**SE321 OBEZBEĐENJE KVALITETA, TESTIRANJE I ODRŽAVANJE**

**SOFTVERA**



**Jesenji semester** **2021/22**

**PROJEKTNI ZADATAK**

PROJEKAT- Sistem lanca apoteka

Student:

Bogdan Janković 3920

**SADRŽAJ PROJEKTA**

[Uvod 4](#_Toc92704480)

[Specifikacija softverskog sistema koji se testira 4](#_Toc92704481)

[Kratak opis poslovnog problema koji se rešava softverskim sistemom i njegova veza sa okruženjem 4](#_Toc92704482)

[Funkcionalni zahtevi softverskog sistema 5](#_Toc92704483)

[Nefunkcionalni zahtevi softverskog sistema 7](#_Toc92704484)

[Upotrebljivost 7](#_Toc92704485)

[Performanse 7](#_Toc92704486)

[Sigurnost 7](#_Toc92704487)

[Dostupnost 7](#_Toc92704488)

[Integritet 7](#_Toc92704489)

[Pouzdanost 8](#_Toc92704490)

[Izmenjivost 8](#_Toc92704491)

[Proverljivost 8](#_Toc92704492)

[Ponovna upotreba 8](#_Toc92704493)

[Dijagram slučajeva korišćenja softverskog sistema 8](#_Toc92704494)

[Funkcionalnosti softverskog sistema koje će biti testirane 9](#_Toc92704495)

[Funkcionalnosti softveskog sistema koje neće biti testirane 10](#_Toc92704496)

[STRATEGIJA TESTIRANJA 10](#_Toc92704497)

[Tipovi testiranja koji će se sprovesti i u kojoj meri 10](#_Toc92704498)

[Pretpostavke, rizici, plan ublažavanja ili plan za vanredne situacije 11](#_Toc92704499)

[Rizici 11](#_Toc92704500)

[*Rizici puštanja novog sistema u rad:* 11](#_Toc92704501)

[Poslovne pretpostavke i zavisnosti 11](#_Toc92704502)

[Tablica povezanosti zahtevi-slučajevi testiranja 12](#_Toc92704503)

[Matrica sledljivosti 12](#_Toc92704504)

[Osoblje uključeno u testiranje, vremenski raspored testiranja i budžet 13](#_Toc92704505)

[Uloge i odgovornosti 13](#_Toc92704506)

[Raspored testiranja 14](#_Toc92704507)

[Kriterijumi: 14](#_Toc92704508)

[Praćenje defekata 14](#_Toc92704509)

[Test okruženje 15](#_Toc92704510)

[PRIMENA 12 CASPER JONES-OVIH PRAVILA NA PRIMERU SISTEMA ZA LANAC APOTEKA 15](#_Toc92704511)

[REALIZACIJA TESTIRANJA SOFTVERSKOG SISTEMA 17](#_Toc92704512)

[Pregled koda - Programming Mistake Detector (PMD) 17](#_Toc92704513)

[Primena Programming Mistake Detectora (PMD): 18](#_Toc92704514)

[Primena McCabe-ova siklomatske složenosti 20](#_Toc92704515)

[Test slučaj 1 21](#_Toc92704516)

[Test slučaj 2 21](#_Toc92704517)

[Test slučaj 3 22](#_Toc92704518)

[Test slučaj 4 23](#_Toc92704519)

[Test slučaj 5 23](#_Toc92704520)

[Slučajevi testiranja na bazi specifikacije 25](#_Toc92704521)

[*JUnit testiranje*: 26](#_Toc92704522)

[Testiranje metodom crne kutije – Black Box 28](#_Toc92704523)

[EKVIVALENTNO PARCELISANJE 28](#_Toc92704524)

[ANALIZA GRANIČNIH VREDNOSTI 30](#_Toc92704525)

[Testiranje metodom bele kutije – White Box 32](#_Toc92704526)

[Pokrivenost naredbi 32](#_Toc92704527)

[Pokrivenost odluka (grana) 32](#_Toc92704528)

[Pokrivenost prostih uslova 33](#_Toc92704529)

[Pokrivenost petlji 33](#_Toc92704530)

[MOCKING TESTIRANJE 33](#_Toc92704531)

[Integraciono testiranje 34](#_Toc92704532)

[Izveštaji o realizovanom testiranju 36](#_Toc92704533)

[Izveštaj o testiranju korišćenjem obrasca za izveštaj o grešci 37](#_Toc92704534)

[ODRŽAVANJE, DALJI RAZVOJ I KONTINUALNI PROCES POBOLJŠANJA SOFTVERA 37](#_Toc92704535)

[Model estimacije troškova prilikom održavanja odabrane aplikacije 37](#_Toc92704536)

[Pristup za postizanje višekratne upotrebljivosti 38](#_Toc92704537)

[ZAKLJUČAK 40](#_Toc92704538)

[LITERATURA 40](#_Toc92704539)

# Uvod

Testiranje softvera je integralni deo procesa razvoja softvera, sa ciljem osiguravanja kvaliteta finalnog proizvoda. U ljudskoj prirodi je da pravimo greške. Neke greške mogu biti sitne i nebitne, dok druge mogu biti veoma ozbiljne i skupe, pa čak i dovesti do ljudskih žrtava. Testiranje softvera je neophodno kako bi se otkrile greške načinjene u svim fazama razvoja softvera, koje se nakon toga mogu ispraviti. Otkrivanjem i ispravljanjem grešaka se obezbeđuje viši nivo kvaliteta softvera, čime se stiče poverenje i zadovoljstvo krajnjih korisnika softvera.

Pojam testiranja softvera je star koliko i samo programiranje. Još od prvih napisanih programa je bilo neophodno proveriti da li se program ponaša ispravno i na takav način kako je definisano. U današnje vreme, iako možda to na prvi pogled ne izgleda tako, naš način života u potpunosti zavisi od softvera koji je ključni faktor u velikom boju sistema koje svakodnevno koristimo. Softver definiše ponašanje mrežnih rutera, bankarskih mreža, telefonskih sistema i samog Interneta.

Komercijalni softver se razvija sa tačno definisanim ciljem, kako bi rešio neki konkretni problem krajnjih korisnika. Neophodno je da se program korektno izvršava – u suprotnom nije od koristi, pošto krajnji korisnici neće moći da ga koriste na predviđen način. Greške u radu programa mogu izazvati nezadovoljstvo korisnika, finansijske gubitke, gubitke značajnih podataka, čak i ljudske žrtve. Kako bi se obezbedilo korektno izvršavanje softvera, softver se razvija po fazama, sa jasnim zahtevima i dokumentacijom, i adekvatno testira.

## Specifikacija softverskog sistema koji se testira

### Kratak opis poslovnog problema koji se rešava softverskim sistemom i njegova veza sa okruženjem

Usled trenutnih svetskih zdravstvenih okolnosti u vezi sa pandemijom virusa Covid-19 došlo se do ideje o razvijanju sistema koji ce olakšati nabavku neophodnih farmaceutskih i medicinskih sredstava usko povezanih za pandemiju i zdravlje generalno i to na inovativnije odnosno načine brze i bezkontaktne dostave.

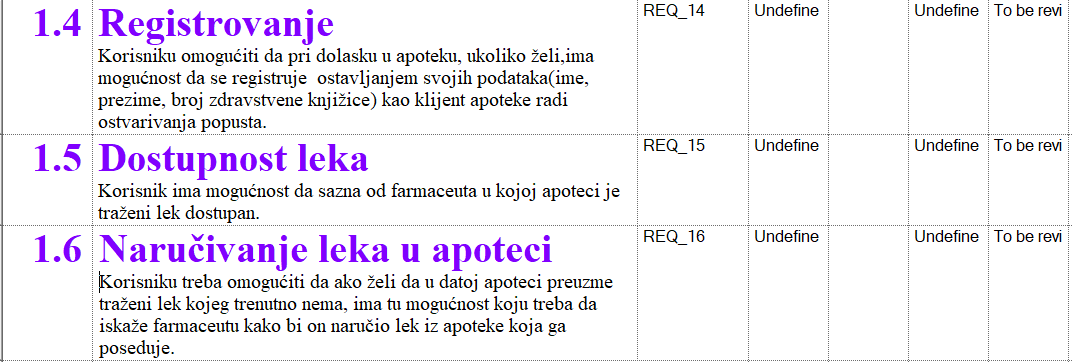
Problem koji ovaj sistem pokušava da reši je često zaboravljena terapija koja je u ovom periodu veoma važna kod svih osoba, da li zahvaćenih pandemijom ili ne. Ovaj problem je inače zastupljen u društvu ali sada ga je pogotovo važno rešiti ili makar pomoći pri rešavanju. Najveći uzrok tome je sve brži način života ali se i rešenje vidi u tome da je sve više ljudi povezano sa pametnim telefonima te njegovim mogućnostima podsećanja na uzimanje terapije. Postojećih rešenja u vidu samih sistema prodaje i administracije apoteka već ima u vidu informacionih web sajtova i ograničene online prodavnice.

Sistem koji se razvija bi pružio i sve popularniju integraciju sa dostavljačkim kompanijama radi što brže dostave kako je većini korisnika to preko potrebno ili čak i nužno zbog neophodne izolacije usled ove bolesti kao i razvoj lekomata (u stilu sveprisutnih paketomata za pošiljke) za bezkontaktno preuzimanje koje je naročito važno usled pandemije i naravno komunikaciju sa farmaceutimau vidu online postavljanja pitanja. Ovakav vid komunikacije, obzirom na to da se radi o preko neophodnoj komunikaciji sa stručnim farmaceutom, pružio to da se korisnici bezbednije osećaju povodom informacije koje dobijaju vezano za svoje probleme ili neke preporuke u moru trenutnih informacija pročitanih na internetu.

### Funkcionalni zahtevi softverskog sistema

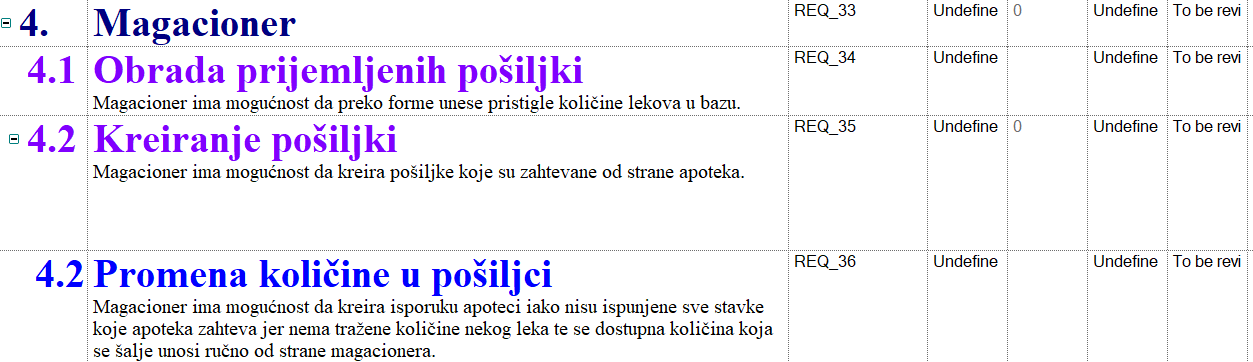














### Nefunkcionalni zahtevi softverskog sistema

### Upotrebljivost

**UP1:** Skup od 95% zadataka odnosno funkcija na sistemu korisnik može da izvrši pravilno bez da mu je potrebna pomoć.

**UP2:** Sva polja koja imaju jasno predefinisane vrednosti trebaju imati automatsku validaciju u slučaju ne validnog unosa.

### Performanse

**PER1:** Sistemu neće biti potrebno duže od 10 sekundi da se ponovo pokrene u slučaju prekida rada.

**PER2:** Vreme odziva celokupne platforme ne sme biti duže od 3 sekunde.

**PER3:** Sve operacije promene podataka se moraju izvršiti za najviše 5 sekundi.

### Sigurnost

**SEC1:** Sistem će zaključati korisnički nalog nakon tri uzastopna neuspela pokušaja logovanja u roku od 5 minuta.

**SEC2:** Farmaceut/Magacioner mora da promeni privremenu lozinku koju mu je dodelio vlasnik radi bezbednosti i to u prethodno nekorišćenu lozinku odmah nakon prve uspešne prijave sa privremenom lozinkom.

