Informatika 2

Tvorba softvéru

doc. Ing. Ján Janech, PhD.
29. 4. 2024

ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Pojmy zavedené v 9. prednáške (1)

- komunikácia s používateľom
  - CLI
  - TUI
  - GUI

Pojmy zavedené v 9. prednáške (2)

- komponenty v SWING
  - JFrame
  - JLabel
  - JButton
  - JPanel
- layout v knižnici SWING
  - GridLayout
  - BorderLayout
  - kombinovanie
- metóda pack()

Pojmy zavedené v 9. prednáške (3)

- poslucháči
  - ActionListener
  - WindowListener/WindowAdapter

Pojmy zavedené v 9. prednáške (4)

- vnorená trieda
  - syntax v jazyku Java
  - prístup k vnútornému pohľadu vonkajšej triedy
  - anonymná trieda
  - uzávery

Cieľ prednášky

- analýza problému
- architektonické návrhové vzory
- návrhové princípy

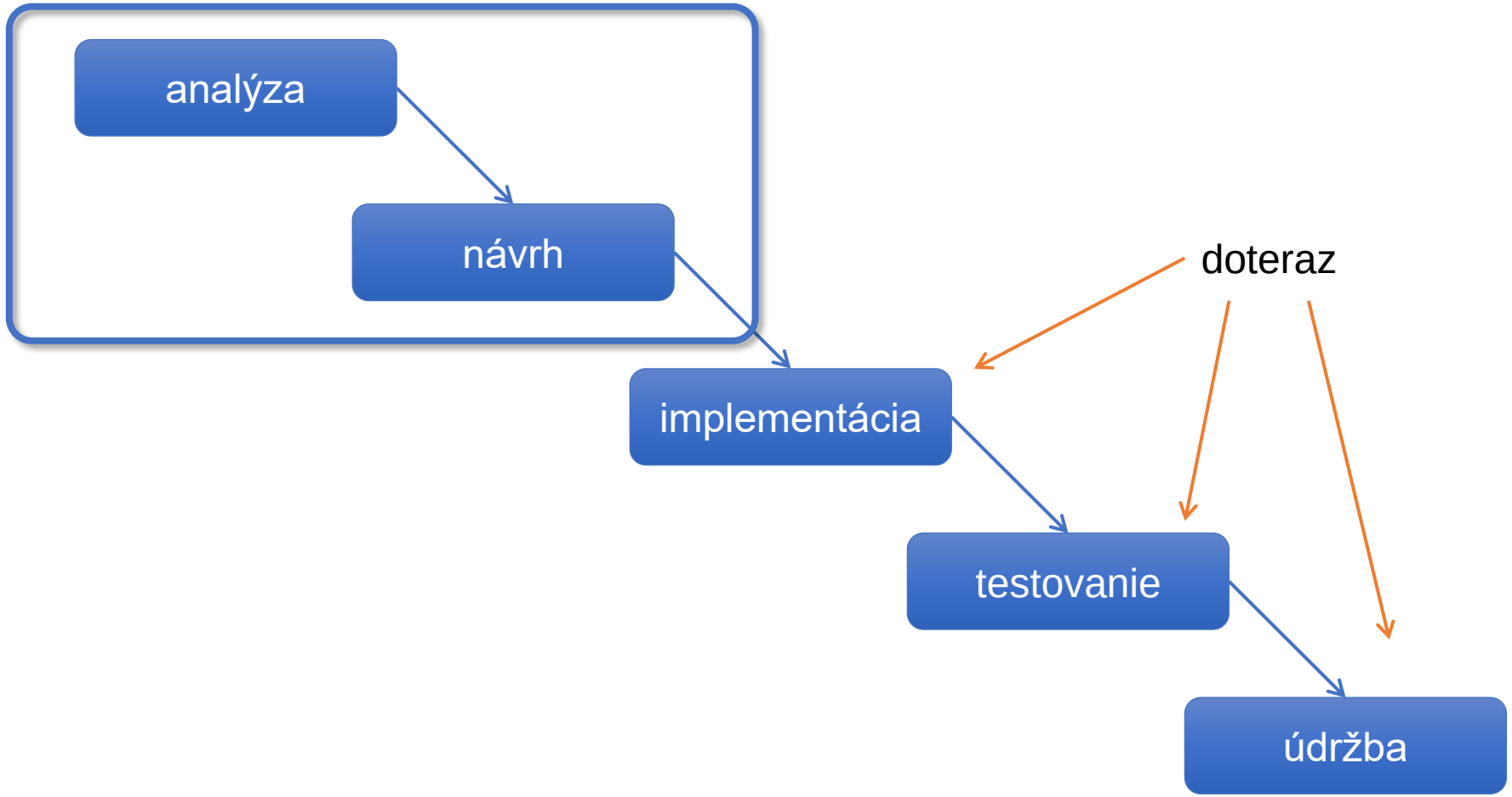
Tvorba softvéru

- projekty doteraz – hotová definícia tried
- ako začať s definíciou tried?
- celý softvér je realizovaný triedami

- životný cyklus softvéru
  - všetky činnosti tvorby a použitia
  - začiatok (vznik) – rozhodnutie riešiť problém
  - koniec (smrť) – softvér sa prestane používať

  - niektorý softvér umiera bez použitia

Vodopádový model



Jednoduché prostriedky analýzy

- cieľ analýzy – vyhľadať a navrhnúť triedy jadra

- metóda verb&noun
- karty CRC

Metóda verb&noun

- rozbor textového zadania

- podstatné mená – kandidáti na triedy a atribúty
- slovesá – kandidáti na operácie (správy, metódy)

- odstránenie synoným
- odstránenie nepotrebných podstatných mien
  - opatrne, neponáhlať

Príklad

Taxislužba používa taxíky a mikrobusy. Taxíky prepravujú jednotlivcov alebo skupiny ľudí z jedného miesta nástupu do jedného cieľa. Mikrobusy sa používajú na zbieranie ľudí na rôznych miestach nástupu a rozvoz do rôznych cieľov. Taxislužba prijíma telefonické objednávky od jednotlivcov, hotelov a cestovných kancelárií. Šofér taxíka i mikrobusu oznamuje do centrály príchod do miesta nástupu aj do cieľa cestujúceho. Centrála neudržiava žiadny systém čakajúcich, objednávky prijíma len ak má voľné vozidlo.

