Zoznam využitých OOP princípov v projekte

- **Dedenie:** z abstraktnej triedy *Pouzival* dedí *Klient* ale aj abstraktná trieda Zamestnanec z ktorej potom dedia zvyšní zamestnanci. Tiež mám dedenie z abstraktnej triedy *Tovar*z nej dedí *Zosit* a *TovarRozmer*.
- **Agregácia:** sa nachádza vo viacerých častiach, každý používateľ má atribút login triedy *Login* a správy typu *Spravy, Klient* má *Adresu*, *Pracovník* stroja *zase Stroj*.
- **Polymorfizmus:** Prekonávam metódu toString() kvôli zobrazeniu informácií o prihlásenom používateľovi na domovskej obrazovke. *Pouzivatel* v metóde toString() vráti meno, email a adresu ako string oddelené novými riadkami. Klient túto metódu prekonáva pretože má aj adresu.
- **Preťažené metódy:** V triede AController mám preťaženú metódu zobraz_okno, ak je v argumente aj stage, prepnem celú scénu, ak je tam iba root a rozmery okna vyskočí nové okno navyše.
- **Oddelenie logiky od GUI:** Oddelenie realizujem tak, že si vytváram manuálne fxml súbory a k nim pripájam vlastné kontroléri, ktoré spracuvávajú akcie a prepínanie medzi scénami. Každá scéna je 1 fxml súbor s vlastným controllerom.
- **Návrhový vzor observer:** Tento vzor využívam viac krát. Každú objednávku sleduje 1 singleton trieda *ManazerObjednavok*. Pri každej zmene objednávky (výroba nejakej časti tovaru) sa upozorní tento *ManazerObjednavok*, ktorý skontroluje či už nie je celá objednávka hotová a nemá sa posunúť skladníkovi na odoslanie. Tiež trieda Sklad má zoznam pozorovateľov (skladníkov) a keď treba sklad doplniť upozorní všetkých skladníkov. Posledný krát kedy využívam Observer je pri Správach používateľov vo výrobnom procese. Kde controller je vlastne observer pre *Spravy* a keď príde nová správa vie že ju má pripojiť do textArea. Tu observer implementuje rozhranie *PozorovatelSprav* ktoré definuje metodu notify s argumentom správy.
- **Návrhový vzor Strategy:** Vzor využívam pri pracovníkoch výroby a samotnom procese výroby. Mám interface *Vyroba*, ktorý implementuje každý pracovník. Tento interface definuje metódu vyrob_tovar, ktorá má ako argument objednávku. Každý pracovník túto metódu prekonáva a má ju upravenú podľa seba. Pri procese výroby volám metódu ((Vyroba)this.p).vyrob_tovar(o) kde sa vyberie „stratégia“ podľa toho či je to *PracovníkFotiek*, Zošitov alebo obálok.
- **Multithreading:** Je využívaný pri výrobe. Výroba spočíva v tom, že pracovník skontroluje či je dostatok materiálu na sklade a ak hej zavolá metódu svojho stroja, ktorý vyrába daný produkt. Táto metóda (spusti_proces) vytvorí novú niť cez triedu *ProcesVyroby* (moja trieda ktorá dedí z Thread) do tohto procesu výroby si posúvam samotný stroj, objednávku a tovar ktorý vyrábam. Potom v *ProcesVyroby* v metóde run volám stroj.zacni_vyrabat. Táto metóda je synchronized a má na starosti výpis a samotnú výrobu tovaru. (Nachvíľu dám thread sleep aby tam bol nejaký časový rozdiel pri výrobe, že to chvíľu trvá).
- **Explicitné použitie RTTI:** Znova sa v projekte nachádza viac krát. Napr. viditeľnosť niektorých tlačidiel je určená podľa toho či je používateľ inštanciou

Zamestnanca/Klienta/Vyroby atď. V ManazerObjednavok v metóde prirad_objednávku_pracovnikom podľa typu Tovaru priraďujem pracovníka, typ tovaru zisťujem cez instanceof. Tretie využitie RTTI je napr. vo funkcií vyrob_tovar() kde preskočím výrobu tovaru, ktorý nemá daný pracovník na starosti (pracovník fotiek vyrába iba fotky)

- **Lambda výraz:** Sa nachádza v triede *Spravy* v metóde toString() kde prechádzam všetky správy a postupne vytváram jeden dlhý string. Správy prechádzam cez foreach kde využívam lambda výraz.
- **Serializácia:** Pôvodne som mal serializáciu pri každom zatvorení celej aplikácie avšak od prezentovania som to zmenil na tlačidlo ulož (niekedy nechcem uložiť stav v akom je celá aplikácia). Serializujem používateľov, objednávky a sklad. Ak sa nepodarí deserializovať načítam používateľov a objednávky z textových súborov a sklad vytvorím nanovo.

Poznámka: Čo sa zmien od prezentovania týka, pridal som vlastnú výnimku, upravil niektoré časti kódu (nefungoval napríklad správne Sklad). Tiež som odstránil interface prístup do skladu a pridal triedu Zamestnanec, ktorý vlastne plní podobnú funkciu, pretože zamestnancom prideľuje atribút sklad. Kód som aj okomentoval, pridal zobrazovanie vybavených objednávok, odosielanie objednávok a zmenil serializáciu ktorá je teraz vykonávaná kliknutím tlačidla ulož.

O projekte

Projekt sa sústredí na plánovanie a proces výroby papierových výrobkov, slúži klientom a zamestnancom firmy. V softvéri sa používateľ prihlási svojimi prihlasovacími údajmi a vykonáva operácie ktoré sú mu povolené. Môžu sa vytvárať nové objednávky (automaticky sa vypočíta cena objednávky a priradí konkrétnym pracovníkom), prezerať aktuálne objednávky a vybavené objednávky. Proces výroby spočíva vo vybratí konkrétnej objednávky, skontrolovania stavu skladu (ak nie je dostatok materiálu nemôže sa pokračovať s výrobou a sklad musí doplniť skladník), posunutie výroby danému stroju, ktorý určitý čas pracuje na výrobe.