

Fakulta informatiky a informačných technológií
Slovenská technická univerzita

Objektovo – orientované programovanie

Účtovnícka firma

Michaela Hanková

Zámer projektu

Vytvoriť zamestnancov účtovníckej firmy, ktorí vedia vypočítať svojim zákazníkom poistné (zdravotné aj sociálne), výsledok hospodárenia (z ich hrubého zisku odpočítať poistné) a daň. Pri počítaní dane z príjmu s. r. o. a akciovej spoločnosti spočíta ich náklady, spočíta výnosy a potom odpočíta náklady od výnosov.

Ak je výsledok väčší ako 100 000, obchodná spoločnosť zaplatí daň vo výške 21% zo zisku, ak sú výnosy vyššie ako náklady, ale rozdiel medzi nimi je menej ako 100 000, spoločnosť zaplatí daň vo výške 15% zo zisku. Ak sú náklady vyššie ako výnosy, spoločnosť neplatí žiadnu daň.

Firma má jedného riaditeľa, ktorý je vytvorený prostredníctvom návrhového vzoru singleton.

Účtovnícka firma má majetok – autá, stoličky, vo svojej budove vie organizovať v niektorej zo svojich dvoch sál oslavy, svadby, a za tieto akcie dostáva zaplatené na základe počtu ľudí, ktorí sa akcie zúčastnili.

V maine sa najprv vytvoria zamestnanci v poradí: účtovník a junior účtovník (brigádnik). Obidvaja sa pridajú do arrayListu zamestnancov. Ďalej sa vytvorí akciová spoločnosť, zadajú sa jej údaje a vypočíta sa jej daň z príjmu. Ďalej sa vytvorí spoločnosť s ručením obmedzeným, zadajú sa jej údaje a vypočíta sa jej daň z príjmu. Ako posledný sa vytvorí živnostník, zadajú sa jeho údaje a vypočíta sa mu daň z príjmu. Hneď po tom sa zmení priemerný mesačný príjem živnostníka, a do konzoly sa vypíše: „Živnostník má zmenený priemerný mesačný príjem, nový príjem je:...“. Tento výpis spôsobuje účtovník, pretože sleduje zmenu mesačného príjmu Živnostníka.

Po tomto sa prostredníctvom RTTI zisťuje počet účtovníkov a počet junior účtovníkov. V tomto prípade mám jedného účtovníka a jedného juniora. Ďalej si junior ráta svoje poistné a svoju výplatu.

Vytvorí sa objekt auta a motora. Motor je vlniezenou triedou auta.

Ako posledné sa uložia do súboru údaje o zamestnancoch pomocou serializácie. Pri deserializácii sa ukladajú údaje zo súboru do nového arrayListu.

Ako posledné sa vytvorí GUI okno, prostredníctvom ktorého môžem vytvoriť nového zákazníka – buď nového živnostníka, alebo s. r. o.

V priečinku javadoc sa nachádza javadoc môjho projektu.

Hlavné kritériá:

- nachádzajú sa tu 2 hierarchie dedenia a každá z nich je troj-úrovňová
- agregácia sa nachádza v triede Zamestnanec a účtovník
- kompozícia sa nachádza v triede Budova (obsahuje hornú a dolnú sálu, tieto 2 sály bez budovy nemôžu existovať)
- zapuzdrenie

Ďalšie kritériá:

Použitie návrhových vzorov:

Observer – použitý v triede Účtovník – účtovník je informovaný, keď si živnostník zmení svoj mesačný príjem.

Factory – používa sa na vytvorenie stoličky, ktorú používajú účtovníci. Tento návrhový vzor vytvorí stoličku bez toho, aby sa musela v maine zadávať jej cena, pretože cena sa stoličke priradí už v triede StoličkaFactory, a táto trieda nám vráti už vytvorený objekt, ktorý má nastavené všetko potrebné.

Builder – používa sa na vytvorenie počítača, ak mu nechceme zadať všetky jeho premenné (kvalita, nákupná cena, značka a operačný systém). Niekedy nám stačí vytvoriť počítač, ktorý má nastavené premenné iba nákupná cena alebo značka počítača.