**SEC3:** Farmaceut u jednom trentku može da bude prijavljen samo u jednoj apoteci.

### Dostupnost

**AVL1:** Platforma u apotekama mora biti dostupna u toku radnog vremena apoteka lanca odnosno radnim danima od 8 do 22h kao i subotom od 9 do 17h.

**AVL2:** Celokupan sistem a mora biti raspoloživ u bilo kom trenutku izuzev u periodu redovnog održavanja sistema.

### Integritet

**INT1:** Sistem se štiti od neovlašćenog dodavanja, brisanja ili modifikacije podataka.

**INT2:** Sistem vrši sigurnosne kopije podataka na kraju svakog radnog dana.

### Pouzdanost

**REL1:** Srednje vreme između kvarova na komponenti za prodaju biće najmanje 90 dana.

### Izmenjivost

**MOD1:** Programer za održavanje koji ima iskustva sa sistemom biće u mogućnosti da modifikuje sistem receptne prodaje u skladu sa propisima RFZOa za manje od 10 sati efektifnog rada.

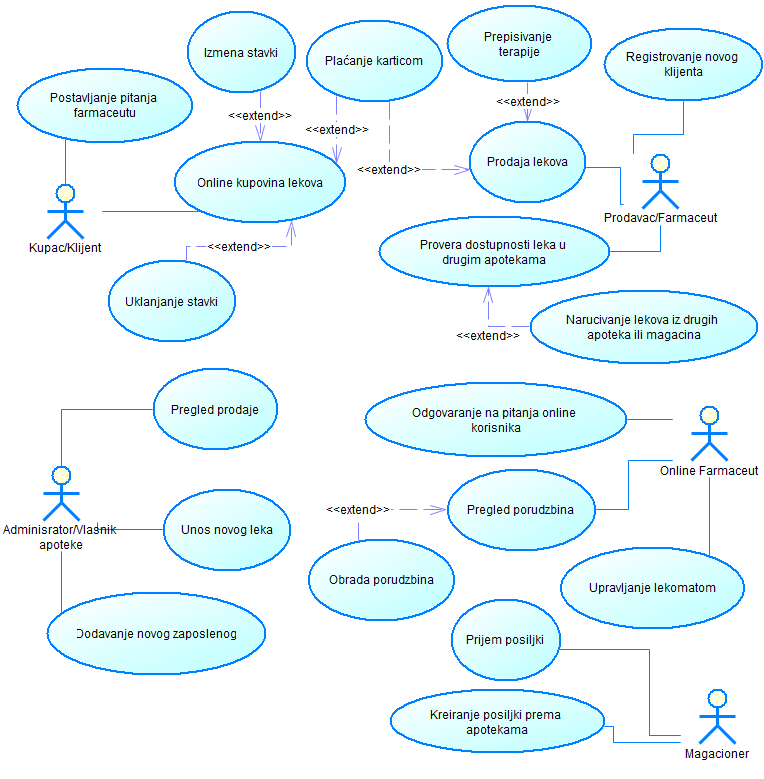
### Proverljivost

**VER1:** Za svaki deo koda biznis logike biće pisani odgovarajući testovi kako prilikom pormena neće biti previše posla oko provere validnost.

### Ponovna upotreba

**REU1:** Kod će biti pisan uz praćenje OOP pristupa razvoja što će olakšati ponovnu upotrebu.

## Dijagram slučajeva korišćenja softverskog sistema

****

## Funkcionalnosti softverskog sistema koje će biti testirane

1. Centralna baza lekova lanca apoteka
2. Online shop
3. Online kupovina sa popust pogodnostima
4. QR kod kod online narudžbina
5. Sistem za prodaju lekova u apotekama
6. Kupovina lekova preko e-recepta
7. Prepisivanje terapije preko integrisanog Google kalendara
8. Naručivanje lekova iz drugih apoteka lanca
9. Registrovanje klijenata
10. Sistem za pitanja i odgovore
11. Glovo integracija
12. Kiosk lekova – lekomat
13. Upravljanje lagerom u magacinu
14. Pregled prodaje
15. Administracija lekova i zaposlenih od strane vlasnika
16. Integracija sa eksternim sistemom ALIMSa

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Funkcionalnost** | | **Izdanje 1.0** | | **Izdanje 1.1** | | **Izdanje 1.2** | |
| FE-1 | Potpuno implementirano | | / | | / | |
| FE-2 | Potpuno implementirano | | / | | / | |
| FE-3 | Nije implementirano | | Potpuno implementirana | | / | |
| FE-4 | Implementirano preko jedinstvenog broja porudžbine | | Potpuno implementirano | | / | |
| FE-5 | Potpuno implementirano | | / | | / | |
| FE-6 | Potpuno implementirano | | / | | / | |
| FE-7 | Implementirano u vidu remindera | | Potpuno implementirano u Google kalendar | | Implementirano i za druge aplikacije kalendara | |
| FE-8 | Implementirano u vidu dodatnog naručivanja iz magacina | | Potpuno implementirano | | / | |
| FE-9 | Nije implementirano | | Potpuno implementirano | | / | |
| FE-10 | Implementirano samo za tesktualni format | | Potpuno implementirano | | / | |
| FE-11 | Potpuno implementirano | | / | | / | |
| FE-12 | Potpuno implementirano | | / | | / | |
| FE-13 | Potpuno implementirano | | / | | / | |
| FE-14 | Implementirani samo osnovni izveštaji | | Implementirano i za PDF, DOC, DOCX datoteke | | Potpuno implementirano | |
| FE-15 | Nije implementirano | | Implementirano za lekove | | Potpuno implementirano | |
| FE-16 | Impelentirano samo za lekove sa receptnom prodajom | | Potpuno implementirano | | / | |

U datom projektu će biti testirane sve implementirane funkcionalnosti iz prvog planiranog izdanja.

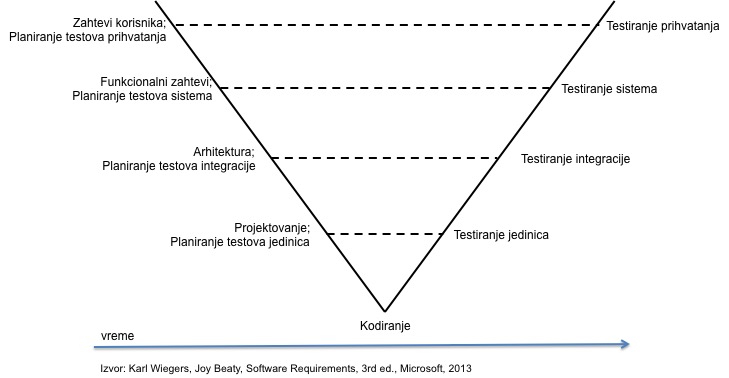
## Funkcionalnosti softveskog sistema koje neće biti testirane

Funkcije ostavljene za naredna dva izdanja će biti testirane u sledećim ciklusima razvoja i testiranja.

# STRATEGIJA TESTIRANJA

## Tipovi testiranja koji će se sprovesti i u kojoj meri

Slika prikazuje V model razvoja softvera koji će se privenjivati. Pokazuje ispitne aktivnosti koje započinju paralelno sa odgovarajućim razvojnim aktivnostima. Ovaj model ukazuje da su testovi prihvatanja izvedeni iz potreba korisnika, testovi sistema zasnovani su na funkcionalnim zahtevima, a testovi integracije zasnovani su na arhitekturi sistema. Ovaj model je primenljiv bez obzira da li su aktivnosti razvoja softvera koji se testiraju odnose na proizvod u celini, posebno izdanje, ili na pojedinačni priraštaj razvoja.



## Pretpostavke, rizici, plan ublažavanja ili plan za vanredne situacije

### Rizici:

RI-1: Apoteka će pretrpeti poslovne gubitke zbog zoga što na vreme nije unapredila svoj poslovni sistem kao i sam odgovor na novonastalu pandemiju usled katastrofalnosti defekata u razvoju.

RI-2: Apoteka će doživeti velike finansijske gubitke ukoliko ne koristi inovativne benefite i nije u korak sa konkurentima.

RI-3: Apoteka će morati da poveća broj ljudskih resursa usled želje za bržim testiranje kao i samim razvojem, što će dovesti do većih rashoda.

RI-4: Rast prodaje može opasti uprkos velikoj potražnji ukoliko kupci lanca apoteka postanu nezadovoljni zbog brzine usluga kao i problema oko neadekvatne ponude rešenja kupovine u novonastaloj situaciji.

## Rizici puštanja novog sistema u rad:

1. Određeni broj zaposlenih će pokazati otpor prema novom sistemu dok se ne prilagode.
2. Biće potrebno angažovati dodatne farmaceute za održavanje novih lekomata i obradu online porudžbina.
3. Biće potrebno nabaviti određeni broj čitača za očitavanje bar i QR kodova.
4. Inicijalno će vreme potrebno za obavljanje rutinskih zadataka na sistemu biti duže u odnosu na stari sistem.

### Poslovne pretpostavke i zavisnosti

1. Svi podaci sa papirnih dokumenata se moraju digitalizovati i preneti u nov sistem.
2. Svi podaci koji su čuvani u trenutnom sistemu se moraju preneti u novi.
3. Interfejsi za integraciju sa drugim sistemima prvenstveno za potrebe dostave se mogu promeniti u toku razvoja i održavanja sistema, i on se mora prilagoditi promenama.
4. RFZO može menjati pravilnik o receptnoj prodaji lekova na koje sistem mora adekvatno reagovati.
5. Sistem mora ograničiti da se u njega unose samo lekovi trenutno odobreni od strane Agencije za lekove i medicnska sredstva (ALIMS).

## Tablica povezanosti zahtevi-slučajevi testiranja

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Broj zahteva** | **Test 1** | **Test 2** | **Test 3** | **Test 4** | **Test 5** | **Test 6** | **Test 7** | **Test 8** |
| **REQ\_2** | **X** | **X** | **X** | **X** | **X** | **X** |  |  |
| **REQ\_3** | **X** | **X** |  |  | **X** |  |  |  |
| **REQ\_4** | **X** | **X** | **X** |  | **X** |  |  |  |
| **REQ\_5** | **X** |  | **X** | **X** | **X** |  |  |  |
| **REQ\_6** | **X** |  |  |  |  |  |  |  |
| **REQ\_7** | **X** |  |  | **X** | **X** |  |  |  |
| **REQ\_8** | **X** |  |  | **X** | **X** |  |  |  |
| **REQ\_9** | **X** |  |  |  |  | **X** |  |  |
| **REQ\_10** |  | **X** |  |  |  |  | **X** |  |
| **REQ\_11** |  | **X** |  | **X** |  |  |  |  |
| **REQ\_12** |  |  |  | **X** |  | **X** | **X** |  |
| **REQ\_13** |  | **X** |  |  | **X** |  |  | **X** |

## Matrica sledljivosti

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Broj zahteva | Modul | High level requirement | Low level requirement | Test case |
| REQ-01  Sistem za prodaju | Centralni sistem prodaje | 1.1 Omogućiti farmaceutima sistem za prodaju lekova | 1.1.1🡪Prodaja lekova | Regularna prodaja-REV 1 |
|  |  |  | 1.1.2🡪Prodaja lekova sa receptom |  |
|  |  |  | 1.1.3🡪 Prodaja sa popust karticom | Loyality card kupovina test |
|  |  |  | 1.1.4🡪 Izmena stavki |  |
|  |  |  | 1.1.5🡪 Uklanjanje stavki |  |
|  |  | 2.1 Komunikacija sa magacinom | 2.1.1🡪 Provera stanja traženog leka na lageru | Komunikacija apoteka-magacin |
|  |  |  | 2.1.2🡪 Naručivanje leka iz magacina |  |
|  |  | 3.1. Registrovanje klijenata | 3.1.1🡪 Registrovanje klijenata sa zdravstvenom knjižicom | Provera registracije – REV 1  Provera registracije – REV 2 |

