**FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ**

**SLOVENSKÁ TECHNICKÁ UNIVERZITA**

Ilkovičova 2, 842 16 Bratislava 4

2021/2022

**Objektovo-orientované programovanie**

Semestrálny projekt

Cvičiaci: Ing. Peter Križan Vypracoval: Patrik Tomčo

Čas cvičení: Streda 16:00 – 17:40 AIS ID: 116316

**Názov projektu: EasyCarBid**

# Zámer projektu

Môj projekt sa bude týkať E-aukcie spojenej s autami. Začne prihlásením, ktoré bude rozoznávať klasického používateľa, prémiového používateľa a administrátora. Administrátor bude mať možnosť vytvoriť novú aukciu, špecifikovaním jej trvania a pre akého používateľa je určená. Prémiový zákazník bude preto vidieť viac aukcií ako klasický používateľ. Klasický používateľ sa stane prémiovým, keď obdrží aspoň dve autá. Všetci používatelia môžu navrhnúť cenu auta a musia tak spraviť aspoň dvaja, aby mohla byť položka auta predaná tomu, kto ponúkol viac. Výherca následne dostane upozornenie o výhre a informácie o vyhratom aute, a to sa uloží medzi jeho obdržané položky (niečo ako osobná garáž). Používateľ bude mať taktiež peňaženku, v ktorej bude vidieť svoje reálne financie a bude si ich môcť zameniť za špeciálnu virtuálnu menu, využívanú v aukcii. Peniaze si bude môcť do účtu vkladať a zamieňať, ako uzná za vhodné, ale nie je možné zameniť špeciálnu virtuálnu menu späť na klasickú. Každé auto bude samostatná položka určená na kúpu a bude obsahovať základné informácie ako: id aukcie, názov auta, motorizácie, kilometre atď. Po rozkliknutí danej položky sa zobrazí jej podrobný popis obsahujúci stav vozidla či výbavu.

## Zmeny oproti zámeru projektu

Aukcie nemajú stanovený čas, ale zato existujú dva druhy aukcií. Klasický používateľ sa nie je schopný stať prémiovým. Ďalej na vyhratie aukcie je potrebné, aby na ňu vsadili aspoň traja používatelia. Výherca síce nedostane upozornenie o získaní danej položky, ale daná aukcie sa stále uloží medzi jeho obdržané aukcie (do garáže). Nakoniec samotné autá nie sú samostatnými položkami, ale iba ako súčasť aukcie.

V projekte sa namiesto týchto funkcií nachádzajú iné. Používateľ si dokáže pridať danú aukciu do svojho “Wish listu”. Vie si taktiež vyobraziť dostupné aukcie, na ktoré môže vsadiť ľubovoľnú sumu špeciálnych peňazí, vie sa dostať do svojej garáže a do svojej peňaženky. V garáži sa nachádzajú autá, ktoré obdržal a dokáže si o nich zobraziť bližšie informácie. Peňaženka mu poskytuje možnosť vložiť peniaze a zameniť si ich za špeciálnu menu, využívanú v aukciách.

Admin (alebo administrátor) má možnosť okrem iného vyobraziť si štatistiku, ktorá obsahuje počet konkrétnych používateľov a konkrétnych aukcií. Ďalej má možnosť vyobraziť si správy, ktoré ho informujú o predaných aukciách. Tieto správy sú pridané všetkým registrovaným Adminom, keď sa aukcia predá. Nakoniec si dokáže vyobraziť 3 najčastejšie používateľmi “Wish listované” aukcie.

# Štruktúra programu

## Hlavné časti

Hlavnými časťami môjho projektu sú:

* dedenie pri používateľoch a ich agregované objekty (peňaženka, garáž, wishList)
* aukcie a ich podtypy, vrátane typov aukcií (SealedBidAuction, AbsoluteAuction)
* databázy uchovávajúce informácie o používateľoch, aukciách a agregovaných objektoch
* “kontrolery“ nad UI elementami