Firma chce zistiť, či má rozšíriť vozový park, lebo rozširuje oblasť, v ktorej poskytuje služby. Softvér by mal dať odpoveď na otázky koľko zákazníkov museli odmietnuť, aký čas strávia vozidlá jazdou bez pasažierov, koľko času stoja nevyužité.

Príklad – verbs

Taxislužba používa taxíky a mikrobusy. Taxíky prepravujú jednotlivcov alebo skupiny ľudí z jedného miesta nástupu do jedného cieľa. Mikrobusy sa používajú na zbieranie ľudí na rôznych miestach nástupu a rozvoz do rôznych cieľov. Taxislužba prijíma telefonické objednávky od jednotlivcov, hotelov a cestovných kancelárií. Šofér taxíka i mikrobusu oznamuje do centrály príchod do miesta nástupu aj do cieľa cestujúceho. Centrála neudržiava žiadny systém čakajúcich, objednávky prijíma len ak má voľné vozidlo.

Firma chce zistiť, či má rozšíriť vozový park, lebo rozširuje oblasť, v ktorej poskytuje služby. Softvér by mal dať odpoveď na otázky koľko zákazníkov museli odmietnuť, aký čas strávia vozidlá jazdou bez pasažierov, koľko času stoja nevyužité.

Kandidáti na triedy

- podstatné mená v jednotnom čísle

- taxislužba, taxík, mikrobus, jednotlivec, skupina, miesto nástupu, cieľ, objednávka, hotel, cestovná kancelária, šofér, cestujúci, centrála, vozidlo, firma, vozový park, oblasť, služba, odpoveď, otázka, softvér, zákazník, pasažier

Odstránenie synoným

- taxislužba, centrála, firma – Firma
- jednotlivec, zákazník, pasažier, cestujúci – Cestujúci
- skupina – vyjadruje počet cestujúcich (nepoužijeme)
- Taxík,
- Mikrobus,
- Vozidlo – taxík aj mikrobus, spoločné veci
- ...