## Osoblje uključeno u testiranje, vremenski raspored testiranja i budžet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Pokretač  (unos ciljeva) | Ograničenje | | Stepen slobode  (unos dozvoljenog opsega) |
| Plan i rokovi | Izdanje 1.0 da bude spremno do 1.2.2022., a izdanje 1.1 do 1.3.2022. godine | | Neophodno veoma detaljno testiranje radi očuvanja integriteta sistema | Rokovi smeju da se probiju za maksimalno nedelju dana |
| Članovi tima | Oformiti tim sa glavnim fokusom za testiranje i kasnije održavanje | | Najveća veličina tima: 1 PM, 1 TL, 1 stručnjak iz domena farmacije, 4 programera i 3 testera | Najveća promena tima: 6 programera I 4 testera |
| Trošak | Ulaganje u novi sistem radi smanjenja grešaka pri radu | | Planirani budžet 15000€ | Prihvatljivo probijanje budžeta do 15% bez recenzije sponzora |

## Uloge i odgovornosti

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Uloga | Akter / korisnik | | Odgovornosti | |
| Farmaceut | Ana Nikolajević | * Prodaja lekova u apoteci * Prodaja lekova sa e-receptom * Registrovanje klijenata | |
| online farmaceut | Nikola Lukić | * Odgovaranje na pitanja korisnika * Administracija online narudžbina | |
| Magacioner | Žarko Lilić | * Upravljanje lagerom u magacinu * Kreiranje pošiljki prema apoteci * Kreiranje porudžbenica prema dobavljačima | |
| Vlasnik | Petar Momčilović | * Izveštavanje o prodaji * Dodavanje lekova u asortiman * Evidencija i administracija rada zaposlenih | |
| Finansijska služba | Ivana Ilić | * Upravljanje knjigovodstvenim uslugama * Obračuni plata * Izveštaji zarade | |
| Marketing služba | Mila Jović | * Kreiranje specijalnih pogodnosti, akcija, programa lojalnosti i konkursa | |

## Raspored testiranja

|  |  |  |
| --- | --- | --- |
| Aktivnost | Planirani rok | Krajnji rok |
| Proučavanje sistema | 03.12.2021. | 20.12.2021. |
| Pisanje test slučajeva | 25.12.2021. | 30.12.2021. |
| Izvršavanje test slučajeva | 20.01.2022. | 01.02.2022. |
| Isporuka | 10.02.2022. | 15.02.2022. |

## Kriterijumi:

• Broj grešaka kategorije 2=0;

• Broj grešaka kategorije 3=0;

• Broj grešaka kategorije 4<3 po modulu;

• Broj uspešno obavljenih testova=99 %;

• Svi planirani testovi moraju biti izvršeni

## Praćenje defekata

Proces zavođenja i praćenja defekata odnosno menadžment defekata veoma je značajan na ovom projektu usled kratkog vremena razvoja, i predstavlja jedno od bitnih zaduženja softver testera. Za menadžment defekata se obično koristi softverski sistem za praćenje defekata. Postoji veliki broj ovakvih alata dostupnih na tržištu, a neki od najpopularnijih koji će se I koristiti su: ***Jira*** i ***Bugzilla***.

Defekt je stvarni problem koji treba da bude rešen. Generalno važi opšte pravilo testiranja: što se ranije bug otkrije i ispravi, niži su ukupni troškovi obezbeđivanja kvaliteta sistema. Sistem za praćenje defekata treba da sadrži informaciju o tome u kojoj fazi razvoja softvera je defekt stvoren i u kojoj fazi je detektovan. Ako su te dve faze iste, onda to znači da defekt nije propagirao. Ako netačan zahtev „promakne“ prilikom statičkog testiranja zahteva i kao takav bude implementiran od strane programera, testiran od strane testera i prijavljen tek prilikom korisničkog testiranja, sav rad na tom zahtevu je protraćeno vreme (a može dovesti i do gubitka poverenja korisnika).

## Test okruženje

Za uspešnu primenu sistema, računari zaposlenih u apotekama moraju da imaju instaliran Windows 10 operativni sistem. Konfiguracijom softvera se definišu detalji o operativnim sistemima kao što su Windows, Linux, UNIX i Mac i raznim čitačima kao što su Google Chrome, Firefox, Safari.

Za potrebe skladištenja svih podataka u digitalnom formatu, potrebno je obezbediti skladište od 2TB. Procenjena količina dokumenata na godišnjem nivou je oko 1TB. Hardverskom konfiguracijom se definišu informacije o različitim veličinama RAM-a, ROM-a i procesora. Farmaceuti, pored računara, moraju da imaju skener koji može da skenira u rezoluciji 300PPI za bar i QR kodove. Osim toga, svaki farmaceut mora da ima i jedistveni ključ za opštem pristupu lekomata.

# PRIMENA 12 CASPER JONES-OVIH PRAVILA NA PRIMERU SISTEMA ZA LANAC APOTEKA

**Pravilo 1** - (Estimacija veličine izvornog koda): tiče se izračunavanja funkcionalnih tačaka i pretvaranja u linije koda.

Funkcionalne tačke FP računate po formuli sa jednostavnim težinskim faktorom:

Broj FP= broj ulaza x 3 + broj izlaza x 4 + broj upita x 3+ broj matičnih datoteka x 7 + broj interfejsa x 5 = 1\* 3 + 2 \* 4 + 3 \* 3 + 4 \* 7 + 1 \* 5 = **53 Funkcionalne tačke**

LOC = 53 x 53= **2 809 linja kod ako koristimo Javu.**

**Pravilo 2** – Estimacija dokumentacije:

Ovo pravilo nam omogućava da estimiramo svu dokumentaciju vezanu za softverski paket, uključujući specifikaciju i uputstva. FP(broj podešenih funkcionalnih tačaka)=**53** Dokumentacija= FP1.15

stranica Dokumentacija=531.15= **96 stranica**

**Pravilo 3**- Estimacija odudaranja korisničkih zahteva: Ovo pravilo kaže da prilikom realizacije projekta odudaranje od korisničkih zahteva je u proseku oko **2%** mesečno.

**Pravilo 4**- Estimacija broja slučajeva testiranja: Funkcionalne tačke nam takođe omogućavaju da estimiramo broj test slučajeva koji su potrebni da se dobro testira softverski proizvod.

FP(broj podešenih funkcionalnih tačaka)=**53**

Broj test slučajeva= FP1.2

Broj test slučajeva=531.2= **117 test slučajeva**

**Pravilo 5**- Estimacija potencijalnog broja grešaka: Način na koji možemo da estimiramo mogućnost defekta u našem softveru je:

Potencijalni broj grešaka=FP1.25

Potencijalni broj grešaka= 531.25= **143 moguće grešake**

**Pravilo 6** – Estimacija efikasnosti otklanjanja greške: Ovo pravilo nam govori kako je teško pronaći greške prilikom testiranja. Ovo pravilo kaže da svaki korak testiranja pronalazi i otklanja određen broj grešaka u svakoj fazi. Ti brojevi dati su u vidu tabele.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Defekti | Faza analize zahteva | FAZA  dIZAJNA | fAZA RAZVOJA | fAZA TESTIRANJA | fAZA  oDRZAVANJA |  |
| ITERACIJA 1 | **40%** | **25%** | **20%** | **10%** | **0 %** |  |
| ITERACIJA 2 | **/** | **30%** | **20%** | **15%** | **0 %** |  |
| ITERACIJA 3 | **/** | **/** | **35%** | **10%** | **2%** |  |
| ITERACIJA 4 | **/** | **/** | **/** | **60%** | **10%** |  |
| ITERACIJA 5 | **/** | **/** | **/** | **/** | **0 %** |  |

**Pravilo 7**- Estimacija efikasnosti organizovanog otklanjanja grešaka: Ovakav način otklanjanje grešaka je skuplji i zahtevniji vremenski ali i efikasniji. Ovo pravilo kaže da svaka inspekcija dizajna će pronaći i otkloniti određen procenat grešaka u tom trenutku.

**Pravilo 8-** Estimacija efikasnosti otklanjanja grešaka nakon puštanja softvera u rad: Programeri zaduženi za održavanje softvera mogu ispraviti određen broj grešaka koji zavisi od TMM i CMM nivoa.

**Pravilo 9**- Estimacija trajanja realizacije projekta: Estimacija trajanja realizacije projekta DM (Development in months) predstavlja jednu od najvažnijih informacija za klijente, project menadžere i softverske programere.

Izračunava se na sledeći način:

FP(broj podešenih funkcionalnih tačaka)=53

DM=FP0.4 [KM] kalendarskih meseci

DM=530.4= **4.8 [KM] kalendarskih meseci**

**Pravilo 10**- Estimacija potrebnih ljudi za realizaciju projekta: Za planiranje projekta je veoma važno koliko vam ljudi treba za realizaciju samog projekta. Ovo pravilo nam pomaže u tome.

Prosečna produktivnost projektanata=***150***

Broj projektanata= FP/Prosečna produktivnost projektanata

Broj projektanata=53 / 150 = **1 projektant**

**Pravilo 11**- Estimacija ljudi potrebnih za održavanje softvera:

Izračunava se na sledeći način:

Broj ljudi za održavanje = FP/ prosečna efikasnost održavanja

Broj ljudi za održavanje=FP / 750 = 53 /750 = **1 čovek**

**Pravilo 12**- Estimacija ukupnih napora u realizaciji softverskog projekta: Ovo pravilo predstavlja kombinaciju pravila 9 i 10.

Primenjuje se na sledeći način:

Ukupni Napor= Ukupno vreme \* broj ljudi= 4,8 \* 1 = **5 čoveka-meseci**

# REALIZACIJA TESTIRANJA SOFTVERSKOG SISTEMA

## Pregled koda - Programming Mistake Detector (PMD)

Pregled koda je sve češća praksa u razvojnim timovima. To je tok posla u kojem programeri predaju svoj kod na povratne informacije pre spajanja grana ili uvođenja koda u produkciju. Ove povratne informacije obično daju kolege, bilo drugi programeri, menadžer ili tehnički rukovodilac. Jedan od najpoznatijih oblika pregleda koda je Github zahtev za povlačenjem, u kojem programeri ostavljaju komentare na određene redove koda i na kraju odobravaju ili odbijaju predložene promene. Aspekti na koje bi trebalo obratiti pažnju pri pregledu koda:

1. Čitljivost, takođe „razumljivost“

Čitljivost u softveru znači da je kod lako razumljiv. U ovom slučaju, razumevanje koda znači mogućnost da se lako vide ulazi i izlazi koda, šta svaki red koda radi i kako se uklapa u širu sliku. Kada čitate kod, trebalo bi da vam bude relativno lako da prepoznate ulogu određenih funkcija, metoda ili klasa.

2. Održavanje

Jedan od najčešćih razloga zbog kojih kod na kraju postaje naporan za rad je zato što nije napisan da bi bio lako proširiv i promenljiv.

3. Bezbednost

Sigurnosne ranjivosti često ulaze u kod jer programeri pišu kod ne razmišljajući o sigurnosti. To može značiti da oni pišu nesigurni kod koji u sistem unosi ranjivosti ili koriste biblioteke i alate koji su zastareli ili imaju poznate bezbednosne probleme.

4. Brzina i performanse

Razmotrite performanse u dve dimenzije: performanse za korisnike i potrošnja resursa. Performanse za korisnike odražavaju fokus na to koliko brzo kod radi za krajnjeg korisnika. Dugotrajni upiti baze podataka, neoptimizirana sredstva i višestruki zahtevi za API mogu uticati na to da vaš kod deluje sporo.

7. Pouzdanost

Pouzdan kod je kod otporan na kvarove. Kada stvari krenu po zlu u pouzdanom kodu, korisničko iskustvo je u najvećoj mogućoj meri zaštićeno od udara. Pouzdan kod se piše pod pretpostavkom da će stvari propasti, da se sredstva ponekad neće učitati, API zahtevi povremeno vraćaju 500 grešaka i nedostaju zapisi baze podataka. Kada se predviđa određeni nivo neuspeha, njime se može elegantno postupati.