Vlastná výnimka: je definovaná v triede ZápornýVstup, ktorá kontroluje, či pri zadávaní základného imania nezadávam zápornú hodnotu. Premenná základné imanie sa je definovaná v triede Sro. Ak zadávam zápornú hodnotu, vypíše sa chybová hláška: Základné imanie nemôže byť menšie ako 0, v tomto prípade: (nasleduje zadané záporné číslo).

Grafické používateľské rozhranie: ak je zákazníkom účtovníckej firmy živnostník alebo spoločnosť s ručením obmedzeným, môžu svoje údaje zapísať do grafického používateľského rozhrania. Aby mohli zistiť výšku dane, ktorú majú zaplatiť, musia najprv do GUI napísať typ svojho podnikania, akceptované sú len slová Živnost, živnost, Sro, sro. Ak sa im podarí prihlásiť, zmení sa okno. Do nového okna napíšu svoje meno, ak je zákazníkom živnostník, alebo názov svojej firmy, ak je zákazníkom s. r. o. Živnostník vypíše svoje meno, počet mesiacov, počas ktorých prevádzkoval svoju živnosť a jeho priemerný mesačný príjem. GUI mu v nasledujúcom okne ukáže vypočítané sociálne, zdravotné poistenie, jeho celkový ročný hrubý príjem a vypočítanú daň, ktorú má zaplatiť.

Ak údaje do okna vkladá Sro, musí vyplniť riadky:

Názov spoločnosti, počet zamestnancov, základné imanie, prijaté úroky, zaplatené úroky, kurzový zisk, kurzová strata, hospodárske výnosy a výdavky na mzdu.

Po zadaní údajov sa v ďalšom okne objavia vypočítané údaje: počet zamestnancov, zdravotné a sociálne poistenie zamestnancov, finančné náklady a finančné výnosy, hospodárske náklady a výnosy a výsledný daň.

explicitné použitie RTTI (riadok 220): Používa sa v maine, na zistenie presného počtu brigádnikov a zamestnancov na plný úväzok. Má arrayList, ktorý celý vo for cykle prejde, v ktorom sú uložení všetci zamestnanci a do premennej početÚčtovníkov ukladá počet zamestnancov na plný úväzok a do premennej početJuniorÚčtovníkov ukladá počet brigádnikov. Používa pri tom instanceof.

Vhniezdená trieda (riadok 239): Vytvorí sa objekt auta. Trieda auta obsahuje v sebe ešte ďalšiu triedu – motor. Trieda motor je vhniezdená. Po vytvorení objektu auta sa vytvorí motor, ktorý patrí autu, ktoré sa práve vytvorilo. V mojom prípade, sa vytvorilo auto Volvo, a potom sa vytvoril k nemu motor, ktorý má rok výroby 2018, nie je pokazený a má výkon 80 kiloWattov.

Implicitná implementácia metód: nachádza sa v interface `MetódyZamestnancov` (definuje metódu `vypočítajPoistné`, čo znamená, že každý zamestnanec si vie vypočítať výšku svojho poistného zo mzdy) a `ZákazníkInterface` (definuje metódu `zaplaťDaň`, ktorá má ako argumenty sumu, ktorú treba zaplatiť, a bankový účet, z ktorého odídu peniaze. Tento interface používajú subjekty, ktoré podnikajú, preto musia vedieť zaplatiť daň z príjmu).

Serializácia: Nachádza sa v maine, keď zapisujem do súboru `udajeZamestnancov.txt` všetky údaje o vytvorených zamestnancoch. Všetci zamestnanci sú uložení v `ArrayListe`. Druhý `ArrayList` používam na deserializáciu, teda na načítanie údajov zo súboru `udajeZamestnancov.txt`, aby som skontrolovala, či mi do súboru dobre uložilo údaje.