## Diagram tried

Obrázok, na ktorom je text, monitor, čierne, obrazovka

Automaticky generovaný popis

Ako je z diagramu zrejmé, v mojom programe sa nachádza viacero využití dedenia. Začínajúc z hora, trieda User sa delí na triedu *Customer* a *Admin*. Trieda *Customer* zoskupuje používateľov, ktorí dokážu vkladať ponuky na aukcie. Tými sú *StandardUser* a *PremiumUser*. Objekty typu Admin sú schopné vytvárať aukcie a majú prehľad o registrovaných používateľoch a o aukciách. Taktiež aj realizujú rozhranie Observer. Ďalej sa nachádza dedenie v rámci tried predstavujúcich databázy. Abstraktná trieda *Database* zastrešuje všetky databázy vyskytujúce sa v programe. Tieto databázy uchovávajú dáta o používateľoch, aukciách, peňaženkách a garážach používateľov a wish listy používateľov. Všetky tieto databázy obsahujú metódy na načítanie dát a na ich uloženie. Ďalej nasleduje dedenie v rámci “kontrolerov“ hlavných obrazoviek (*MainScreenController* a *AdminMainScreenController*). Obidve triedy dedia od spoločnej nadtriedy *MainScreen* a realizujú rozhranie s názvom *MainScreenInterface.* V ňom sa nachádza implicitná implementácia metódy logOut(). Nasleduje dedenie v rámci tried reprezentujúcich autá. Od triedy *Car* dedia triedy *StandardCar*, *ElectricCar* a *HybridCar*. Tieto triedy sú využívané na rozdelenie aukcií na kategórie, podľa áut, ktoré obsahujú. Nakoniec ostávajú realizácie rozhraní s názvami *AuctionType* a *Subject*. Rozhranie *Subject* plní rolu v *Observer* vývojovom vzore a *AuctionType* vo vývojovom vzore *Strategy*.

# Hlavné kritériá

## Dedenie

Dedenie je realizované v mojom projekte na viacerých miestach.

Medzi používateľmi, ktorí sa delia na Admin a *Customer*. *Customer* sa ďalej delí na *StandardUser* a *PremiumUser*.

Medzi autami, ktoré sa delia na *StandardCar*, *ElectricCar* a *HybridCar*.

Taktiež aj pri databázach, kde všetky databázy zastrešuje jedna abstraktná trieda *Database*. Ostatné databázy dedia od tejto nadradenej. Medzi databázy patria napríklad: *DatabaseOfUser* alebo *DatabaseOfAuctions*. Ďalšie databázy uchovávajú dáta o agregovaných objektoch.

## Polymorfizmus

Polymorfizmus sa využíva napríklad pri prihlasovaní sa používateľa do programu. Program iba zistí, o ktorého používateľa sa jedná, a ten sa prihlási podľa toho, akého je typu.

Polymorfizmus je taktiež využívaný pri načítavaní objektov to databáz pri spustení programu. Všetky rozličné databázy sú uložené v jednom poli a na začiatku programu každá podľa seba načíta dané dáta.

Taktiež je využitý pri metódach placeBid(), ktoré rozhodujú, ako môžu objekty typu *Customer* podávať ponuky na aukcie.

Nakoniec je polymorfizmus využitý aj pri vývojových vzoroch ako Obsever a Strategy.

## Zapuzdrenie

Zapuzdrenie sa v mojom projekte nachádza na mnohých miestach. Atribúty v rôznych triedach sú označené ako *private*, *proteced* alebo *public*. Pre všetky atribúty označené *private kvantifikátorom existujú príslušné getery a setery.*

## Agregácia

V mojom projekte existuje viacero agregovaných objektov. Medzi nie patrí napríklad peňaženka, garáž a wishList, ktoré sú priradené každému objektu typu *Customer*. Ďalej každá aukcia obsahuje auto, ktoré je jej hlavnou časťou.

# Vedľajšie kritériá

## Použitie návrhových vzorov

V mojom programe sa vyskytuje návrhový vzor Observer a Strategy.

Observer je využívaný na informovanie každého registrovaného admina o predaní aukcie. To znamená, že zakaždým ak je nejaká aukcie predaná, AuctionManager (Subject) informuje všetkých Adminov (Observers), o predaní danej aukcie a uloží túto správu medzi ich novinky, ktoré si po prihlásení môžu pozrieť,

Strategy je využitý pri rozhodovaní o aký typ aukcie ide. Rozlišujem dva typy aukcií. SealedBidAuction a AbsoluteAuction. SealedBidAucion je typ aukcie, kde používatelia ponúkajú ľubovoľnú sumu väčšiu ako minimálna cena aukcie. Aukciu vyhrá používateľ, ktorý ponúkol najvyššiu sumu. To je vyhodnotené hneď po tom, ako je vložená posledná (tretia) ponuka. AbsoluteAuction je typ aukcie, kde každý používateľ musí ponúknuť vyššiu sumu peňazí ako predchádzajúci používateľ. To znamená, že posledný používateľ je výherca.