Príklad – nouns

Taxislužba používa taxíky a mikrobusy. Taxíky prepravujú jednotlivcov alebo skupiny ľudí z jedného miesta nástupu do jedného cieľa. Mikrobusy sa používajú na zbieranie ľudí na rôznych miestach nástupu a rozvoz do rôznych cieľov. Taxislužba prijíma telefonické objednávky od jednotlivcov, hotelov a cestovných kancelárií. Šofér taxíka i mikrobusu oznamuje do centrály príchod do miesta nástupu aj do cieľa cestujúceho. Centrála neudržiava žiadny systém čakajúcich, objednávky prijíma len ak má voľné vozidlo.

Firma chce zistiť, či má rozšíriť vozový park, lebo rozširuje oblasť, v ktorej poskytuje služby. Softvér by mal dať odpoveď na otázky koľko zákazníkov museli odmietnuť, aký čas strávia vozidlá jazdou bez pasažierov, koľko času stoja nevyužité.

Operácie

- firma
  - riadi taxíky a mikrobusy
  - prijíma objednávky
  - plánuje jazdy vozidiel
- cestujúci
- poloha
- zdroj cestujúcich
  - žiada firmu o službu pre cestujúceho
- ...

Metóda kariet CRC

- CRC: class – responsibility – colaborators
- fyzické karty – formát A6

Názov triedy	
zodpovednosť	spolupráca

Vytváranie CRC kariet

- spísanie scenárov
  - popis konkrétnej operácie v programe
- „vykonávanie“ scenárov
  - vytváranie CRC kariet pre neexistujúce triedy
  - zápis spolupracujúcich tried
  - zápis zodpovedností (operácií, vlastností objektu)

Scenár – preprava cestujúceho

- zdroj cestujúcich vytvorí nového cestujúceho
- zdroj cestujúcich vygeneruje polohu a cieľ cestujúceho
- cestujúci príjme polohu a cieľ
- zdroj cestujúcich požiada firmu o prepravu cestujúceho

Scenár do CRC

- zdroj cestujúcich vytvorí nového cestujúceho

ZdrojCestujucich	
vytvori cestujuceho	Cestujuci

Cestujuci	

Vysledok analýzy

- pri V&N aj CRC
- vytvorenie modelu v UML
- došpecifikovanie
  - chýbajúce triedy/operácie (hlavne V&N)
  - doplnenie polymorfizmu (dedičnosť/interface)
- vytvorenie zdrojového kódu tried
  - prázdne telá metód a konštruktorov
  - niektoré nástroje umožňujú vygenerovať
- a ďalej to už poznáte ☺

Iný prístup – TDD

- TDD – Test Driven Development
- testom riadený vývoj

- testy sa vymyslia a napíšu skôr ako sa začne s vývojom funkcionality
- v každej fáze vývoja sa dá jednoducho skontrolovať funkčnosť programu

- Kent Beck: Programování řízené testy, Grada,
  ISBN 80-247-0901-5

Postup práce

1. pridaj test pre scenár
2. spusti test – pravdepodobne zlyhá
3. vytvor implementáciu
   a) implementuj prázdne metódy – aby bolo možné test preložiť
   b) implementuj telá metód – aby test prešiel
4. spusti testy (regresné testovanie)
5. vykonaj refaktoring
6. opakuj od bodu 1 pre nový scenár

TDD – výhody

- efektívnosť
  - rýchlejší vývoj
- kvalita programu
  - v každom okamihu si vieme skontrolovať funkčnosť scenárov
  - refaktoring
  - program obsahuje len otestovanú funkcionalitu
- lepší návrh?
  - najskôr navrhujeme rozhranie (pri tvorbe testu)
- komunikácia so zákazníkom
  - môže priamo odsúhlasovať scenáre (testy)

Špeciálna podpora pre TDD (Cucumber)

```
Feature: Search Courses
      In order to ensure better utilization of courses
      Potential students should be able to search for courses

      Scenario: Search by topic
            Given there are 240 courses which do not have
                  the topic "biology"
            And there are 2 courses A001, B205 that each have
                  "biology" as one of the topics
            When I search for "biology"
            Then I should see the following courses:
            | Course code   |
            | A001          |
            | B205          |
```