8. Ponovna upotrebljivost

Proverite da li je kod napisan imajući na umu moguće buduće slučajeve upotrebe. Na primer, ako pregledate kod za tržište koje brzo širi svoj asortiman, uverite se da se kod može lako ažurirati kako bi podržao nove vrste proizvoda u budućnosti.

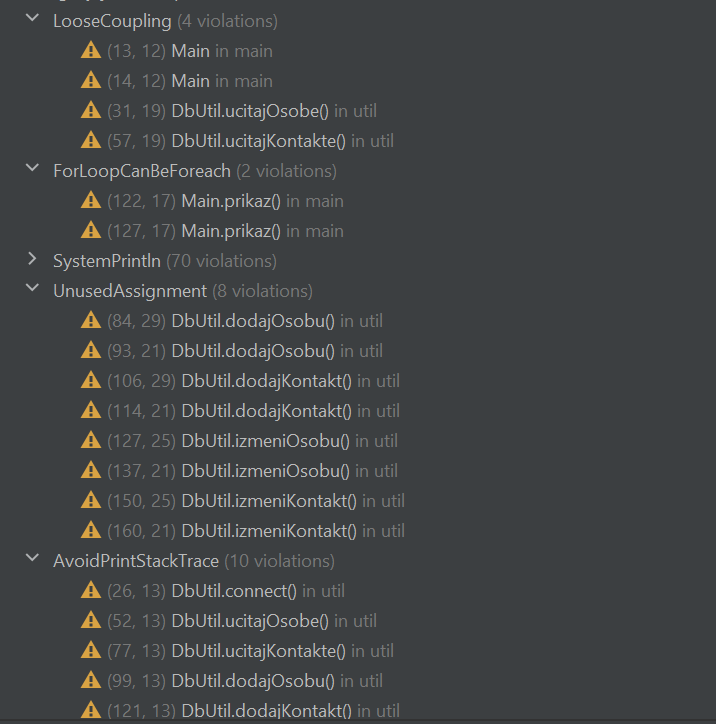
9. Primetiti šta nedostaje

Pregled koda može da podstakne pristrasnost prema razmatranju samo onoga što je pred vama. Ali šta je sa kodom koji nije tamo? Na primer, važno je razmisliti o krajnjim slučajevima, neočekivanim unosima i scenarijima rukovanja greškama koje autor koda možda nije uzeo u obzir.

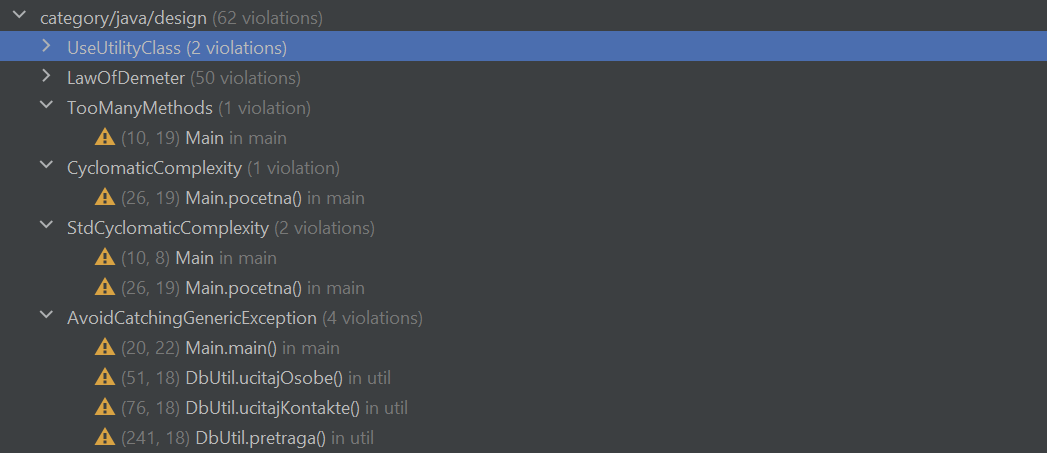
10. Obuhvat testom i kvalitet testa

Pregled koda je jednako važan za testove kao i za kod koji se testira. To je zato što je neispravan test opasniji od nema testa. Polaganje testova omogućava programeru da se oseća sigurno i voljno da novi proizvod pokrene novi proizvod. Ali šta ako jedan od testova prolazi iz pogrešnog razloga ili ne testira ono što bi trebalo da testira?

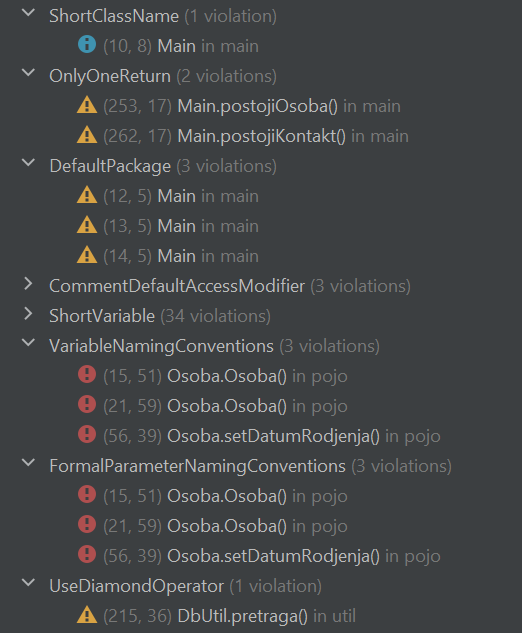
## Primena Programming Mistake Detectora (PMD):

****

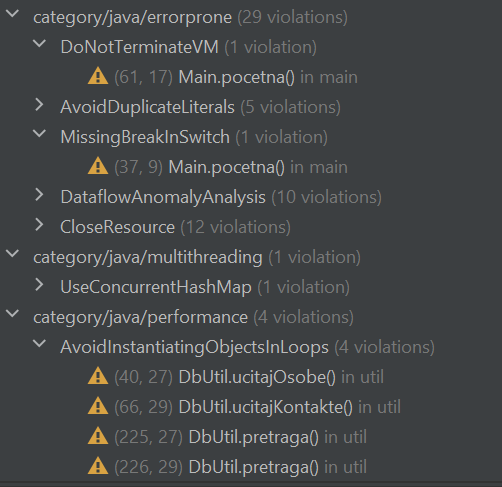
Provera koda iz priloga ovim alatom je donela preporuke u vidi pronadjene slabe povezanosti između komponenata sistema, preporuku o korišćenju for-each petlje umesto obične for petlje,zatim primedbe na nekorišćene metode kao i preporuku o izbegavanju direktnog prikaza steka grešaka.

****

Sledeće primedbe se odnose na kompleksnost Main metode, tačnije da ona sadrži suviše metoda i cikličnu kompleksnost upotrebe istih kao i preporuku o izbegavanju hvatanja generičkih tipova grešaka.

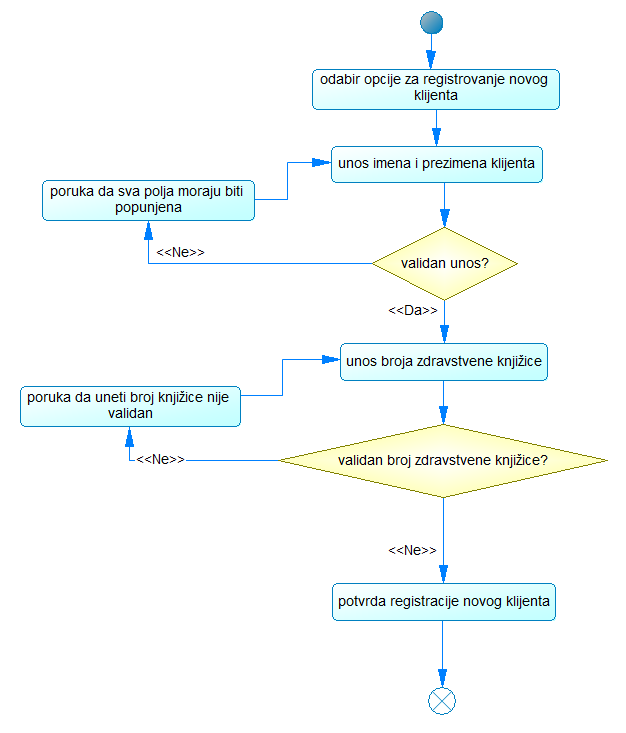


Gore prikazana upozorenja se odnose na primedbe PMD-a o konvencijama naziva klasa te da je naziv klase suviše kratak, da metoda ima samo jednu povratnu vrednost, da se klasa nalazi u defaultnom paketu kao i nazivima parametara metoda i varijabli u programu.



Ovde su takođe prikazana obaveštenja o postojanju metoda za direktno gašenje virtuelne mašine, nedostatku break naredbe u switch operatoru kao i nezatvorenim resursima kao i preporuci da se koristi HashMapa i preporuka o izbegavanju instanciranja objekata u petlji.

## Primena McCabe-ova siklomatske složenosti



McCabe-ova siklomatska složenost

**v(G) = E - N + 2P**

• E - broj grana na grafu programa,

• N - broj čvorova na grafu,

• P - broj nepovezanih komponenti ili se može posmatrati kao broj izlaza iz programa

**v(G) = 11 - 8 + 2 = 5 test slučajeva**

### Test slučaj 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera registracije korisnika | Rev 1 | Autor | Bogdan Janković | Datum | 30.12.2021. |
| Cilj | Provera ispravnosti sistema za registraciju novih korisnika u apoteci | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 3 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da se sva polja moraju popuniti a kasnije se i uneti broj zdravstevene knjižice korisnika (LBO) proverava u bazi Ministarstva zdravlja Srbije, te on kao takav mora biti validan |
|  | Aplikacija je pokrenuta i to od strane farmaceuta koji bira opciju za registraciju novog korisnika apoteke |
|  | Korisnik koristi tastaturu za unos podataka |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Farmaceut je izvršio unos imena i prezimena kao i validan broj knjižice korisnika koji želi da se registruje | Uneti su svi neophodni podaci odnosno ime, prezime i broj zdravstvene knjižice | Nakon svih popunjenih polja i validnih unosa očekuje se portvrda registracije novog klijenta | Sistem je odreagovao očekivano i sa svim validnim unosima, uspešno registrovao korisnika u bazi podataka apoteke. | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Farmaceutu je prikazan prozor sa porukom da je uspešno registrovao novog klijenta apoteke |

### Test slučaj 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera registracije korisnika | Rev 1 | Autor | Bogdan Janković | Datum | 30.12.2021. |
| Cilj | Provera odgovora sistema kada nisu uneti svi parametri (ime ili prezime) | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 5 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da se sva polja moraju popuniti a kasnije se i uneti broj zdravstevene knjižice korisnika (LBO) proverava u bazi Ministarstva zdravlja Srbije, te on kao takav mora biti validan |
|  | Aplikacija je pokrenuta i to od strane farmaceuta koji bira opciju za registraciju novog korisnika apoteke |
|  | Korisnik koristi tastaturu za unos podataka |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Farmaceut je izvršio unos imena ali ne i prezimena korisnika koji želi da se registruje | Unet je samo jedan podatak (ime) | Nakon pritiska na dugme za potvrdu, sistem treba da izbaci poruku o grešci odnosno da je potrebno popuniti sva polja | Sistem je odreagovao očekivano i izbacio prozor sa porukom da je neophodno popuniti sva polja za dalji nastavak registracije. | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Farmaceutu je prikazan prozor sa porukom da je neophodno da popuni sva polja |

### Test slučaj 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera registracije korisnika | Rev 1 | Autor | Bogdan Janković | Datum | 30.12.2021. |
| Cilj | Provera ispravnosti sistema kada knjižica korisnika nije validna | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 7 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da se sva polja moraju popuniti a kasnije se i uneti broj zdravstevene knjižice korisnika (LBO) proverava u bazi Ministarstva zdravlja Srbije, te on kao takav mora biti validan |
|  | Aplikacija je pokrenuta i to od strane farmaceuta koji bira opciju za registraciju novog korisnika apoteke |
|  | Korisnik koristi tastaturu za unos podataka |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Farmaceut je izvršio unos imena i prezimena i prešao na unos broja zdravstvene knjižice ali ona nije validna | Uneti su inicijalno neophodni ime i prezime kao i nevalidni broj zdravstvene knjižice | Nakon svih popunjenih polja i potvrde sistem pri proveri validnosti broja knjižice očekuje se povratna poruka o grešci, odnosno da knjižica nije validna te nema dalje registracije novog klijenta | Sistem je odreagovao očekivano i sa unetim nevalidnim brojem knjižice, uspešno obavestio farmaceuta u vidu iskakajućeg prozora sa porukom. | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Farmaceutu je prikazan prozor sa porukom da je uneti broj zdravstvene knjižice novog klijenta apoteke nevalidn, te je ponovo vraćen na ponovni unos |