Zmeny v gitHub classroom:

1. commit - pridali sa základné triedy, plus zámer projektu – dokument.
2. V druhom trieda Sro zmenila svojho rodiča, z triedy `Živnostník` na triedu `Zákazník`
3. pridanie readme dokumentu do GitHubu
4. upravenie triedy `auto`
5. pridanie interface triede `auto`
6. pridanie abstraktnej metódy v triede `Zamestnanec` – metóda `Zobraz údaje`, ktorá vypíše základné údaje o zamestnancovi, ktorý volá túto metódu
7. pridanie triedy `budova`, a kompozície, pretože obsahuje dve premenné, ktoré bez vytvorenej triedy `budova` nemôžu existovať – horná a dolná sála
8. pridanie troch návrhových vzorov. Z týchto troch vzorov v projekte zostali dva – `builder` a `factory`
9. vyrátanie zisku prvého živnostníka – v maine sa vytvorili zamestnanci účtovníckej firmy a živnostník. Po zadaní základných údajov sa vypočítala jeho daň z príjmu
10. vypočítanie dane z príjmu Sro – v maine sa vytvoril objekt Sro s názvom predajňa bicyklov. Po zadaní potrebných údajov sa jej vypočítala daň z príjmu
11. vymazanie premenných, ktoré nie sú potrebné
12. pridanie RTTI na zistenie počtu zamestnancov na plný úväzok a počtu brigádnikov
13. vytvorenie vhniezdennej triedy `Motor` do triedy `Auto`
14. pridanie serializácie na uloženie údajov o zamestnancoch do súboru
15. spravené kompletne GUI pre živnostníka – ak chceme vypočítať daň nejakému živnostníkovi, nemusíme to robiť ručne cez main, ale stačí potrebné údaje napísať do GUI okna
16. vytvorenie javadocu, ktorý ešte nie je kompletný
17. spravené kompletne GUI pre s. r. o.


Od prezentácie sa zmenilo:


- sú 3 nové návrhové vzory (observer, builder a factory)
- pridané RTTI
- pridaná serializácia
- vyrobené GUI
- pridaný Javadoc
- pridané interface: `MetodyZamestnancov` a `ZakaznikInterface`


Popis class diagramu:


Nachádzajú sa v ňom 2 hierarchie dedenia, ktorú sú nosnou časťou celého projektu. Jedna hierarchia je zameraná na zamestnancov účtovníckej firmy, druhá na jej zákazníkov. Z triedy `zamestnanec` dedí `JuniorUctovnik`, a od tohto `Juniora` dedí `Uctovnik`.


Z triedy `zákazník` dedí `Sro`, a od tejto `Sro` dedí `AkciováSpoločnosť` (má navyše rozdelenie zisku akcionárom a odvod časti zisku na sporiaci bankový účet).


<<Java Class>>	
 Zakaznik	zakaznik
✧ nazov: String	
✧ adresa: Adresa	
✧ plataciUcet: BankovyUcet	
✎ Zakaznik()	
● getNazov():String	
● getAdresa():Adresa	
● setNazov(String):void	
● setAdresa(Adresa):void	
✎ vypisNazov():void	

<<Java Class>>	
 AkciovaSpolocnost	zakaznik
▣ pocetAkcionarov: int	
▣ dividendy: double	
▣ cistyZisk: double	
▣ ziskNaRozdelenie: double	
▲ sporaciUcet: BankovyUcet	
✎ AkciováSpoločnosť()	
● setSporaciUcet(BankovyUcet):void	
● setPocetAkcionarov(int):void	
● setCistyZisk(double):void	
● getPocetAkcionarov():int	
● getSporaciUcet():BankovyUcet	
● getDanZPrjmu():double	
● getCistyZisk():double	
● getVysledokHospodarenia():double	
● vypisNazov():void	
● odoberCasizSkutuUcet(BankovyUcet):void	
● vypocitatDivendu(double):double	