Návrh tried

- pri CRC aj TDD nie je niekedy jasné zaradenie funkcionality do tried
- RDD – Responsibility Driven Desing
  - zodpovednosťou riadený návrh

Zodpovednosťou riadený návrh (1)

- objekt má len jednu úlohu
- objekt zodpovedá za svoj stav – dáta
- objekt zodpovedá za zmeny stavu – operácie s dátami
- informácie o svojom stave poskytuje objekt
- atribúty tvoria logický celok
- metódy implementujú jednu operáciu

Zodpovednosťou riadený návrh (2)

- do ktorej triedy pridať novú metódu?

  =

- čie dáta bude metóda spracovávať?

- každá trieda je zodpovedná za operácie so svojimi vlastnými dátami

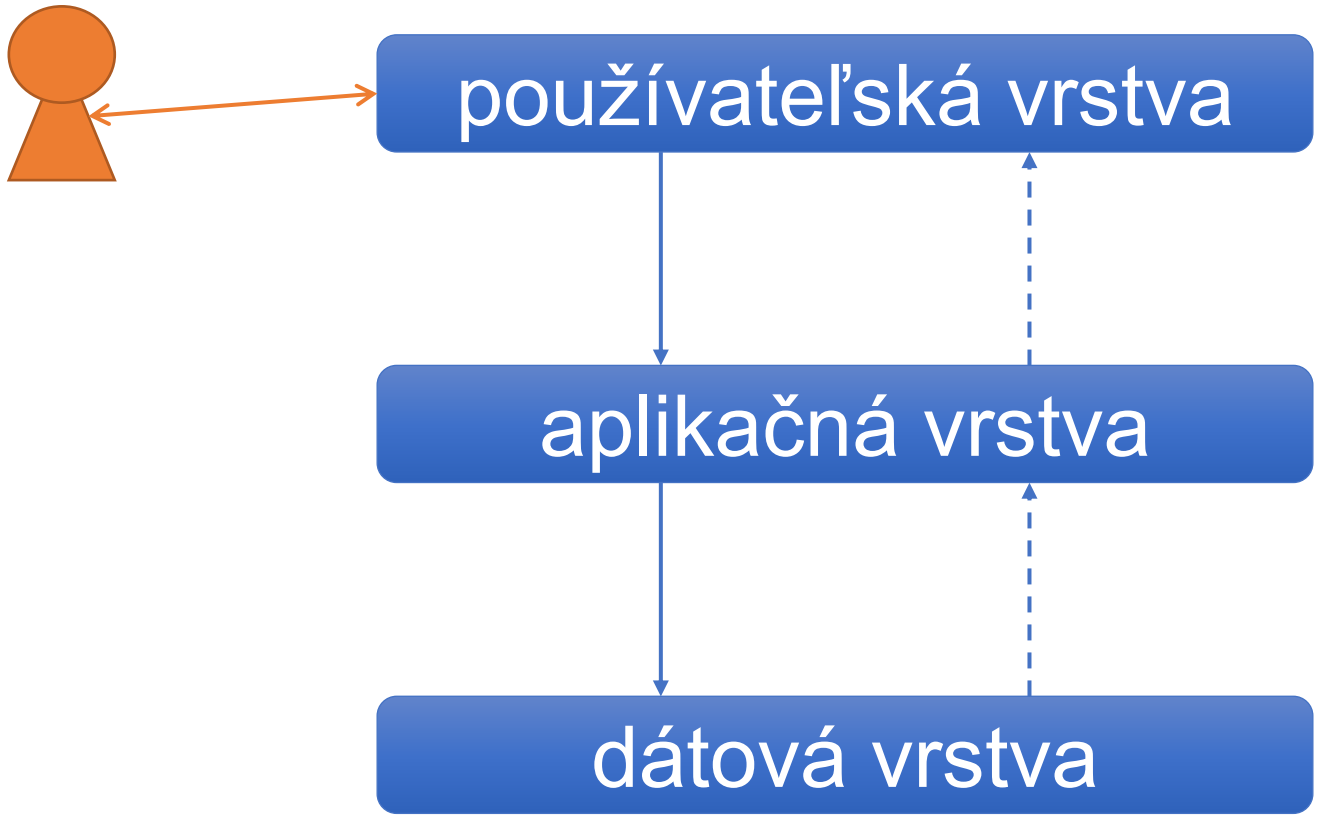
Trojvrstvový model aplikácie (1)

