Objektovo orientované programovanie

1. prednáška

Vladislav Novák

Rozvrh

Prednášky

pondelok 10:00 – 11:50

Cvičenie/seminár:

utorok 8:00 – 9:50

Vyučujúci

- prednášajúci:
  - RNDr. Július Šiška, PhD.
- cvičiaci:
  - Ing. Vladislav Novák
  - Ing. Filip Hodoň
  - Mgr. Dominika Závacká, PhD.

Podmienky hodnotenia

- Semester 40 bodov
  - 7 zadaní (domácich úloh)
    - za rôzne počty bodov, najprv jednoduchšie, neskôr zložitejšie
    - na vypracovanie približne týždeň
    - 20 bodov
  - písomky
    - 2 písomky
    - 20 bodov
- Skúška 60 bodov
  - Podmienkou účasti na skúšku je získať cez semester aspoň 20 bodov

Webová stránka

- ku predmetu
  - https://uim.fei.stuba.sk/predmet/b-oop/
- ku cvičeniam
  - https://useobjects.net/oop/

Java

- Verzia 25
  - najnovšia LTS (Long-Term-Support) verzia
- Novšie verzie majú kratšiu dobu podpory
- Túto verziu použite aj pri riešení zadaní!

Release	GA Date	Premier Support Until	Extended Support Until
25 (LTS)	September 2025	September 2030****	September 2033****
26 (non-LTS)***	March 2026	September 2026	Not Available
27 (non-LTS)***	September 2026	March 2027	Not Available

https://www.oracle.com/java/technologies/java-se-support-roadmap.html

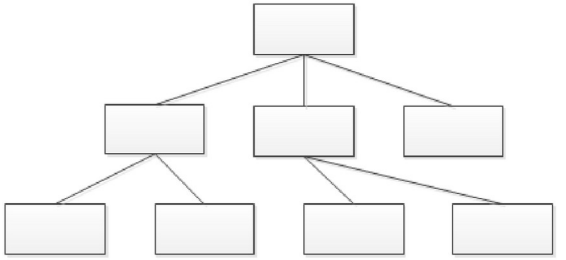
https://en.wikipedia.org/wiki/Java_version_history

Učivo

- Úvod, Java
- Objektovo orientované programovanie
- Návrhové vzory

Ako zvládnuť veľké projekty

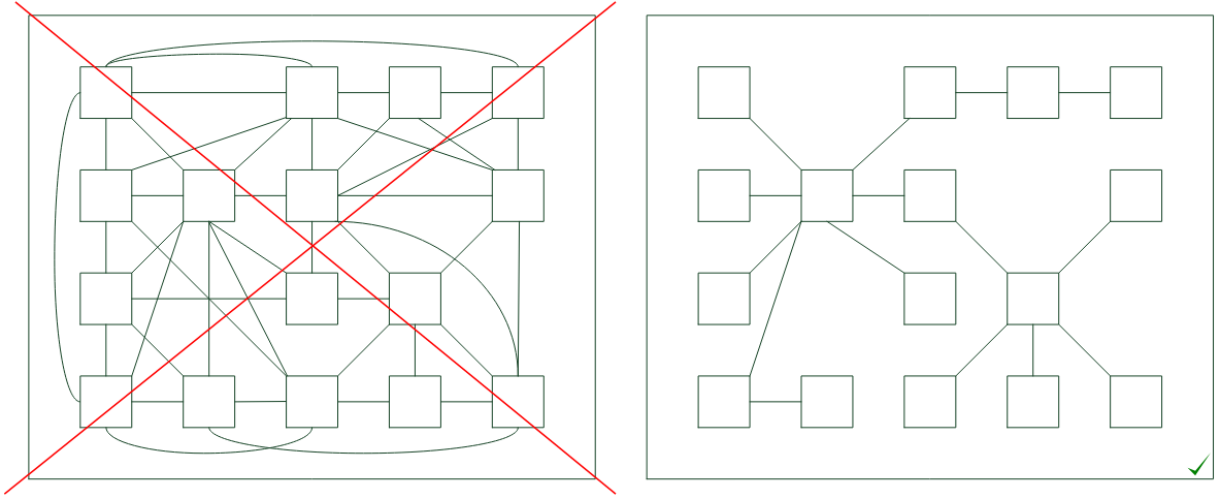
- rozdelením na menšie časti



- **minimalizovať** vzájomné **previazanie** jednotlivých častí (ideálne každú časť riešiť nezávysle na inej časti)
- **maximalizovať súdržnosť** každej časti systému (vždy sa zaoberať práve jednou častou systému)

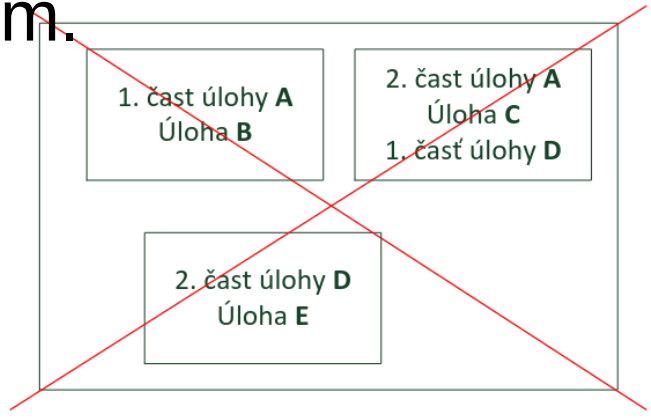
Vzájomná previazanosť (coupling)

