Meno: Daniel Cok

Názov projektu: Hraničná kontrola!

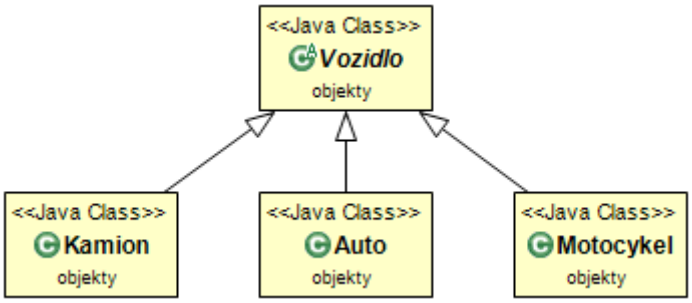
Zámer projektu:

V programe Hraničná kontrola si bude používateľ môcť podať žiadosť o registráciu vozidla dovezeného zo zahraničia na Slovensko práve na hranici. Od posledného zadania som sa zameral na dopravné prostriedky ,čím so zúžil pojem tovar ako taký. Žiadateľ zadá pri tom svoje osobné údaje(meno, priezvisko, pohlavie, vek, číslo občianskeho preukazu,...), a potom údaje o vozidle(typ vozidla, značka, rok výroby, VIN, výkon, najazdené kilometre, krajina pôvodu, údaje o emisiách...), ktoré chce zaregistrovať. Úradník na druhej strane žiadosť zamietne alebo potvrdí, a následne na základe daných údajov o program vypočíta sumu, ktorú má žiadateľ zaplatiť - clo. Cena za registráciu bude závisieť od typu vozidla(kamión, osobný automobil, motocykel,...) a od krajiny odkiaľ bolo dovezené(dve sadzby – jedna pre vozidlá dovezené z krajín Európskej únie a druhá pre vozidlá z krajín mimo Európskej únie). Používateľ po prihlásení so svojím emailom a heslom si bude môcť pozrieť stav svojej žiadosti – či bola potvrdená, zamietnutá, alebo ešte nebola posúdená.

Hlavné kritériá:

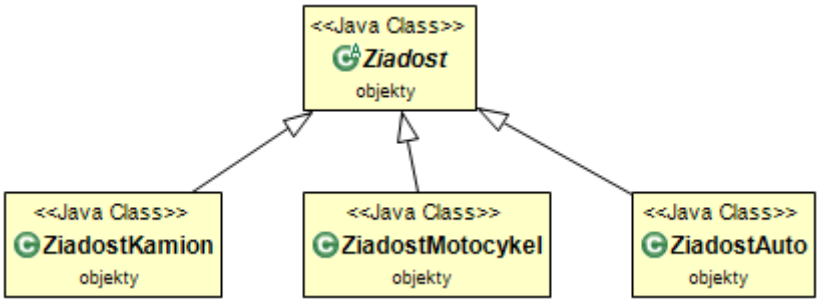
Dedenie používam v dvoch oddelených hierarchiách dedenia.

UML diagram Hierarchie *Vozidiel*:



1. UML: Hierarchia Vozidiel

UML diagram hierarchie *Žiadostí*:



2. UML: hierarchia Žiadostí

Polymorfizmus používam v dvoch oddelených hierarchiách.

Prvá línia: Konštruktor triedy *Uzivatel* akceptuje argument typu *Vozidlo.* V triedach *ZiadostAuto*, *ZiadostMotocykel* a *ZiadostKamion* v metóde *poziadaj*() do neho posielam typy *Auto*, *Motocykel* a *Kamion.*

Druhá línia: V triede *Registracia* je atribút typu *Ziadost,* jej metóda *setZiadost* prijíma argument typu *Ziadost.* V triede *RegistraciaOkno* posielam do tejto metódy typy *ZiadostAuto, ZiadostMotocykel, ZiadostKamion.*

Zapuzdrenie používam vo všetkých triedach, atribúty sú private a pristupujem k nim pomocou get a set metód.

