**Záverečný Projekt**

**Maximilián Strečanský**

**ID: 116298**

**Slepé E-aukcie**

*Krátka dokumentácia v ktorej popisujem ako podľa mňa projekt plní kritéria hodnotenia finálnej verzie*

**ZÁMER:**

*Cieľom tejto práce je zrealizovať aplikáciu na slepé aukcie, kde budú používatelia vytvárať ponuky pre kupcov. Takýto typ aukcie by mohol pomôcť aj chudobnejším obdržať predmet, ktorý by na klasickej aukcií nikdy nemohli obdržať kvôli ostatným ponúkateľom.*

**HLAVNÉ KRITÉRIÁ:**

**PRVÁ HIERARCHIA – Používateľ**

UserClass

Buyer

Seller

InstantBuyer

VipBuyer

**DEDENIE:**

*public* *class* Buyer *extends* *UserClass* {

* Trieda Buyer dedí z prvotnej triedy UserClass
* Rovnako platí aj pre triedu Seller, ktorá dedí z UserClass.
* VipBuyer a InstantBuyer dedí z Buyer

**PREKONÁVANIE**

@*Override*

*public* *int* getRole() {

return 1;

}

* Každa trieda v tejto hierarchií prekonáva metódu getRole()
* Vrátené číslo nám určuje rolu používateľa
* 0-Kupec, 1-Predajca, 2-Vip Kupec, 3-Instantný Kupec
* Pomocou tejto metódy určujeme či môže používateľ predávať/vytvárať aukcie alebo v prípade VIP Kupca môžeme ponúkať na viac ako 1 predmet

**DRUHÁ HIERARCHIA - Predmet**

ItemClass

New

Historical

**DEDENIE**

*public* *class* Historical *extends* *ItemClass* {

* Trieda Historical a New dedí z triedy ItemClass

**PREKONÁVANIE**

*// Vratime 1 pre typ historicky predmet*

@*Override*

*public* *int* getType() {

return 1;

}

* Obe triedy Historical a New prekonávajú metódu getType()
* 1 -> typ predmetu je Historical a 2 -> typ predmetu je New

**AGREGÁCIA**

*public* *class* ItemClass *implements* *Serializable*{

*private* *String* name;

*private* *UserClass* owner;

*private* *boolean* sold;

*private* *int* highestOffer;

*private* *UserClass* hOfferOwner;

* V triede predmetu uchovávame triedu majiteľa a toho, kto ponúkol najväčšiu ponuku
* V prípade, že aukcia skončí, nastavíme majiteľa na majiteľa najväčšej ponuky

**KÓD VHODNE ORGANIZOVANÝ DO BALÍKOV**

* Balíky: exceptions, items, linkedlist, pages, users a spúšťací program Main.java

**ĎALŠIE KRITÉRIÁ**

**OŠETRENIE MIMORIADNYCH STAVOV**

*// Osetrenie prazdneho mena a hesla*

if (password.length() == 0 || username.length() == 0) {

try {

throw new AlertException("Invalid username or password");

} catch (*AlertException* *e1*) {

e1.printStackTrace();

}

return;

}

* Na prihlasovacej stránke LoginPage.java využívame ošetrenie stavu kedy políčko pre meno alebo heslo nie je vyplnené.
* Rovnaký princíp využívame na hlavnej stránke MainPage.java
* Vyhodíme okno s informáciou o zle zadanom hesle alebo mene

**POSKYTNUTIE GRAFICKÉHO POUŽÍVATEĽSKÉHO ROZHRANIA A ODDELENIE OD APLIKAČNEJ LOGIKY**

* Všetky využité metódy sú uložené v triedach poprípade v balíku pages/controllers

**POUŽITIE VIACNIŤOVOSTI**

* Využité v hlavnom okne MainPage.java, kde spúšťame samostatnú niť pre vypisovanie aktuálneho času

**POUŽITIE GENERICKOSTI – SPÁJANÝ ZOZNAM**

*public* *class* LList {

* Využívame vlastnú implementáciu spájaného zoznamu pre uchovávanie prihlasovacích údajov v triede UserClass.java

**POUŽITIE RTTI**

*// Vyuzitie RTTI pre zistenie roly*

role.setText("Rola: " + user.getClass().getSimpleName());

* Využívame pri hlavnej stránke MainPage.java pre vypísanie role používateľa v používateľskom rozhraní

**POUŽITIE LAMBDA VÝRAZOV**

*StringFunction* newLine = (s) -> s + "\n";

* V triede UserClass v metóde pre vypisovanie aktuálnych ponúk alebo ponúk, ktoré vlastní daný používateľ využívame Lambda výraz
* Tento výraz nám ku koncu prvku pridá nový riadok pre korektné vypísanie a oddelenie prvkov

**POUŽITIE SERIALIZÁCIE / DESERIALIZÁCIE**

* Využívame na ukladanie a načítanie ponúk v hlavnom okne MainPage.java v balíku controllers -> MainController.java