# Projekt OOP – Zdravotnícky systém – Dokumentácia

## Pôvodný zámer projektu (z 13.3.2020)

Navrhovaný softvér disponuje možnosťou prihlásiť sa v niekoľkých roliach, použitím prihlasovacieho mena, hesla a zadaním SMS kľúča, čo zabezpečuje jeho dôveryhodnosť vzhľadom na spracovávanie citlivých údajov. Lekári, lekárnik a zdravotná poisťovňa dokážu pristupovať k niektorým informáciám o pacientovi na základe jeho rodného čísla.

**Pacient** v systéme vidí predpísané lieky, nárok na čerpanie doplatku na lieky a existenciu výmenných lístkov. Môže meniť svoje osobné údaje, ako adresu, či telefónne číslo.

**Všeobecný lekár** má možnosť predpisovať pacientovi lieky na **predpis**. Môže predpísať bežné lieky, alebo, na základe odporúčania, môže predpísať aj špecializované lieky. Taktiež vydáva pacientovi výmenný lístok pre špecializovaného lekára. Dokáže pacientovi meniť údaje o krvnej skupine a alergiách na lieky. Pridáva do evidencie nového pacienta.

**Lekár – špecialista** má rovnaké možnosti ako všeobecný lekár, no podľa špecializácie má oprávnenie vydávať predpisy a odporúčania na predpis špecializovaných liekov a zdravotných pomôcok. Môže vyšetriť len pacienta s platným výmenným lístkom.

**Lekárnik** vidí len dosiaľ nevydané recepty na lieky pacienta a nárok na čerpanie doplatku od zdravotnej poisťovne. Liek môže vydať.

**Zdravotná poisťovňa** pacientovi eviduje informáciu o platbe zdravotného poistenia, na základe ktorej môže čerpať doplatok na lieky. Prepláca zdravotnú starostlivosť o pacientov lekárom a doplatky na lieky lekárnikom. Pridáva do evidencie nového lekára.

## Štruktúru systému vo forme diagramu

*tba*

## Plnenie hlavných kritérií

Program je oproti **zámeru projektu** zredukovaný, z dôvodu nepotrebnosti implementovať všetku opísanú funkcionalitu pre úspešné splnenie kritérií. SMS autentifikácia, vydávanie odporúčaní, či možnosť upravovať svoje osobné údaje boli z finálnej verzie projektu vypustené.

Program obsahuje **2 oddelené hierarchie dedenia**. Lekár je podtriedou triedy Lekárnik, pretože dedí základnú štruktúru agregovaných údajov. Lekár má oproti lekárnikovi pridanú funkcionalitu a dokáže evidovať pacienta, vydávať výmenné lístky a predpisovať lieky. Špecializovaný lekár je podtriedou Lekára, dedí všetky jeho metódy, prekonáva metódu evidencie pacientov, pretože zatiaľ čo na navštívenie Lekára nie je potreba mať výmenný lístok, pre navštívenie Špecializovaného lekára je nevyhnutné disponovať výmenným lístkom ku konkrétnemu špecialistovi. Špecializovaný lekár tiež pridáva metódu na vytvorenie predpisu pre zdravotnícku pomôcku, ktorú nemôže predpísať Všeobecný lekár. Lekári a Špecializovaní lekári sú evidovaní v zozname Lekárov v Zdravotnej Poisťovni. Tu dochádza k polymorfizmu. Pri evidovaní nového pacienta pre nejakého Lekára zo zoznamu lekárov je volaná podľa typu lekára prekonaná, alebo pôvodná metóda evidencie. Špecializovaný lekár taktiež volá metódy svojho nadtypu, najmä v GUI pri vypisovaní pacientov.

Ďalšia **hierarchia dedenia** je dedenie Predpisu. Výmenný Lístok a Zdravotnícka pomôcka sú 2 na sebe nezávislé podtriedy Predpisu. Výmenný lístok pridáva oproti predpisu metódu na zistenie typu lekára, pre ktorého je určený. Zdravotnícka pomôcka v sebe okrem zdedených atribútov Predpisu drží aj typ zdravotníckej pomôcky a taktiež má metódu na jej zistenie. Pacient eviduje Predpisy a Zdravotnícke pomôcke v jednom poli, z ktorého číta tieto záznamy, teda dochádza k rozdielnemu spôsobu výpisu predpisu pre volanie prekonanej metódy, hoci sú držané spolu, v jednom poli.

V projekte nachádza niekoľko **rozhraní**. Všetky typy osôb používajú rozhrania predpisujúce metódy na zápis a vrátenie osobných a prihlasovacích údajov. Údaje o osobách, a osoby v rámci poisťovne sú zapúzdrené a **agregované**.

**Aplikačná logika oddelená od GUI** – tba.

Kód je organizovaný do 3 **balíkov**, podľa príslušnosti ku GUI, osobám alebo poisťovni.

**Dokumentácia** – tba.

**JavaDoc** – tba.

## Plnenie ďalších kritérií (priebežne 9)

**Použitie návrhových vzorov** – Použitý návrhový vzor Observer pri výpise zoznamu Lekárov.

**Ošetrenie mimoriadnych stavov prostredníctvom vlastných výnimiek** – Poisťovňa overuje prihlasovacie údaje, ak sa nenájde zhoda, metóda vyhodí výnimku, ktorá sa prejaví ako informačné okno.

**Poskytnutie grafického používateľského rozhrania oddelene od aplikačnej logiky a s aspoň časťou spracovateľov udalostí (handlers) vytvorenou manuálne** – handlery sú vytvorené manuálne všetky, GUI je oddelené od AL. Ráta sa ako splnenie 2 kritérií.

**Použitie serializácie** – Objekt poisťovne a jej agregované objekty sú serializované.

**Použitie lambda výrazov alebo referencií na metódy (method references)** – vytvorené funkčné rozhranie Správa, ktorým sa v triede Lekárnik generujú správy pomocou lambda výrazov.

**Použitie implicitnej implementácie metód v rozhraniach (default method implementation)** – rozhranie ZistiOsobneUdaje obsahuje default metódu na výpis všetkých údajov o prihlásenom používateľovi. Použitie sa nachádza v GUI, pri prihlásení Lekára.

**Explicitné použitie viacniťovosti (multithreading) – spustenie vlastnej nite priamo alebo prostredníctvom API vyššej úrovne (trieda Task a pod.)** – použité na vypisovanie Predpisov v triede Pacient. V triede Pacient v metóde citajPredpisy sa nachádza vnorená trieda ktorá beží v samostatnej niti a vypisuje predpisy. Zabezpečuje plynulý beh programu, ak je predpisov na vypísanie príliš veľa. Ak by v programe nebola a predpisov na vypísanie by bolo príliš veľa, GUI by zamrzlo, kým by sa všetky nevypísali. Takto sa dá s GUI plynule pracovať ďalej aj pri náročnom výpise. Podobné riešenie novou niťou vo vhniezdenej triede je aj v metóde citajListky triedy Pacient, určenej na čítanie výmenných lístkov.

**Použitie vhniezdených tried a rozhraní** – vhniezdenie triedy s novou niťou, ktorá je popísaná v predchádzajúcom kritériu. Vhniezdená v triede Pacient.

*V programe sa zatiaľ nenachádza (priebežne 3):*

*Použitie generickosti vo vlastných triedach – implementácia a použitie vlastnej generickej triedy*

*Explicitné použitie RTTI – napr. na zistenie typu objektu alebo vytvorenie objektu príslušného typu*

*Použitie aspektovo-orientovaného programovania (AspectJ)*

## Zoznam hlavných verzií programu odovzdaných do GitHub

*tba*