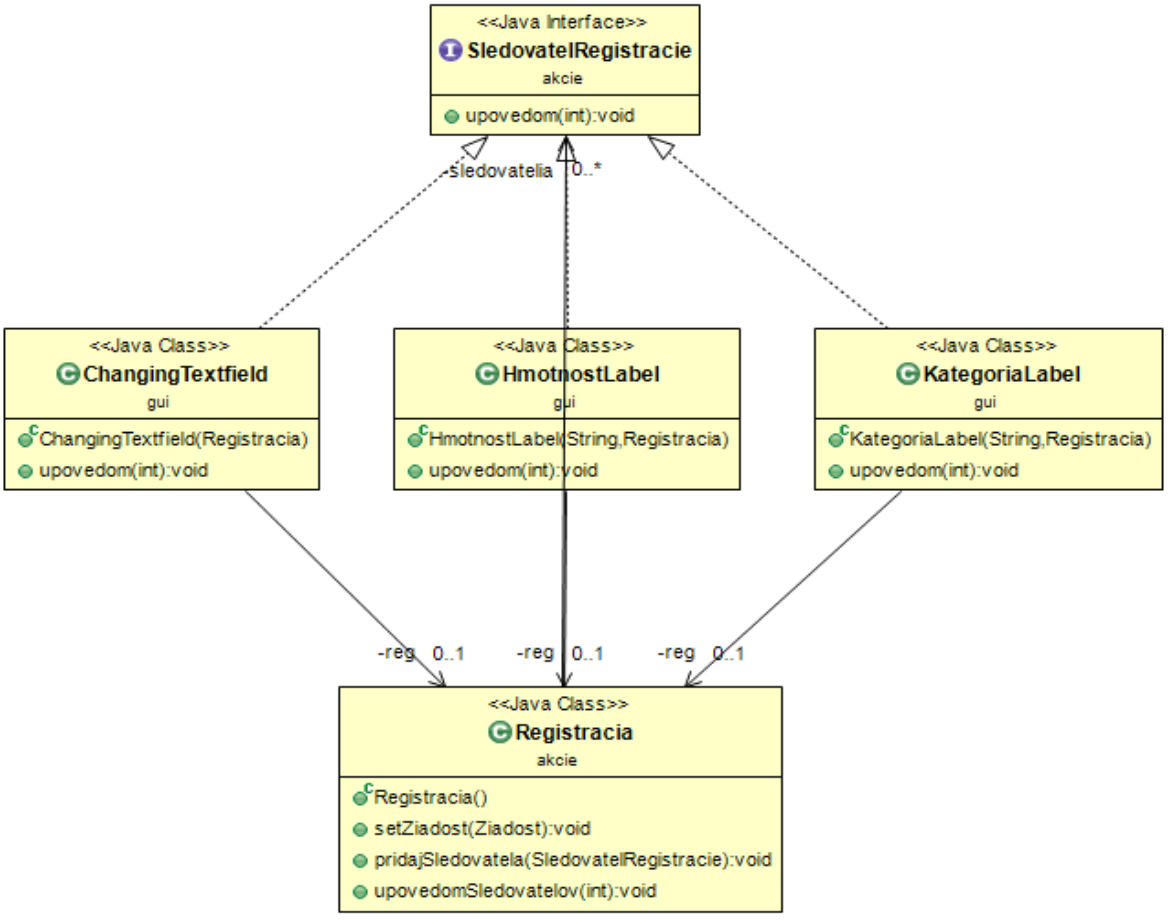
Agregáciu používam napríklad v triede *Uzivatel,* ktorá obsahuje *Vozidlo.* Potom ju používam aj v triede *Ziadost,* ktorá obsahuje *Uzivatel* aj *Vozidlo.*

Oddelenie GUI od aplikačnej logiky – každá trieda GUI obsahuje controller.