### Test slučaj 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera registracije korisnika | Rev 1 | Autor | Bogdan Janković | Datum | 30.12.2021. |
| Cilj | Provera ispravnosti sistema kada je potvrđena registracija a polje za unos broja zdravstvene knjižice prazno | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 5 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da se sva polja moraju popuniti a kasnije se i uneti broj zdravstevene knjižice korisnika (LBO) proverava u bazi Ministarstva zdravlja Srbije, te on kao takav mora biti validan |
|  | Aplikacija je pokrenuta i to od strane farmaceuta koji bira opciju za registraciju novog korisnika apoteke |
|  | Korisnik koristi tastaturu za unos podataka |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Farmaceut je izvršio unos imena i prezimena i uspešno prešao na unos broja zdravstvene knjižice ali to polje pri potvrdi ostavio prazno | Uneti su podaci odnosno ime, prezime ali je broj zdravstvene knjižice izostavljen pri završnoj potvrdi registracije | Nakon popunjenih polja za ime i prezime i ali praznog za broj zdravstvene knjižice očekuje se poruka da nije moguća registracije novog klijenta bez unosa validnog broja zdravstvene knjižice. | Sistem je odreagovao očekivano i sa unetim samo imenom i prezimenom a bez broja knjižice, uspešno obavestio farmaceuta u vidu iskakajućeg prozora sa porukom. | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Farmaceutu je prikazan prozor sa porukom da je neophodno da unese odnosno popuni polje za unos broja zdravstvene knjižice |

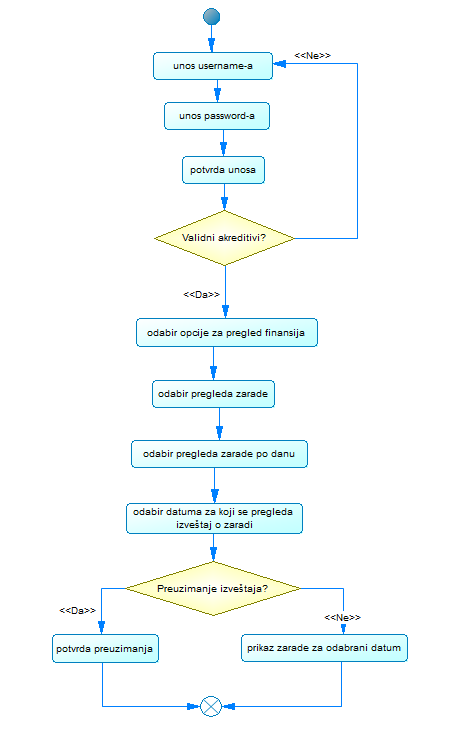
### Test slučaj 5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera registracije korisnika | Rev 1 | Autor | Bogdan Janković | Datum | 30.12.2021. |
| Cilj | Provera ispravnosti sistema kada uneti broj knjižice nije validan više puta | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 8 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da se sva polja moraju popuniti a kasnije se i uneti broj zdravstevene knjižice korisnika (LBO) proverava u bazi Ministarstva zdravlja Srbije, te on kao takav mora biti validan |
|  | Aplikacija je pokrenuta i to od strane farmaceuta koji bira opciju za registraciju novog korisnika apoteke |
|  | Korisnik koristi tastaturu za unos podataka |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Farmaceut je izvršio unos imena i prezimena i prešao na unos broja zdravstvene knjižice ali ona nije validna, i nakon drugog unosa broj knjižice ostaje nevalidan | Uneti su inicijalno neophodni ime i prezime kao i nevalidni broj zdravstvene knjižice isto tako pri ponovno pokušaju odnosno broj knjižice drugi put nevalidan | Nakon svih popunjenih polja i potvrde sistem pri svakoj narednoj proveri validnosti broja knjižice očekuje se povratna poruka o grešci, odnosno da knjižica nije validna te nema dalje registracije novog klijenta | Sistem je odreagovao očekivano i sa drugi put unetim nevalidnim brojem knjižice, uspešno obavestio farmaceuta u vidu iskakajućeg prozora sa porukom. | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Farmaceutu je ponovo prikazan prozor sa porukom da je uneti broj zdravstvene knjižice novog klijenta apoteke nevalidn, te je ponovo vraćen na ponovni unos |

**** McCabe-ova siklomatska složenost

**v(G) = E - N + 2P**

• E - broj grana na grafu programa,

• N - broj čvorova na grafu,

• P - broj nepovezanih komponenti ili se može posmatrati kao broj izlaza iz programa

**v(G) = 14 – 11 + 4 = 7 test slučajeva**

## Slučajevi testiranja na bazi specifikacije

**Testiranje sistema lanca apoteka na bazi specifikacije:**

|  |  |
| --- | --- |
| Glavni koraci  u sistemskom procesu | **1**. Farmaceut bira opciju za logovanje na sistem  **2**. Sistem traži unos njegovog jedinstvenog pristupnog PINa  **3**. Farmaceut unosi ispravan pristupni PIN  **4**. Sistem validira unos odnosno proverava da li je pristupn PIN postojeći  **5**. Sistem prikazuje prozor sa porukom o uspešnom logovanju na sistem apoteke |
| **Mogući problemi na osnovu kojih su izvedeni test slučajevi** | 1. **Farmaceut je potvrdio unos a polje je ostalo prazno** 2. **Uneti PIN nije validan** |

Test pogrešan pristupni pin farmaceuta-Svrha ovog testa je da proveri ispravnost prijavljivanja odnosno logovanja farmaceuta na sistem.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Pogrešan pristupni pin farmaceuta | Rev 1 | Autor | Bogdan Janković | Datum | 30.12.2021. |
| Cilj | Provera ispravnosti sistema za pristupanje farmaceuta sistemu korišćenjem pin-a | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 5 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da se pristupni pin čita iz fajla pristupniPin.txt koji je sastavni deo sistema i u kome se nalazi pristupni pin sa obrnutim redosledom karaktera |
|  | Aplikacija je pokrenuta i to od strane farmaceuta koji bira opciju 1 za nastavak njenog korišćenja |
|  | Korisnik koristi tastaturu za unos podataka |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Farmaceut je izvršio unos pristupnog pin-a i pritisnuo taster Enter radi potvrde unosa | Unet jepristupni pin koji je jedinstven za svakog farmaceuta | Ukoliko se uneti pin ne podudara sa pinom koji se nalazi u fajlu u obrnutom redosledu karaktera od sistema se očekuje da ispiše grešku i ponudi odnosno omogući ponovni unos | Sistem je odreagovao očekivano na pogrešan pristupni pin, ispisao poruku o grešci odnosno pogrešnom pin-u i ponudio ponovni unos istog. | 0 |

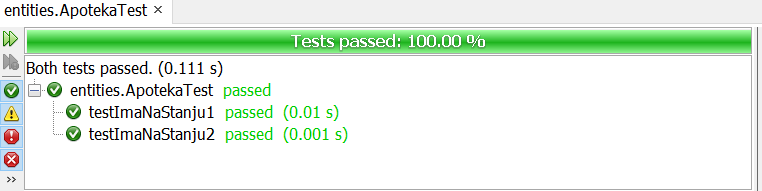
|  |  |
| --- | --- |
| Opis postuslova | |
|  | Korisniku je prikazano da je u pitanju greška u unosu podataka za pristupni pin |
|  | Sistem očekuje od korisnika da unese ispravan podatak radi nastavka upotrebe sistema |

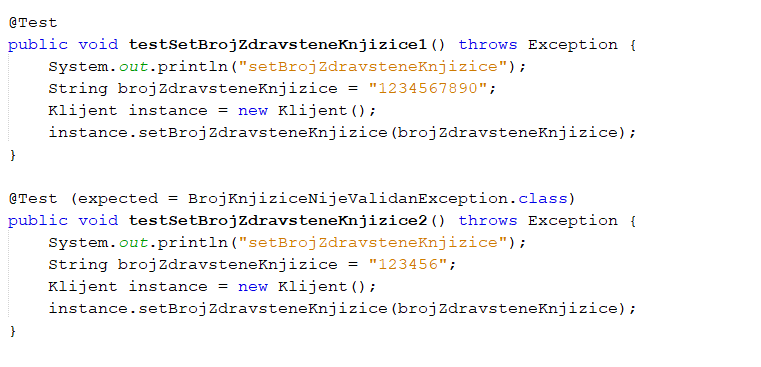
## JUnit testiranje:



Date testovi za metodu koja porverava da li postoji lek u traženoj količini i to tako da proverava podatak iz objekta klase StavkaApoteke i unete tražene količine te da li je ona manja ili jednaka dostupnoj količini leka iz objekta klase StavkaApoteke. Najpre je potrebno da kreiramo objekat klase Lek a zatim i objekat StavkaApoteke kako bi uneli i podatak o količini tog leka koji apoteka trenutno poseduje. Metoda vraća true ili false vrednost te u prvom testu vidimo da je količina dostupna jednaka 4 a da je tražena 6 te se od metode očekuje da vrati false što test i potvrđuje dok je drugi test za slučaj kada leka ima u traženoj količini.

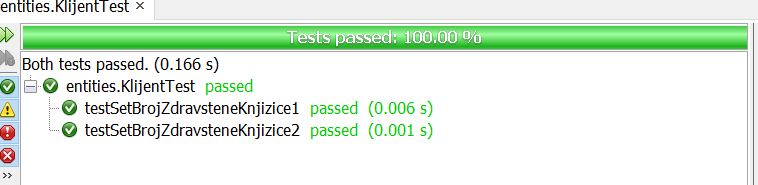
Prikaz rezultata JUnit testova:

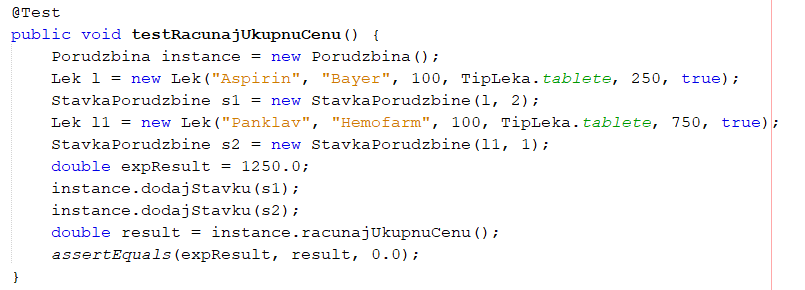




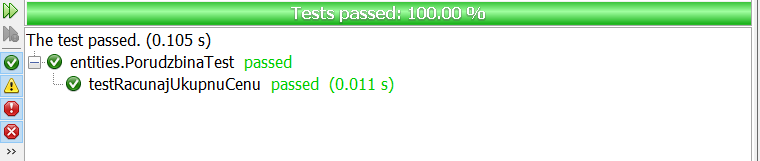
Dati testovi proveravaju setere atributa brojaZdravstveneKnjižice klase Klijent. Kako klijenti imaju mogućnost da se registruju u apoteci, neophodno je proveriti uneti broj tačnije da li je on dužine od 9 do 11 karaktera. Prvi test proverava slučaj korektnog unosa sa dužinom od 10 karaktera dok se kod drugog očekuje izbacivanje izuzetka jer je uneto 6 karaktera, to je kako bi test prošao označeno kao expected.