- tri skupiny – vrstvy – objektov aplikácie

- aplikačná vrstva – objekty logiky aplikácie
- používateľská vrstva (GUI, TUI) – komunikácia človek – aplikácia, preberanie vstupných údajov, prezentácia výsledkov
- dátová vrstva – uloženie dát na opätovné použitie

- komunikujú len cez rozhrania

Trojvrstvový model aplikácie (2)



Používateľská vrstva

- resp. prezentačná/servisná vrstva
- realizuje tzv. prezentačnú logiku
  - objekty používateľského prostredia
  - spracúva používateľské vstupy
  - aktualizuje používateľské prostredie podľa výsledkov operácií
- GUI/TUI/CLI/WebUI...
- nevykonáva žiadne logické operácie

Aplikačná vrstva

- resp. logická/biznis vrstva
- obsahuje logiku aplikácie
- väčšinou najzložitejšia časť objektovej štruktúry
- nekomunikuje s používateľom

- nezávislá na použitej technológii prezentačnej vrstvy

Dátová vrstva

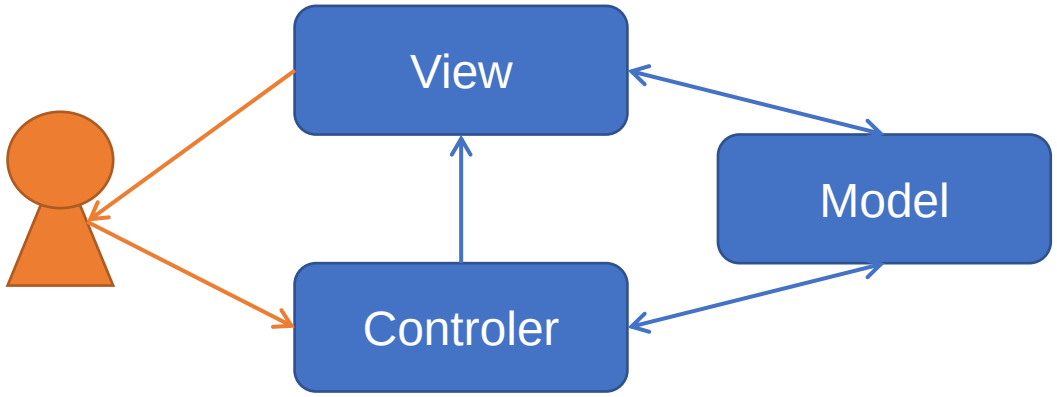
- resp. perzistenčná vrstva
- ukladá dáta (súbory, databáza, ...)

MVC

- MVC – Model View Controler
- vhodné iba pre GUI

- tri vrstvy
  - Model – predstavuje logickú a dátovú časť aplikácie
  - View – zobrazuje výsledky používateľovi
  - Controler – reaguje na vstupy od používateľa

Architektonický vzor MVC



Vrstva Model

- realizuje zväčša dátovú časť aplikácie
- obsahuje časť aplikačnej logiky
- každú zmenu oznamuje vrstve View
  - návrhový vzor Observer – samoštúdium

Vrstva View

- realizuje zobrazenie informácií používateľovi
- vo SWINGu – JFrame a grafické komponenty
- prijíma oznamy o zmene vo vrstve Model
  - aktualizuje obsah okna

Vrstva Controler

- reakcia na používateľove akcie
- obsahuje časť aplikačnej logiky
- vo SWINGu – poslucháči (Listener)
- upravuje model (posiela správy na zmeny stavu)
- môže vymeniť view (zmení aktuálne okno)

MVC

- moderný architektonický vzor
- základ väčšiny dnes používaných
  - MVP
  - MVVM
  - MVC model 2

Princíp KISS

- Keep It Simple, Stupid
- (nekomplikuj to, pako)

- ak existuje viac možností na riešenie – použiť najjednoduchšie
- ťažko povedať, čo je najjednoduchšie ☺