Organizácia do balíkov – v programe mám 3 balíky. Balík *gui* obsahuje triedy GUI, balík *akcie* obsahuje controllery a balík *objekty* obsahuje objekty.

Ďalšie kritériá:

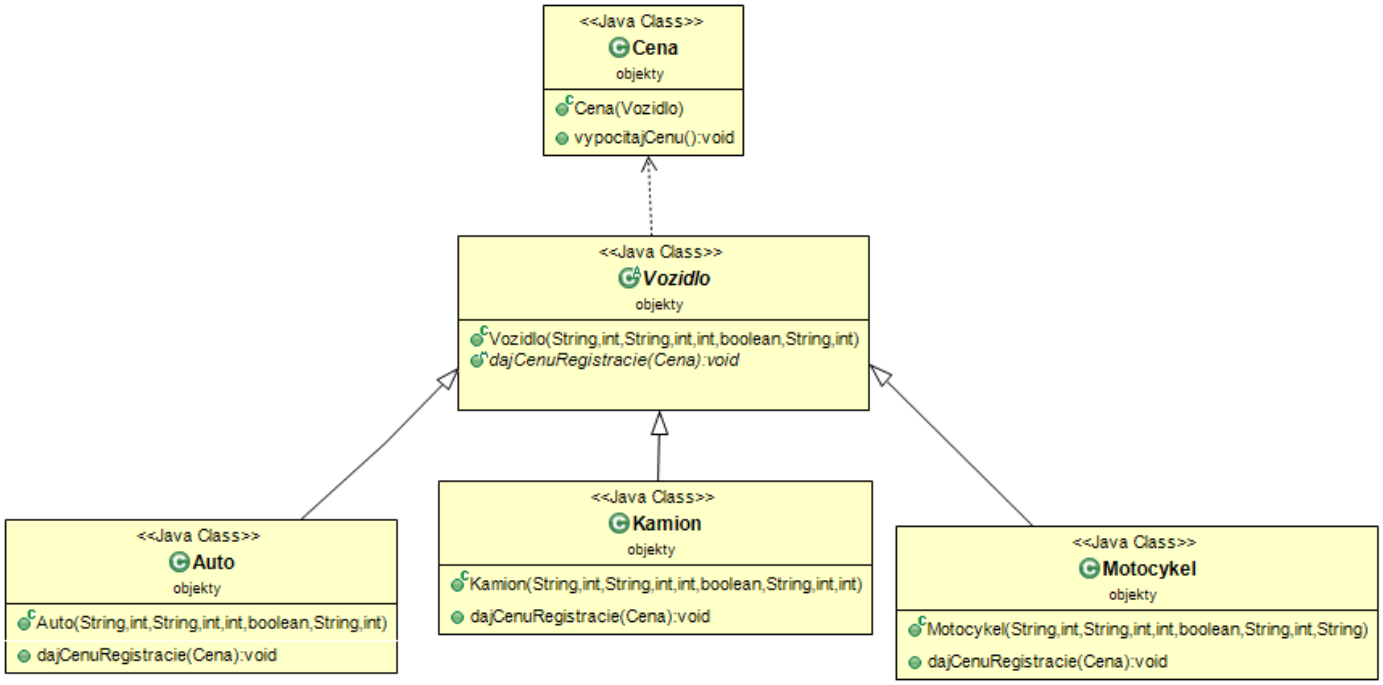
Použitie návrhových vzorov – v programe používam 2 návrhové vzory – Observer a Visitor.