<<Java Class>>	
 Sro	zakaznik
▲ pocetZamestnancov: int	
▲ vydavkyNaMzdu: double	
▲ zdravotneZamestnancov: double	
▲ socialneZamestnancov: double	
▲ financneNaklady: double	
▲ financneVynosy: double	
▲ kurzovyZisk: double	
▲ kurzovaStrata: double	
▲ prijateUroky: double	
▲ zaplateneUroky: double	
▲ trzbyZPredajtovaru: double	
▲ nakladyNaMzdy: double	
▲ hospodarskeNaklady: double	
▲ hospodarskeVynosy: double	
▲ zakladatelmanie: double	
▲ vysledokHospodarenia: double	
▲ danZPrjmu: double	
✎ Sro()	
● getPocetZamestnancov():int	
● getVydavkyNaMzdu():double	
● getDanZPrjmu():double	
● getKurzovyZisk():double	
● getKurzovaStrata():double	
● getPrijateUroky():double	
● getZaplateneUroky():double	
● getFinancneNaklady():double	
● getFinancneVynosy():double	
● getHospodarskeNaklady():double	
● getHospodarskeVynosy():double	
● getZdravotne():double	
● getSocialne():double	
● getPrijate(BankovyUcet):BankovyUcet	
● getZakladatelmanie():double	
● setPocetZamestnancov(int):void	
● setFinancneNaklady(double):void	
● setFinancneVynosy(double):void	
● setHospodarskeNaklady(double):void	
● setHospodarskeVynosy(double):void	
● setVydavkyNaMzdu(double):void	
● setPrijateUroky(double):void	
● setZaplateneUroky(double):void	
● setVysledokHospodarenia(double):void	
● setZdravotne(double):void	
● setSocialne(double):void	
● setKurzovyZisk(double):void	
● setKurzovaStrata(double):void	
● setPrijate(BankovyUcet)(BankovyUcet):void	
● setZakladatelmanie(double):void	
● setDanZPrjmu(double):void	
● vypisNazov():void	
● zaplatDan(double, BankovyUcet):void	

<<Java Class>>	
 JuniorUctovnik	zamestnanec
▣ pocitac: Pocitac	
✎ hodnotaMzda: double	
▣ odpracovaneHodiny: double	
▣ mzda: double	
▣ vyplata: double	
✎ dz: DecimalFormat	
✎ JuniorUctovnik()	
● getOdpracovaneHodiny():double	
● getMzda():double	
● getPocitac():Pocitac	
● setOdpracovaneHodiny(int):void	
● setVyplata(double):void	
● selfPocitac(Pocitac):void	
● selfVyplata(double):void	
● selfPoistne():double	
● getVyplata():double	
● povetStojleMeno():void	
● menoZamestnanca():void	
● vypocitatZdravotnePoistenieZmnoznika(double):double	
● vypocitatSocialnePoistenieZamestnanca(int, double):double	
● vypocitatZdravotnePoistenieZamestnanca(int, double):double	
● vypocitatSocialnePoistenieZamestnanca(int, double):double	
● vypocitatNakladyZHospodarskejCinnosti(double, double, double):double	
● vypocitatVynosyZFinananejCinnosti(double, double):double	
● vypocitatVHPreDdanou(double, double, double, double):double	
● zobrazaUdaje():void	

<<Java Class>>	
 Zamestnanec	zamestnanec
✧ meno: String	
▣ vek: int	
▣ poistne: double	
▣ stolicika: Stolicika	
✎ Zamestnanec()	
● getMeno():String	
● getVek():int	
● setMeno(String):void	
● setVek(int):void	
✎ zobrazUdaje():void	

<<Java Class>>	
 Uctovnik	zamestnanec
✎ dennaMzda: double	
▣ hrubaMzda: double	
▣ vyplata: double	
▣ poistne: double	
▣ odpracovaneDni: int	
▣ adresa: Adresa	
▣ pocetDniDovolenky: int	
▣ pocitac: Pocitac	
▲ vysledokHospodarenia: double	
✎ Uctovnik()	
● getOdpracovaneDni():int	
● getHrubuMzdu():double	
● vypocitatPoistne():double	
● getVyplata():double	
● getStolicika():Stolicika	
● getPocetDniDovolenky():double	
● setPocetDniDovolenky(int):void	
● setStolicika(Stolicika):void	
● getAdresa():Adresa	
● setAdresa(Adresa):void	
● setOdpracovaneDni(int):void	
● getOdpracovaneHodiny():double	
✎ vypocitatVHZmnoznika(double, double, double):double	
✎ vypocitatDanObchodnejsPolocnosti(double, double, double, double):double	
✎ vypocitatDanZmnoznika(double):double	
● menoZamestnanca():void	
● update(Observable, Object):void	