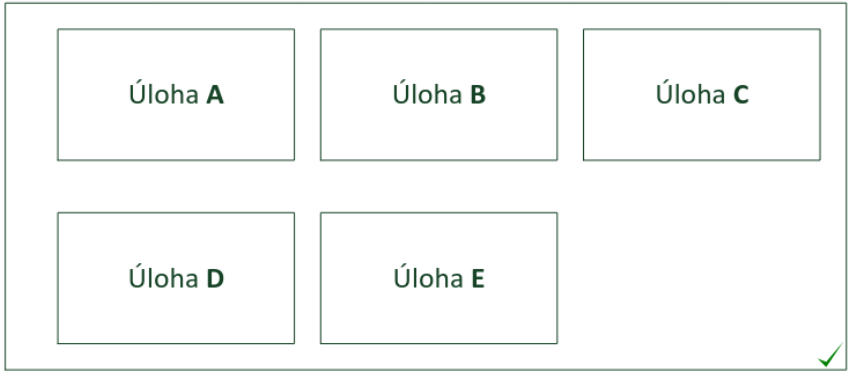
- Previazanosť je určená množstvom väzieb medzi softvérovými entitami. Treba ju minimalizovať. Minimalizácia množstva väziab medzi entitami znižuje vzájomnú závislosť.



Súdržnosť časti (cohesion)

- To čo spolu súvisí, je vhodné implementovať práve v jednej softvérovej entite. Nie je vhodné implementovať niekoľko vecí naraz. Súvislosť vyjadruje dodržanie tejto zásady.
- Súdržnosť je vhodné maximalizovať. Ak úloha je zložitejšia, treba jednotlivé podúlohy rozdeliť jednotlivým podsystémom.





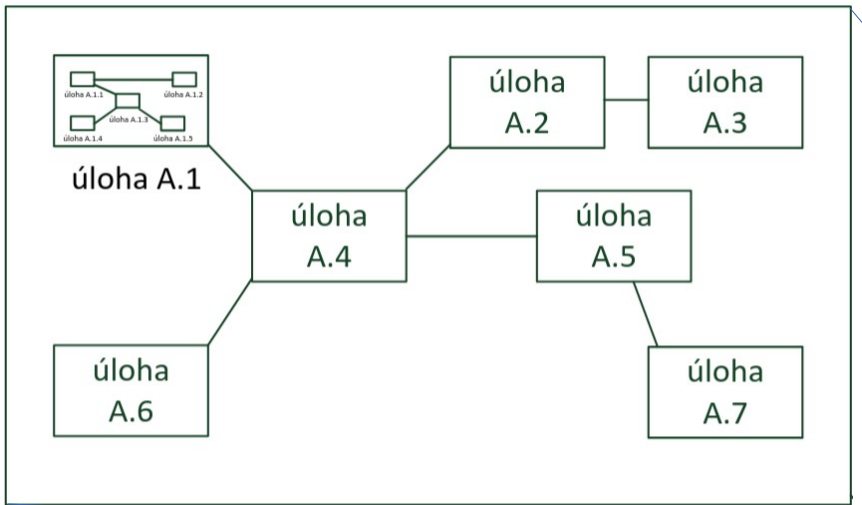
Softvérová entita

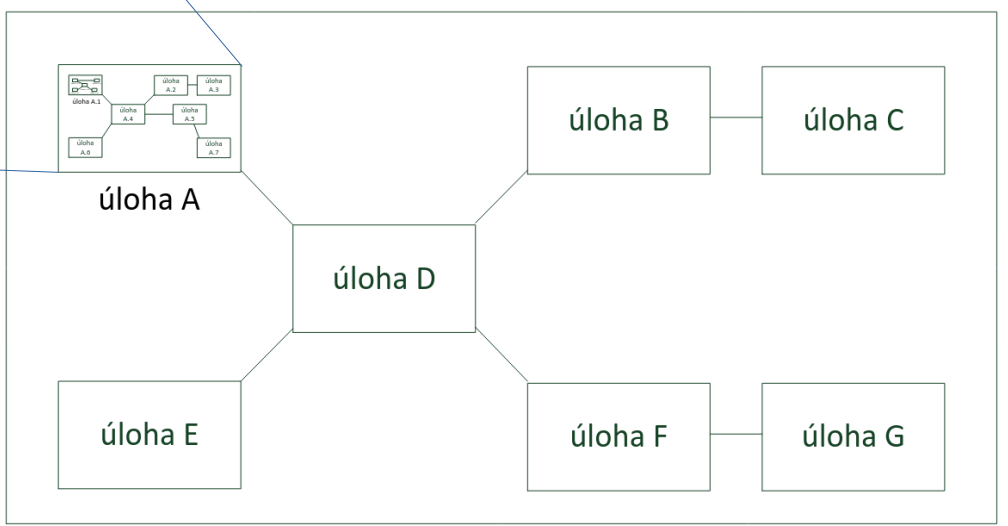
- funkcia / metóda
- trieda
- balík
- komponent, modul, . . .