3. UML: vzor Observer

Vzor **Observer** je použitý na sledovanie triedy *Registracia.* V okne, v ktorom používateľ podáva žiadosť je ChoiceBox a na základe vybraného prvku z ChoiceBoxu sa mení viditeľnosť prvkov *HmotnostLabel, ChangingTextField* a *KategoriaLabel,* ktoré implementujú rozhranie *SledovatelRegistracie.*

Vzor **Visitor** je použitý na výpočet ceny za registráciu pre dané vozidlo. Trieda *Cena* predstavuje *Element.* V konštruktore obsahuje metódu *accept(visitor),* a v metóde *vypocitajCenu* volá *visitor.visit(this).* Abstraktná trieda *Vozidlo* predstavuje *Visitor* a predpisuje použitie metódy *dajCenuRegistracie(Cena).* Táto metóda predstavuje metódu *visit(Element).* Pre každý typ vozidla sa táto metóda správa inak.



4. UML: Vzor Visitor

Vlastné výnimky - **ChybnyVstupException** používam na zachytenie udalostí, keď používateľ zadá nesprávny vstup(do TextFieldu kde má ísť integer, zadá string). Výnimka **ZlyLoginException** zachytáva zadanie nesprávnej kombinácie loginu a hesla pri prihlasovaní používateľa alebo administrátora. Obe tieto výnimky po zachytení vypíšu Alert s hláškou.

Poskytnutie GUI – Program má GUI, celé vytvorené manuálne, bez použitia nejakých nástrojov na tvorbu GUI. Je dostatočne oddelené od aplikačnej logiky, každé okno má controller.

Použitie RTTI – *instanceof* používam pri výpise detailov o vozidle, napríklad v okne po prihlásení používateľa – trieda *PrihlasenyUserOkno*. Ak je typ vozidla *Motocykel* tak v ListView sa zobrazia detaily o motocykli, ak je typ *Auto* vypíše detaily o aute a ak je typ vozidla *Kamion* tak ListView vypíše údaje pre kamión.

Použitie vhniezdených tried – Vhniezdenú triedu *Prihlas* používam v triede *LoginAdminOkno*, trieda *Prihlas* implementuje EventHandler a obsahuje metódu *handle().* Cez túto vhniezdenú triedu vykonávam kliknutie na tlačidlo *prihlasit.* Použitie: *prihlasit.setOnAction(new Prihlas())*

Použitie lambda výrazov – Takmer všetky kliknutia na tlačidlo vykonávam cez lambda výrazy. Napríklad v triede *LoginUserOkno* po kliknutí na tlačidlo *Naspat*.

Použitie referencie na metódu – Používam ju v triede *LoginAdminOkno*, pri kliknutí na tlačidlo *Naspat*.

Zoznam dôležitých commitov do GitHubu:

1. Spravené základné GUI, prvých pár okien a preklikávanie sa medzi nimi.
2. Pridane hlavne triedy do programu
3. Pridaný ArrayList do ktorého sa ukladajú používatelia
4. Pridaná serializácia
5. Funguje načítavanie zo serializovaného súboru
6. Pridané okno na registráciu vozidla
7. Pridané okno admina, kde vidí všetky žiadosti v ListView
8. Po vybraní žiadosti si admin môže pozrieť detaily zadané používateľom
9. Admin môže prijať alebo zamietnuť žiadosť
10. Pridaný vzor Observer, pri výbere v ChoiceBoxe sa mení na GUI vybraný prvok
11. Pridané prihlasovanie sa heslom pre používateľa, zobrazuje sa mu stav jeho žiadosti
12. Pridaný vzor Visitor na výpočet Ceny
13. Pridaná vlastná výnimka na chybný vstup
14. Pridaná druhá hierarchia dedenia a polymorfizmu – hierarchia žiadostí
15. Pridaná vlastná výnimka na nesprávne prihlasovacie údaje
16. Žiadosti majú v adminovom okne v ListViewe farbu podľa ich stavu
17. Už raz posúdená žiadosť sa viac nedá adminom zmeniť
18. Pridaná vhniezdená trieda, ktorá slúži na vykonanie akcie po kliknutí na tlačidlo
19. Začnem robiť komentáre na JavaDoc
20. Pridané RTTI a referencia na metódu
21. Používateľovi sa po prihlásení zobrazujú aj detaily ktoré zadal pri žiadosti