Príklad KISS (1)

```
public String nazovDna(int den) {
    switch (den) {
        case 1: return "Pondelok";
        case 2: return "Utorok";
        case 3: return "Streda";
        case 4: return "Stvrtok";
        case 5: return "Piatok";
        case 6: return "Sobota";
        case 7: return "Nedela";
        default: throw new IllegalArgumentException();
    }
} // zdroj: http://principles-wiki.net/principles:keep_it_simple_stupid
```

Príklad KISS (2)

```
private static final String[] DNI = {
        "Pondelok", "Utorok", "Streda",
        "Stvrtok", "Piatok", "Sobota", "Nedela"};

public String nazovDna(int den) {
    if ((den < 1) || (den > 7)) {
        throw new IllegalArgumentException();
    }

    return Calendar.DNI[den - 1];
} // zdroj: http://principles-wiki.net/principles:keep_it_simple_stupid
```

KISS v Zen of Python

- Simple is better than complex.
- Complex is better than complicated.

- Jednoduché je lepšie, ako komplexné
- Komplexné je lepšie, ako komplikované

Princíp DRY

- Don’t Repeat Yourself
- (neopakuj sa)

- opak princípu WET (Waste Everyone’s Time, Write Everything Twice)

- za každú informáciu je zodpovedná jedna konkrétna časť kódu

Porušenie DRY

- duplicity v kóde
- literály miesto pomenovaných konštánt
- literály miesto hodnôt odvodených od konštánt

- dotýka sa aj komentárov, dokumentácie

Dodržiavanie DRY

- zjednodušuje/komplikuje úpravy programu
- zvyšuje/znižuje implementačnú závislosť

Príklad WET

```
private int[][] sachovnica;
...
this.sachovnica = new int[8][8];
...
for (var y = 0; y < 8; y++) {
    for (var x = 0; x < 8; x++) {
        this.sachovnica[y][x] = 5;
    }
}
...
System.out.println("Sachovnica ma 64 policok");
```

Príklad DRY

```
private int[][] sachovnica;
private final static SIRKA = 8;
private final static VYSKA = 8;
private final static POCET_POLICOK = SIRKA*VYSKA;
```

SOLID princíp

- tzv. tvrdý princíp

- zoskupenie 5 princípov v OOP

Solid

- SRP – Single Responsibility Principle
- princíp jednej zodpovednosti

- každá jedna trieda má mať len jednu dobre definovanú zodpovednosť

- súdržnosť – Cohesion

sOlid

- Open-Closed Principle

- každá trieda má byť otvorená pre rozširovanie, uzavretá pre modifikáciu

- dve samostatné vetvy:
  - OCP pri úpravách zdrojového kódu
  - OCP pri dedičnosti

OCP pri úpravách kódu

- návrh musí myslieť „dopredu“
- programátor nepotrebuje upravovať existujúcu funkcionalitu
- programátor má možnosť rozširovať funkcionalitu triedy

OCP pri dedičnosti

- potomok neupravuje funkcionalitu triedy
  - neprekrýva metódy s konkrétnym kódom
- možnosť rozširovať, pridávať funkčnosť
  - prekrývanie prázdnych metód
  - implementácia abstraktných metód
  - pridávanie nových metód/konštruktorov/atribútov

soLid

- The Liskov Substitution Principle

kuk ☺

solId

- Interface Segregation Principle
- princíp delenia rozhrania
- nevytvárame jeden veľký interface
- znižovanie implementačnej závislosti
  - triedy závisia len na tom, čo potrebujú

Príklad ISP

- interface Predmet
  - getNazov
  - getPopis
  - pouzi

vs:

- interface Predmet
- interface Nazvany
  - getNazov
- interface Popisany
  - getPopis
- interface Pouzitelny
  - pouzi

ISP – extrémy

- jeden interface
  - bez využitia ISP
  - triedy závisia na celom rozhraní
- jeden interface pre každú správu
  - často nezmysel

- rozumnejšie: jeden interface pre súvisiacu skupinu správ

soliD

- Dependency Inversion Principle

- závislosť na interface, miesto závislosť na konkrétnej triede
- trieda nevytvára inštancie, dostáva ich ako parametre (metód, konštruktorov)