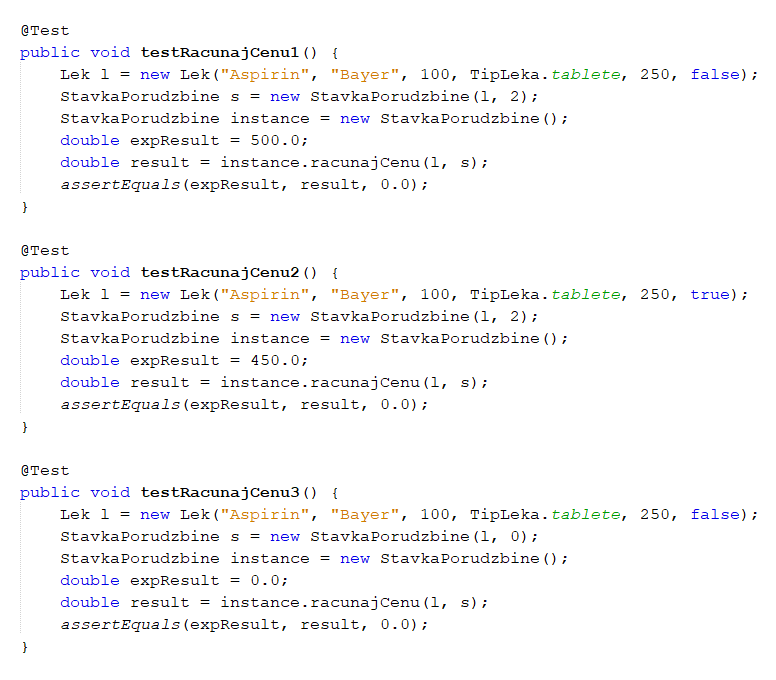
Prikaz rezultata JUnit testova:



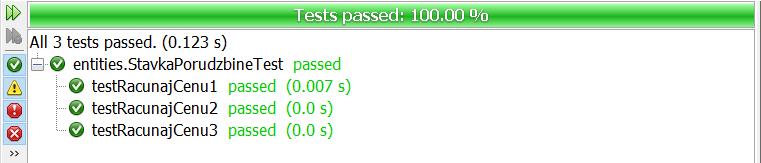


Prikazani test proverava metodu koja računa ukupnu cenu jedne porudžbine i to tako što prolazi kroz sve stavke porudžbine, uzima količinu, a iz samog leka njegovu cenu. Tako u navedenom testu imamo lek sa cenom 250 u količini 2 i jedan lek sa cenom 750 u količini 1 te se očekuje da je ukupna cena porudžbine 1250 što ovaj test i potvrđuje.





Prikazani testovi za metodu koja računa cenu svake stavke posebno te i obračunava popust od 10% ukoliko je lek sa receptom. U prvom testu imamo objekat klase Lek sa cenom od 250 i količinom 2 te se očekuje da je zbirna cena ove stavke jednaka 500 što je testom potvrđeno. U drugom testu se testira obračun popusta i ovde je navedeno da je atribut recept true te se očekuje obračun od 10% po stavci odnosno da sada cena iznosi 450. U trećem testu se proverava cena po stavci ukoliko se za količinu stavke unese 0 te se očekuje da je i cena za te stavke 0, što je testom i potvrđeno.



# Testiranje metodom crne kutije – Black Box

## EKVIVALENTNO PARCELISANJE

Tehnika podele na klase ekvivalencije polazi od ideje da se ulazni podaci mogu razvrstati u reprezentativne klase tako da se za sve pripadnike jedne klase program ponaša na sličan način. Te reprezentativne klase su nazvane klasama ekvivalencije. U ovom slučaju, klase ekvivalencije su međusobno disjunktne i pokrivaju ceo prostor vrednosti ulaza.

Testiranje se obavlja samo za jednu reprezentativnu vrednost ulaza iz svake klase ekvivalencije, zato što se smatra da je to jednako delotvorno kao i testiranje bilo kojom drugom vrednošću iz iste klase. Naime, očekuje se da bi se u svim ovim slučajevima pronašla ista greška u programu. Klase ekvivalencije se formiraju na osnovu svih uslova iz specifikacije koji se odnose na ulaze programa.

Za svaki uslov se posmatraju dve grupe klasa prema zadovoljenosti tog uslova:

• legalne klase koje obuhvataju dozvoljene situacije

• nelegalne klase koje obuhvataju sve ostale situacije

***Razmatranom tehnikom će se testirati deo programa koji ispituje ispravnost unetog telefonskog broja kod unosa dobavljača lekova za lanac apoteka.***

Dozvoljeni format broja je: **+ | 00 (nnn) nn – nnn – nnn** gde n predstavlja numerički simbol, a | „ili“operator. Grupa (nnn) označava državu i pripada skupu (381}, a grupa nn označava telefonskog operatera i pripada skupu {60,61,62,63,64,65,66,69}.

Analizom svakog od navedenih uslova, mogu se dobiti legalne i nelegalne klase ekvivalencije (oznake klasa date su u zagradama):

|  |  |  |
| --- | --- | --- |
| Uslov | Legalne klase | Nelegalne klase |
| Početni simbol | (1) +, 00 | (8) sve ostalo |
| Oznaka države | (2) 381 | (9) sve ostalo |
| Broj simbola u oznaci operatera | (3) 2 | (10) > ili < od 2 |
| Prvi simbol u oznaci operatera | (4) 6 | (11) sve ostalo |
| Drugi simbol u oznaci operatera | (5) 0,1,2,3,4,5,6,7,9 | (12) sve van definsianog skupa odnosno legalne klase |
| Separatori između grupa simbola | (6) – | (13) bilo koji drugi simbol |
| Br. Simb. U poslednje dve grupe | (7) sve numeričke vrednosti | (14) nenumerički simboli |

**Na osnovu date tabele, mogu se napraviti sledeći test primeri:**

+(381) 61 – 284 – 667 zadovoljava sve legalne klase, (1) do (7)

- (381) 62 – 224 – 017 zadovoljava nelegalnu klasu (8)

00(368) 68 – 284 – 667 zadovoljava nelegalne klase (9) i (12)

+(381) 611 – 284 – 667 zadovoljava nelegalnu klasu (10)

+(381) 61 – 284 – 667 zadovoljava nelegalnu klasu (11)

+(381) 61/284 / 667 zadovoljava nelegalnu klasu (13)

+(381) 61 – 24a – 67 zadovoljava nelegalnu klasu (14)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Validnost unetog kontakta | Rev 1 | Autor | Bogdan Janković | Datum | 03.01.2022. |
| Cilj | Provera ispravnosti sistema za validaciju pri registraciji dobavljača lanca apoteke | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 5 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Sistem prihvata samo predefinisani odnosno dozvoljeni format broja koji je u sledećem formatu: **+ | 00 (nnn) nn – nnn – nnn** |
|  | Aplikacija je pokrenuta i to od strane vlasnika odnosno admina koji bira opciju za registraciju dobavljača lekova |
|  | Korisnik koristi tastaturu za unos podataka |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je izvršio unos kontakt broja odnosno broja telefona i pritisnuo taster Enter radi potvrde unosa | Unet je broj +(381) 611 – 284 – 667 koji zadovoljava jednu od definisanih nelegalnih klasa | Ukoliko se uneti broj ne podudara sa definisanim formatom od sistema se očekuje da ispiše grešku i ponudi odnosno omogući ponovni unos | Sistem je odreagovao očekivano na pogrešan format broja, ispisao poruku o grešci odnosno pogrešnom formatu broja koji sistem ne podržava i ponudio ponovni unos istog. | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Korisniku je prikazano da je u pitanju greška pri unosu podataka za kontakt |
|  | Sistem prikazuje format broja koji podržava |
|  | Sistem očekuje od korisnika da unese ispravan podatak radi nastavka upotrebe sistema |

## ANALIZA GRANIČNIH VREDNOSTI

U radu sa klasama ekvivalencije često se dešavaju greške zbog neadekvatnog definisanja njihovih granica. Na primer, programer može umesto oznake < pogrešno da iskoristi oznaku <. Stoga je uvedena analiza graničnih vrednosti koja se primenjuje pri generisanju test primera u cilju potpunije podele opsega ulaznih podataka. Analiza graničnih vrednosti će se koristiti tako što se test primeri biraju na granicama različitih klasa ekvivalencije. Pri tome se vodi računa, ne samo o ulaznim uslovima, već i o izlaznim podacima iz programa.

***Analiza graničnih vrednosti će biti korišćena na primeru pisanja formata datuma u godini kod registracije farmaceuta u sistemu i kod registracije novih lekova lanca apoteke gde se ovakav format traži:* dd.mm*, gde dd predstavlja dan, a mm mesec u godini***.

Kao što je poznato, za mesece januar, mart, maj, jul, avgust, oktobar i decembar, dd pripada opsegu 1 do 31, za februar opsegu 1 do 29, a za ostale mesece opsegu 1 do 30. Slično ovome, mm pripada opsegu 1 do 12.

+ Ovo su legalne klase u primeru i na osnovu njih mogu se napisati test primeri sa graničnim vrednostima: **01.01., 31.12., 23.09, 31.08., 29.02.**

﻿- Nelegalne klase predstavljaju vrednosti van navedenih opsega. Za njih se mogu napraviti sledeći test primeri sa graničnim vrednostima: **00.05., 32.05. , 06.13., 06.00.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Validnost unetog datuma | Rev 1 | Autor | Bogdan Janković | Datum | 03.01.2022. |
| Cilj | Provera ispravnosti sistema za validaciju pri unosu datuma u sistem | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 5 min |

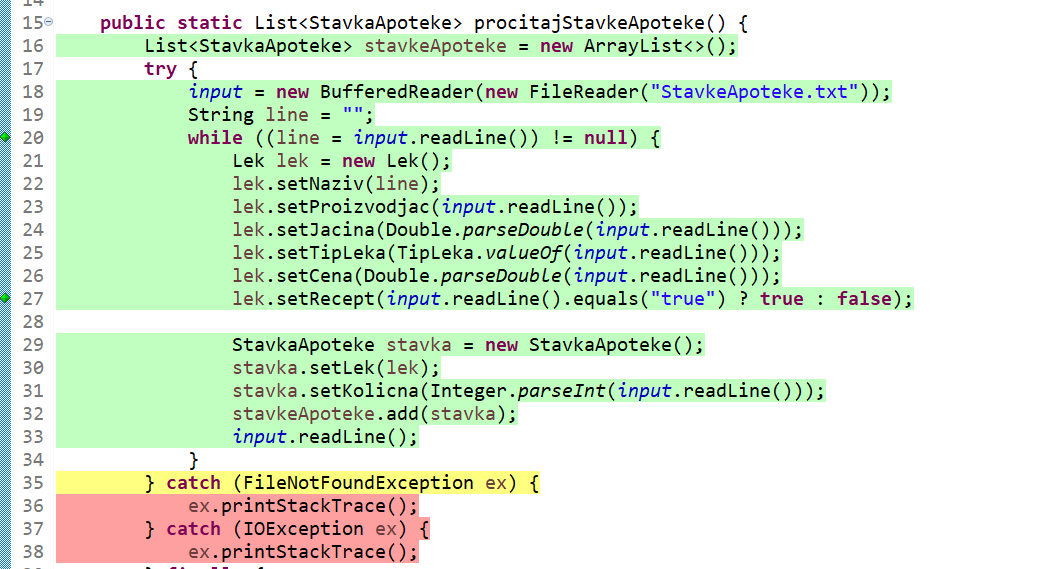
|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Sistem prihvata samo predefinisani odnosno dozvoljeni format datuma pri registaciji dozvole lekova koji je u sledećem formatu: **dd.mm** |
|  | Aplikacija je pokrenuta i to od strane vlasnika odnosno admina koji bira opciju za registraciju lekova gde se traži unos datuma kada je lek dobio dozvolu za upotrebu u tekućoj godini |
|  | Korisnik koristi tastaturu za unos podataka |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je izvršio unos datuma odnosno broja kao vrednosti datuma za dan i mesec i pritisnuo taster Enter radi potvrde unosa | Unet je datum 32.05.koji predstavlja vrednosti van dozvoljenog odnosno opsega koji je definisan u sistemu | Ukoliko se uneti datum ne podudara sa definisanim opseg koji sistem validira od sistema se očekuje da ispiše grešku i ponudi odnosno omogući ponovni unos | Sistem je odreagovao očekivano na vrednost datuma van definisanog opsega, ispisao poruku o grešci odnosno pogrešnom formatu broja koji sistem ne podržava i ponudio ponovni unos istog. | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Korisniku je prikazano da je u pitanju greška pri unosu vrednosti za datum |
|  | Sistem očekuje od korisnika da unese ispravan podatak radi nastavka upotrebe sistema |