## Ošetrenie mimoriadnych stavov

Ošetrenie mimoriadnych stavov je realizované napríklad pri vytváraní aukcie, kedy používateľ musí vybrať práve jeden druh aukcie a práve jeden typ aukcie. Keď zaškrtne viac políčok, vyhodí sa TooManySelectedBoxesException a odznačí všetky políčka a informuje používateľa a nutnosti vybrať práve jedno políčko z každej kategórie. Pokiaľ používateľ neoznačil ani jedno políčko, vyhodí sa NoSelectedBoxException a upozorní používateľa. Ďalej, ak sa v peňaženke pokúša používateľ zameniť viac peňazí, ako aktuálne má, vyhodí sa TooMuchCurrencyExeption a upozorní používateľa a zároveň nastaví hodnotu peňazí na zmenenie na najväčšiu možnú podľa toho, koľko peňazí používateľ vlastní.

## Poskytnutie grafického používateľského rozhrania

Moje používateľské rozhranie je vytvorené s pomocou programu s názvom SceneBuilder a ide o FXML súbor. To zabezpečuje oddelenie od aplikačnej logiky. EventHandler-y sú v mojom programe zodpovedné za vyobrazenie okna s nápovedou, keď používateľ drží pravé tlačídko myši na určitých tlačídkach. Taktiež sa zmení farba písma na príslušnom tlačídku. Po uvoľnení pravého tlačídka myši pomocné okno zmizne a farba písma sa vráti do pôvodného stavu.

## Explicitné použitie viacniťovosti

Explicitné použitie viacniťovosti je v mojom programe využité napríklad pri výpise všetkých dostupných aukcií. Vtedy vytvorím novú niť, lamba výrazom popíšem, čo sa má vykonať a spustím túto novú niť. Taktiež používam príkaz Platform.runLater(), ktorý zaradí vykonanie bloku kódu, na ktorý ukazuje lambda výraz, do rady za ostatné úlohy, ktoré sa vykonávajú.

## Použitie generickosti vo vlastných triedach

Použitie generickosti vo vlastných triedach je v mojom programe využité na dvoch miestach. V triede *WishList*, ktorá obsahuje parametrizovaný ArrayList pložiek, ktoré sa doň dokážu uložiť. Ďalej je generickosť využitá v triede *Serializator* v ktorej sa nachádza metóda storeObejct(). Táto metóda dokáže vďaka generickosti zaserializovať akýkoľvek objekt.

## Explicitné použitie RTTI

Explicitné použitie RTTI je využívané napríklad pri počítaní počtu konkrétnych používateľov, či aukcií. Taktiež to je využité pri vyobrazovaní aukcií, keďže štandardný používateľ nemôže vidieť prémiové aukcie.

## Použitie vhniezdených tried a rozhraní

Použitie vhniezdených tried bolo popísané vyššie v sekcii hlavné požiadavky. Čo sa rozhraní týka, tak v programe sa vyskytujú práve 4. Dve z nich (*Subject*, *Observer*) sú súčasťou *Observer* návrhového vzoru. Rozhranie s názvom *AuctionType* je časťou *Strategy* návrhového vzoru a pomáha pri rozhodovaní, o aký typ aukcie ide. Posledným rozhraním je *MainScreenInterface,* ktorý predpisuje, aké metódy musia hlavné obrazovky obsahovať.

## Použitie lambda výrazov alebo referencií na metódy

Použitie lambda výrazov alebo referencií na metódy je využité napríklad pri vyššie spomenutej metóde Platform.runLater(). Taktiež sú využívané pri vytváraní eventHandler-ov. Lambda výrazy sú taktiež použité pri vytváraní a spúšťaní vlastných nití.

## Použitie implicitnej implementácie metód v rozhraniach

Použitie implicitnej implementácie metód v rozhraniach je realizované v rozhraní s názvom *MainScreenInterface,* kde sa nachádza default metóda logOut(). Keďže ju takisto volajú objekty typu *Customer* a *Admin,* nachádza sa v jednom rozhraní, ku ktorému majú prístup objekty oboch typov.

## Použitie serializácie

Objekty v mojom programe serializujem rovnakým spôsobom, aký bol prednesený na prednáške. Vytvorím objekt špeciálneho zapisovača a objekt pridám do ArrayListu a ten následne serializujem do textového súboru.

# Hlavné verzie programu

Vytvorenie aukcie – prvý skutočný pokrok v programe. Zatiaľ existoval iba logIn a GUI elementy

Strategy – Vylepšenie aukcií a pridanie nových typov aukcií a pribudlo taktiež vypisovanie aukcií

Peňaženka a garáž – Ukladanie a načítavanie agregovanných objektov a práca s nimi

Pracovná verzia – Majoritná časť zadania sprevádzaná úpravami kódu a vnútornej štruktúry tried atď.

Observer – Pridanie observer návrhového vzoru

Finálna verzia – pridanie ďalších funkcionalít akou je napríklad wish list a ďalšie zlepšovanie kódu