Závislosť softvérových entít

- zmena niektorej časti entity, spôsobý potrebu niečo zmeniť aj v inej entite

Úloha – podúloha - podpodúloha





Príklad

- ukážka v zdrojákoch
  - createList(...)
    - funkcia má robiť celú funkcionalitu
  - appendAndPrintAndGetLength(...)
    - zvlášť funkica pre každú funkcionalitu
  - append(...)
    - podfunkcionalita môže byť v ďalšej funkcii

Grafické vývojové prostredia

- IntelliJ IDEA
  - https://www.jetbrains.com/idea/
  - stačí community edition (download → na web-stránke nižšie)
- Netbeans
  - https://netbeans.apache.org
- Eclipse
  - https://www.eclipse.org/downloads/packages/release/2023-12/r/eclipse-ide-java-developers

• IntelliJ IDEA

- návod v inom pdf
  - bude sa priebežne aktualizovať

Java - projekt

- v projekte môže byť viacero main-ov
- spúšťa sa vybraný main

std out (aj formátovaný), err, in

- System.out.println()
- System.err.println()
- IO.println()
- IO.readln()
- System.out.format()
- System.out.printf()

Intellij Idea – live templates

- v inom pdf

Primitívne vs referenčné typy

- Premenné v Jave majú definované dátový typy
- Rozdelenie
  - primitívne dátové typy
  - referenčné dátové typy
    - pozor na kopírovanie a porovnavanie referenčných typov!

Primitívne typy

```
int pocet = 10;
double vyska = 1.5;
char pismeno = 'p';
char pismeno2 = '\u3072'; //  ひ
boolean b = true;
```

Primitívne typy

Primitívne číselné typy sú znamienkové.
Celočíselné typy používajú doplnkový kód

- byte – 8 bitový, <-128, 127>
- short – 16 bitov, <-32 768, 32 767>
- int – 32 bitov, $<-2^{31}, 2^{31}-1>$
- long – 64 bitov, $<-2^{63}, 2^{63}-1>$
- float – 32 bitov, plávajúca desatinná čiarka, IEEE 754
- double – 64 bitov, plávajúca desačinná čiarka, IEEE 754
- char – 16 bitov, Unicode
  - unicode môže používať rôzne druhy kódovania
- boolean = {true, false},
  - pamäťová veľkosť nie je presne definovaná

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.htm

Primitívne vs referenčné typy

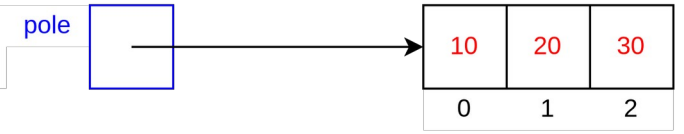
```
int dlzka = 10;
```


```
String text = "programovanie";
```

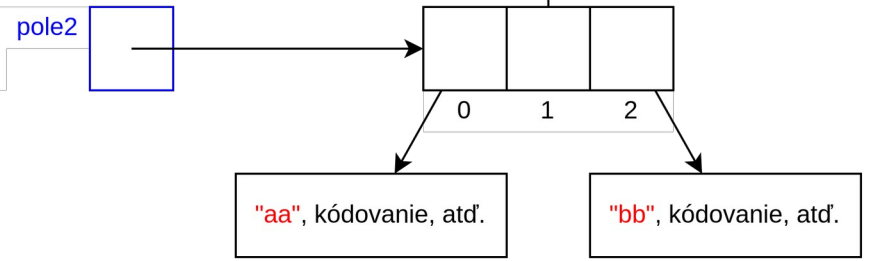

```
Color zlta = new Color(255, 255, 0);
```

https://www.csfieldguide.org.nz/en/interactives/rgb-mixer/

```
int [] pole = {10, 20, 30};
```




```
String [] pole2 = {"aa", text, "bb"};
```



Literály

```java
int a = 1;
long b = 1L;
int c = 1_000_000;

int d = 0xFF; // 15*16 + 15*1 = 255
System.out.println(d);

int e = 0b1100; // 8 + 4 + 0 + 0 = 12
System.out.println(e);

float  f = 1.2f; // f alebo F
double g = 1.2d; // d alebo D
double h = 1.234e2;
double i = 1.234e2d;
float  j = 1.234e2f;
System.out.println(h);
```

```java
boolean podmienka1 = true;
boolean podmienka2 = false;

char pismeno = 'a';

String text1 = "abcd";

String text2 = null;
```

linka: Escape Sequences

```java
String text3 = "jeden\ndva\n"; // \n \t \b
System.out.println(text3);

// String.class
```

Defaultné hodnoty

- nastavené v niektorých prípadoch

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Používanie objektov a tried

- String, StringBuilder
- Math