# Testiranje metodom bele kutije – White Box

## Pokrivenost naredbi

****

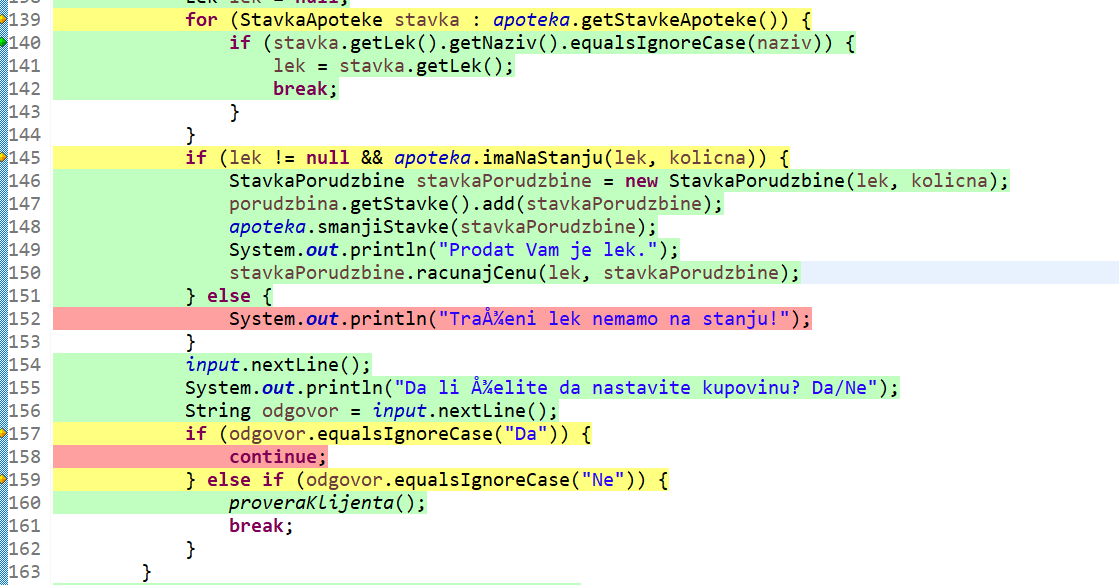
Na prikazanoj slici vidimo pokrivenost naredbi koje su neophodne za čitanje podataka, kreiranje objekata date klase i njihovo setovanje u same atribute objekata. Može videti da su sve naredbe u try bloku pokrivene ovim testom dok, kako nije bilo očekivanih izuzetaka, nije došlo ni do hvatanja odnosno pokrića istih u catch bloku.

Pri testiranju programa, test primeri se prave tako da se svaki iskaz u programu bar jednom izvrši, jer se drugačije ne može znati da li u njemu postoji greška. Međutim, ako se iskaz ispravno izvrši za jednu ulaznu vrednost, to ne znači da će se dobro izvršiti i za neku drugu ulaznu vrednost. Ovo je glavni nedostata razmatrane tehnike.

## Pokrivenost odluka (grana)

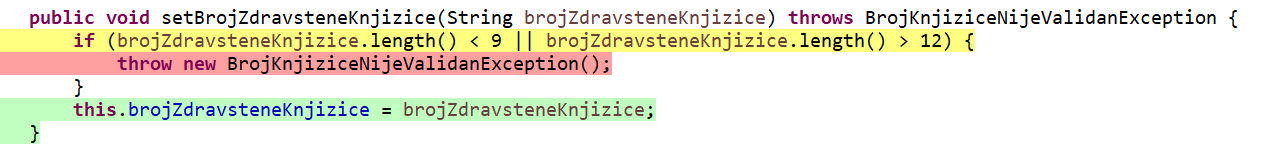
Tehnika pokrivanja odluka podrazumeva projektovanje test primera tako da se svaka od različitih grana uslovnog iskaza izvrši bar jednom. Ovo je ,jači“ metod od pokrivanja iskaza zato što pokrivanje odluka garantuje pokrivanje iskaza.

Odlučivanje se vrši na osnovu iskaza: if A then B else C Ako je uslov A ispunjen, pokrivaju se svi iskazi B koji se nalaze u odgovarajućoj grani, dok se posebno mora napisati test kada uslov A nije ispunjen, kako bi se pokrili i iskazi C.

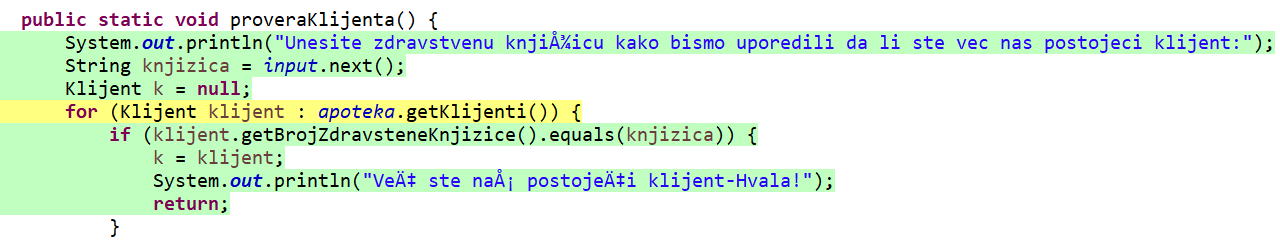


## Pokrivenost prostih uslova

U tehnici pokrivanja uslova, test primeri se pišu tako da svaka elementarna komponenta nekog složenog uslova uzima vrednost iz dozvoljenog i nedozvoljenog skupa vrednosti. Elementarni uslovi se posmatraju potpuno nezavisno jedan od drugog. Ovaj metod ne garantuje pokrivanje svih odluka, pa samim tim ni pokrivanje svih iskaza u programu.



## Pokrivenost petlji



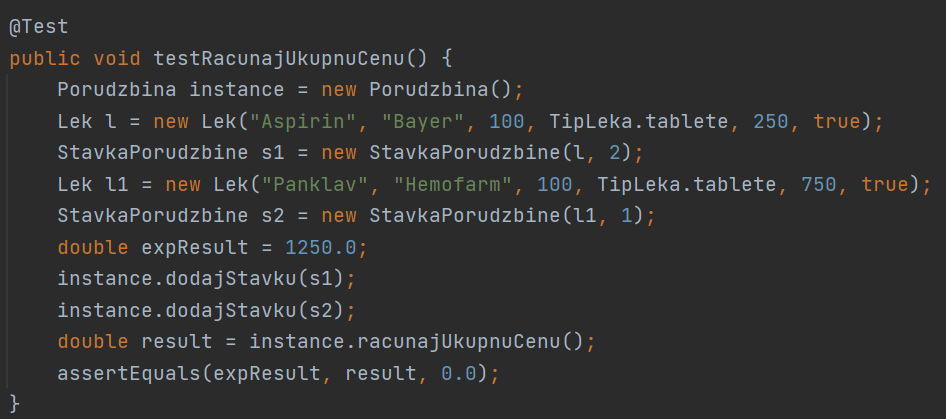
Iz datog primera vidimo da je pokrivenost petlje u metodi takva da je petlja pretraživala listu klijenata sve dok nije naišla na podudaranje unetog broja knjižice sa postojećim brojem kao atributom objekta, te da nije prošla kroz celu listu odnosno izvršila sve iteracije.

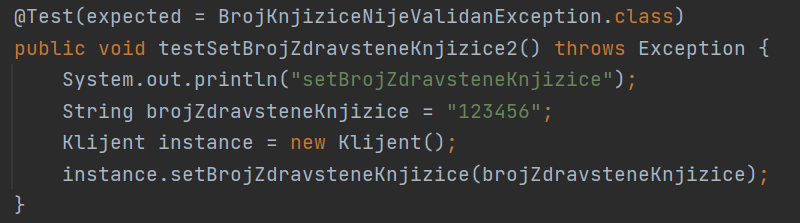
# MOCKING TESTIRANJE

Mocking pokušava da na lak način reši kreiranje lažnih objekata (klasa) koji pomažu procesu jediničnog testiranja. Tako lažni objekti su dizajnirani da „zavaraju“ Java objekt (koji se jedinično testira) tako da se ima utisak da on komunicira sa drugim stvarnim objektima.

Primeri testova sa kreiranjem lažnih objekata radi samog testiranja aplikacije:







Lažni objekti omogućavaju da postavite kontrolisano okruženje sa strogo definisanim determinističkim ponašanjem svih objekata koji su uključeni u objekat koja se testira.

# Integraciono testiranje

Integraciono testiranje se radi posle jediničnog testiranja (“Unit testing”). Nakon što su sve individualne jedinice (metode) napravljene i testirane, počinjemo sa kombinovanjem ovih jedinično-iztestiranih modula i tako radimo integraciono testiranje. Integraciono testiranje se obavlja u toku razvoja softvera a ne na kraju razvoja softvera. Koristi se mogu dobiti od ovakve pažljive integacije su lakša dijagnoza defekata, manje defekata, manje vremena do prvog operativnog proizvoda, kraće ukupno vreme razvoja proizvoda, veće šanse da se projekat uradi uspešno i na samom kraju pouzdanije procene rokova i naravno bolji kvalitet programa.

Postupak integracionog testiranja:

1. individualni moduli se prvo testiraju u izolaciji od ostalih modula odnosno jedinično testiraju
2. integrisati tj. kombinovati jedinično-iztestirane module, jedan po jedan, u složenije celine
3. testirati sve dok se ne integrišu svi moduli, i to uključujući kombinovano ponašanje modula
4. testirati svaki ovaj složeni modul kao jednu jedinicu
5. glavni cilj da se testiraju interfejsi između jedinica/modula i da se izvrši validacija da li su zahtevi implementirani korektno

Koraci koji su potrebni kod integracionog testiranja, tj. izvršavanje integracionih testova:

• proučiti arhitekturu aplikacije

Apoteka poseduje bazu koja cuva sve podatke o lekovima odnosno stavkama apoteke, klijentima i prodaji. Njom se upravlja posebnim softverskim upravljackim sistemom za transakcionu obradu sto zajedno **predstavlja arhitekturu sa skladistem podataka** gde ova baza predstavlja centralni repozitorijum.

• identifikovati module

Dva osnovna modula koja čine ovaj sistem su modul farmaceuta i modul korisnika koji shodno svom načinu pristupa sistemu imaju i drugačije funkcionalnosti.

• identifikovati šta svaki modul radi

Modul Farmaceut: omogućava autentifikaciju farmaceuta koji pristupa sistemu i to unošenjem jedinstvenog pin-a, zatim pregled svih lekova dostupnih u apoteci kao i njihove trenutne dostupne količine i pregled svih registrovanih klijenata u sistemu apoteke.

Modul Klijent: pruža korisnicima mogućnost kupovine odabranih lekova i željenoj količini naravno ako je ona dostupna kao i registraciju pri istoj.