Zhrnutie

- existuje veľa princípov, ktorých sa možno držať
- všetko s mierou

- OOP je filozofia

- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
  - The Zen of Python

Informatika 2

Ukončenie predmetu

doc. Ing. Ján Janech, PhD.
14. 5. 2023

ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Odovzdávanie semestrálnych prác

- T: nedeľa 19. 5. 2024 polnoc
- nutné obhájiť – termín určuje cvičiaci
- zadanie na moodli, odovzdať je nutné:
  - všetko v jednom ZIPe
  - dokumentácia v PDF (word, open office, LaTeX, wordpad – to je jedno)
  - UML diagram v PNG (nemusí byť UML .FRI – nie generovaný!!!)
  - zdrojové kódy
    - splnený checkstyle
    - dokumentačné komentáre
    - projekt v Intellij Idea, alebo podobnom IDE
    - ak je projekt veľký, nemusia byť obrázky/videá/zvuky
- kuk pravidlá na moodli

Hodnotenie semestrálnych prác (1)

Položka hodnotenia	Max. pocet bodov
Rozsah semestrálnej práce	max 6
Navrhnuté algoritmy	max 7
Navrhnutá objektová štruktúra	max 7
Dodržanie princípov objektového programovania	povinné
Splnenie checkstyle	povinné
Dokumentacné komentáre	povinné
Dokumentácia	povinné
UML diagram	povinné
Spolu	**max 20 (min 5)**

Hodnotenie semestrálnych prác (2)

- hodnotíme iba vlastnú prácu
- hodnotíme iba to, čo bolo náplňou predmetu
  - (za niečo navyše môže byť drobné plus)
- dobrá semestrálna práca je za 10 bodov
  - slabá menej
  - skvelá viac

Polymorfizmus v semestrálnych prácach

- povinné správne využitie polymorfizmu
  - nie je povinná dedičnosť
  - nie sú povinné interface
- polymorfizmus musí byť implementovaný správne
  - rozdiel v algoritmoch
  - polymorfizmus má program zjednodušiť/sprehľadniť

Cudzí kód v semestrálnych prácach (1)

- je povolený
- cudzí kód:
  - stiahnuté (aj modifikované) z internetu
  - poradené od kamaráta
  - skopírované od kamaráta
  - odkukané z tutoriálu
  - ...

Cudzí kód v semestrálnych prácach (2)

- nutné označiť
  - priamo na mieste v komentári (ak sa dá)
  - v textovej dokumentácii
  - uviesť presný zdroj (odkaz, meno autora, atp...)
- pozor na autorský zákon
- nehodnotí sa, ale musíte mu rozumieť
- neoznačený cudzí kód je považovaný za plagiát – podanie na disciplinárnu komisiu

Prihlásenie na skúšku

- min. 25 bodov zo semestra, min. 5 bodov zo semestrálnej práce
- prihlasovanie cez systém vzdelávanie
  - 5 a viac možností v rovnakom čase – rôzne miestnosti
- jeden riadny a dva opravné termíny – podľa študijného poriadku
- ak nemôžete prísť a už sa nedá odhlásiť, píšte mne
  - spätne ale termíny neruším

- !!! neobsadzujte termín ak nepotrebujete

Priebeh skúšky (1)

- prísť načas
- max 60min úvod
  - prezencia
  - technický úvod
  - rozdanie zadaní na papieri
  - vysvetlenie zadania
  - otázky a odpovede
- zadanie
  - UML
  - textový popis
  - rozpis bodov po metódach/triedach
  - okolo 150 riadkov kódu

Priebeh skúšky (2)

- vypracovanie 120min
- nepodvádzať!!!
  - Proctoring systém
  - podanie na DK
- povolené pomôcky
  - čistý papier a pero – ukázať skúšajúcemu
  - hlúpa kalkulačka (obyčajná/vedecká – žiadna programovateľná, žiadna smart, žiadna v mobile, ...) alebo Windows kalkulačka
  - Javadoc štandardnej knižnice – naučiť sa otvárať v BlueJ

Informatika 3

- prerekvizita: Informatika 1
- programovací jazyk C++
- iný tím vyučujúcich