedy spomíname)
- Objekty vytvárame pomocou kľúčového slova ***new***

```
new Student();
```

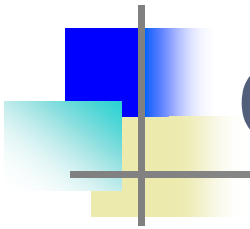
- Takto vznikne inštancia bez toho aby sme na ňu mali dosah



Objekt

```
new Student();
```

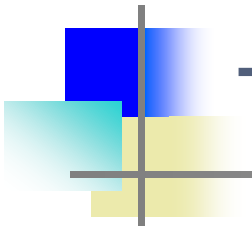
Potrebujeme mať k tej inštancii prístup?



Objekt

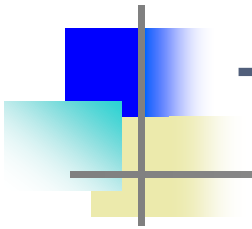
Preto:

```
Student myFirstStudent = new Student();
```



Trieda

Triedy popisujú spoločné vlastnosti svojich inštancií



Trieda

Uveďte príklady objektov z reálneho sveta, ktoré majú spoločné vlastnosti.
Abstrahujte detaily.

Trieda

- Asociácia pečiatky ako triedy, ktorej inštancie sú odtlačky pečiatky na papieri a pod.
- Vyvinuli sa zo štruktúr definovaných ako typy v jazyku C

```
typedef struct miery {
    int vyska;
    float vaha;
    int rocnik;
} MIERY ;
MIERY pavol, jan, peter;
```

```
public class Miery {
    private int vyska;
    private float vaha;
    private int rocnik;
    //chýbajú metódy
}
```

```
Miery pavol, jan, peter;
```

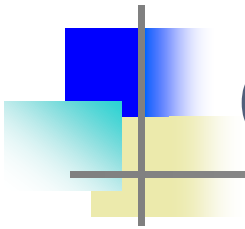
Trieda

- Vyvinuli sa zo štruktúr definovaných ako typy v jazyku C

```
typedef struct student {
    char[20] meno;
    char[20] priezvisko;
    int rocnik;
} Student;

Student x;
strcpy(x.meno,"Milan");
strcpy(x.priezvisko,"Milov");
x.rocnik= 1;
. . .
Student y = („Janko", „Mrkvička", 2);
```

Zdroj: Prednášky z predmetu Objektovo-orientované programovanie, Valentino Vranić



OO program

- Program sa uskutočňuje ako interakcia objektov. Interakcia - vzájomné alebo recipročné konanie, ovplyvňovanie
- Grafická interpretácia inštancií dokumentuje objekty, ktoré si navzájom posielajú správy. V podstate je to podnet na prejav dotknutého objektu. Môžeme sa napr. spýtať žiarovky či svieti. V reálnom živote to funguje tak, že sa pozrieme na žiarovku
- Priblíženie OOP prostredníctvom príkladu
  - **Course Administration system**
  - Identifikované objekty (student, course...
  - Vlastnosti objektov - stav (id, firstName, lastName,...) a správanie (isBirthday, moveTo,...)

OO program

V čom budeme naše príklady vyvíjať?

- Java - objektovo-orientovaný programovací jazyk "3. generace - 3GL" (imperativny jazyk vysokej úrovne), univerzálny (napísaný program v Jave sa dá vykonávať všade kde je dostupné Java runtime environment JRE). Inštalovaná v počítačoch v CPU. Do vlastných počítačov je potrebné nainštalovať java JDK (JRE pre vývoj nestačí) Java JDK - Java SE Development Kit (v. 8) https://www.oracle.com/java/technologies/downloads/#jdk19-windows

  (https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html)
- V akom prostredí (IDE)? Primárne v Eclipse, http://eclipse.org/ (alt. NetBeans, IntelliJ). Štandardne nainštalovaný na počítačoch v CPU. Do vlastných počítačov je potrebné nainštalovať IDE Eclipse https://www.eclipse.org/downloads/

Menné konvencie

- **Code Conventions for the Java™ Programming Language** Revised April 20, 1999[1]
- Použitie konzistentne správnej konvencie je správnou vizitkou programátora a vyjadruje jeho vzťah k poriadku
- **Malé písmená** Lowercase - while, if, package,...
- **Veľké písmená** Uppercase (ak ide o viacslovné prípady používame podčiarknik) - MAX_VALUE, ...
- CamelCase (**Upper CamelCase**) - kde každé nové slovo začína veľkým písmenom a vynechávajú sa medzery – CustomerAccount, CamelCase,...
- Mixed Case (**Lower CamelCase**) – je to isté ako CamelCase s tým, že prvé písmeno je malé - customerAccount, camelCase,...

[1] http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

Menné konvencie

- **Premenné** - pomenovania by mali byť v tvare mixed case, Mali by reprezentovať to čo reprezentuje hodnota premennej

```
string firstName
int orderNumber
```

- **Triedy** - je dobré ak sú písané CamelCase spôsobom. Mali by byť použité podstatné mená pretože pomenúvavajú veci reálneho sveta napr:

```
class Chatrc
class Zviera
```

- **Balíky** – mali by byť napísané malými písmenami. Taktiež by mala byť použitá konvencia zápisu, ktorá zohľadní doménu subjektu v reverznom tvare

```
sk.stuba.fiit.zoop
```

Príklad

```
public class Ahoj {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Ahoj");
    }

}
```

Príklad – s uvedením balíka

```
package sk.stuba.fiit.p01;

public class Ahoj {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Ahoj");
    }

}
```

Príklad – sprístupnením java.util.*

```
package sk.stuba.fiit.example01;

import java.util.*;

   public class HelloDate {

      /**
       * @param args
       */
      public static void main(String[] args) {
         System.out.println("Hello, it's: ");
         System.out.println(new Date());
   }
}
```

Príklad

```
package sk.stuba.fiit.p01;

import java.util.Scanner;

public class CitajZnakZKlavesnice {
    public static void main(String[] args) {

    Scanner sc = new
      Scanner(System.in).useDelimiter("\\s*");
        while (!sc.hasNext("z")) {
            char ch = sc.next().charAt(0);
            System.out.print("[" + ch + "] ");
        }
    }
}
```

Príklad

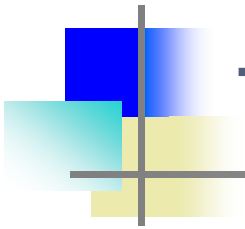
```
package sk.stuba.fiit.retazce;

    public class Hlavny {

    public static void main(String[] args) {
        String s = new String("abc");
        StringBuffer sb = new StringBuffer("xyz");

        s = "cdfg";
        s = "cdefghijkl";

        System.out.println(s);
        System.out.println(sb);
    }
}
```



TODO

- Váše odporúčanie, komentár či otázka k predmetu. Nájdete komunikačný formulár na webe predmetu
- ...po prednáške (ZOOP) (google.com)