• identifikovati kako se podaci razmenjuju tj. prenose između pojedinih modula

• identifikovati koji su ulazi i koji izlazi cele aplikacije

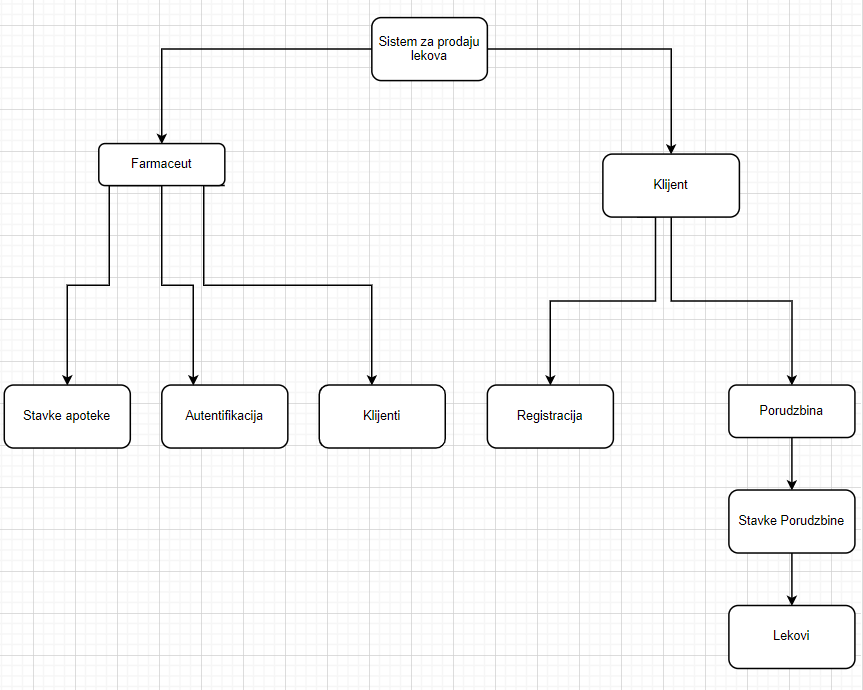
Ulazi: odabrane opcije u vidu broja za kretanje kroz aplikaciju, pristupni pin farmaceuta, naziv leka koji se kupuje, količina traženog leka, broj zdravstvene knjižice klijenta (i ime i prezime pri registraciji).

Izlaz: liste lekova, klijenata, račun sa svim stavkama i ukupnom cenom, obrađene runtime greške

• identifikovati i napraviti test-slučajeve

Primer integracionog testiranja sa test slučajevima je u test paketu priloženog projekta, u ***klasi IntegracionoTestiranje***.

Slika prikaza putanje integracionog testiranja sistema koristeći top-down tehniku:



## Izveštaji o realizovanom testiranju

**Primer jednog testa:**

**Naslov:** Registrovanje korisnika

**• Actors:** Farmaceut

**• Uslov:** Sistem je pokrenu i farmaceut je uspešno ulogovan na sistem

**1**. Farmaceut bira opciju za registrovanje novog korisnika

**2**. Sistem prvo traži korisnikovo ime i prezime

**3**. Farmaceut unosi ime i prezime korisnika koji se registruje

**4**. Sistem validira unos odnosno proverava da polja nisu prazna

**5**. Sistem prikazuje prozor sa poljem za unos broja zdravstvene knjižice osiguranika

**6**. Sistem validira unos broja zdravstvene knjižice

**7**. Sistem prikazuje potvrdnu poruku da je novi klijent apoteke uspešno registrovan

**Alternative:**

• **4a:** nisu sva polja popunjena

◦ **4a1:** Sistem prikazuje poruku da sva polja treba da budu popunjena

• **6a**: Validacija je nekorektna i broj pokušaja < 3

◦ **6a.1:** Vrati se na korak 5

• **6b**: Validacija knjižice korisnika nekorektna i broj pokušaja >= 3

◦ **6b.1**: Sistem prikazuje poruku o grešci za uneti broj zdravstvene knjižice

### Izveštaj o testiranju korišćenjem obrasca za izveštaj o grešci

**1. Lokacija:** Modul za registraciju novih korisnika apoteke.

**2. Vreme:** Faza razvoja u kojoj je greška otkrivena je faza testiranja.

**3. Simptom:** Pregledom je utvrđeno da je korisnik upisan u bazu apoteke iako je uneti broj njegove zdravstvene knjižice nevalidan (neohodna dužina od tačno 10 cifara).

**4. Konačni rezultat:** Otkazi koje je greška izazvala su nekonzistencija baze podataka.

**5. Mehanizant**: Nakon otkrivene greške, validacija unosa se vrši dvostruko, odnosno i na frontend-u kao i pri perzistenciji pre upisa u bazu.

**6. Uzrok**: Farmaceut je uneo nevalidan broj zdravstvene knjižice.

**7. Ozbiljnost**: Mogućnost otkaza celokupnog modula za registraciju novih korisnika apoteke.

**8. Cena**: Potreban rad za pronalaženje i ispravljanje se može meriti u satima i iznosi oko 8 sati angažovanja jednog programera, dok bi ovo bilo značajno manje da je greška otkrivena u ranijim fazama razvoja sosftvera.

# ODRŽAVANJE, DALJI RAZVOJ I KONTINUALNI PROCES POBOLJŠANJA SOFTVERA

## Model estimacije troškova prilikom održavanja odabrane aplikacije

***Vođa projekta*** (Team leader) - Radiće na upravljanju svih aktivnosti na projektu, dobijaće izveštaje od strane softverskih inženjera i administratora sistema. Predviđeno radno vreme na projektu je 160 radnih sati. Cena rada po satu iznosi 200 dinara.

160 \* 200 = 32 000din

***Administrator sistema*** – Podešavanje softvera i hardvera u serverskom delu sistema, unos novih korisnika, lekova, famraceuta kao i povezivanje korisnika sa karticama lojalnosti kao I celokupnog sistema sa RFZO sistemom, su aktivnosti koje će raditi administrator sistema. Predviđeno radno vreme na projektu je 754 radna sata. Cena rada po satu iznosi 180 dinara.

754 \* 180 = 135 720din

***Softverski inženjer 1*** – Biće zadužen za vezu starog sistema i novog koji se razvija, na kreiranju modela baze podataka i na planiranju modula e-recept. Predviđeno radno vreme na projektu je 400 radnih sati. Cena rada po satu iznosi 200 dinara.

400 \* 200= 80 000din

***Softverski inženjer*** ***2***– Biće zadužen za grafički korisnički interfejs, na kreiranju dokumentacije i na planiranju modula za integraciju za dostavu. Predviđeno radno vreme na projektu je 400 radnih sati. Cena rada po satu iznosi 200 dinara.

400 \* 200= 80 000din

***Tester*** – Angažovanje prilikom testiranja sistema, njegovih osnovih funkcija i novih modula. Testiraće i bazu podataka sa velikim brojem unetih lekova kao i grafički interfejs sistema. Predviđeno radno vreme na projektu je 48 radnih sati. Cena rada po satu iznosi 100 dinara.

48 \* 100= 4800din

Ostali resursi:

• PC-Server – 130.000 din

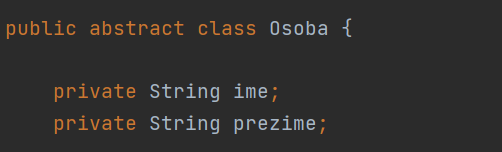
• PC 1 – 80.000 din

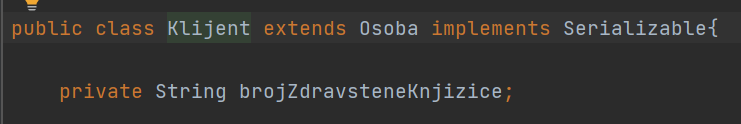
• PC 2 – 70.000 din

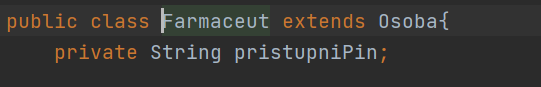
• PC 3 – 60.000 din

## Pristup za postizanje višekratne upotrebljivosti

Primer rejuzabilnosti klasa primenom nasleđivanja:







Generalizacija je tehnika koja se koristi da bi se na jednostavan način rešavale složene

situacije. Uopštene klase sadrže atribute i operacije (metode) koje imaju druge klase, koje

su onda podklase uopštene klase. Podklase nasleđuju atribute i metode svojih klasa-roditelja

(nad klasa) te se na taj način pojednostavljuje opis klasa sistema, jer su česti slučajevi kada

više klasa imaju iste atribute i metode (operacije).

Generalizacija omogućava da se zajedničke informacije (atributi i metode) definišu i

održavaju samo na jednom mestu (klasi roditelju). Kada se treba izvršiti neka promena, onda

se promena obavi samo na jednom mestu, umesto da se vrši u raznim klasama. U Javi

(i drugim OO jezicima), generalizacija se primenjuje mehanizmom nasleđivanja koji je deo

jezika.

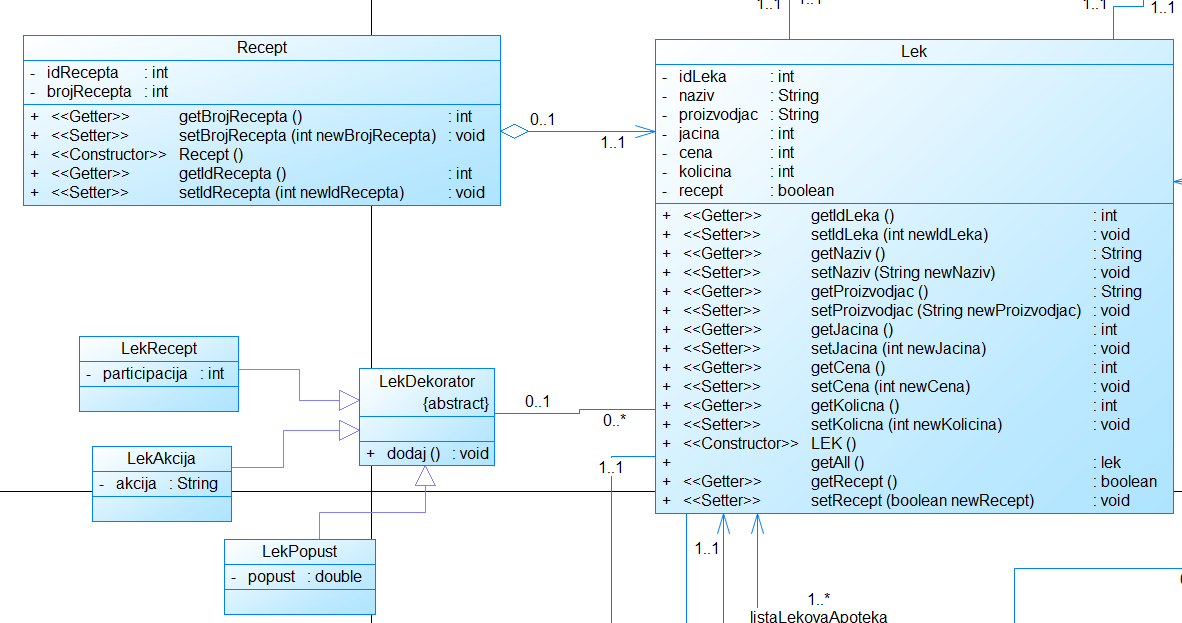
Primer primene design pattern-a:

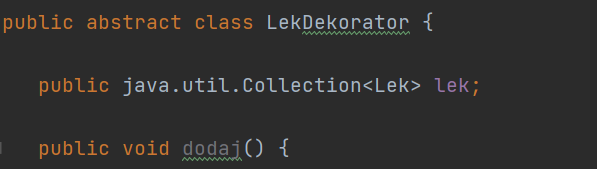
***Decorator šablon*** dozvoljava dodavanje nove funkcionalnosti postojećeg objekta bez menjanja njegove structure.

Decorator šablon dozvoljava dodavanje nove funkcionalnosti postojećeg objekta bez menjanja njegove strukture.Ovaj šablon kreira Decorator klasu omotač originalne klase i obezbeđuje dodatnu funkcionalnost bez promena potpisa metoda klase

Donosi dinamičku dodatnu odgovornost nekom objektu. Decorator objekti obezbeđuju fleksibilnu alternativu korišćenju potklasa za proširenje funkcionalnosti.

Nekada želimo da dodamo funkcionalnost samo nekim objektima, a ne celoj klasi (npr. GUI). Jedan način za ovo je primena nasleđivanja. To je nefleksibilno i statički, jer svi objekti jedne klase to dobijaju. Drugi fleksibilniji način je da se objekat omota drugom frugim (Decorator) koja dodaje potrebnu funkcionalnost.





Decorator šablon odnosno njegova primena u ovom projektu ima više prednosti:

1. Veća fleksibilnost nego statičko nasleđivanje. Mogu se dodavati i uklanjati u vreme izvršenja. Svojstva se mogu i dva puta dodavati inkrementalno kao i dodavanje odgovornosti primenom Decorator objekata u fazi izvršenja.
2. Decorator i njegova komponenta nisu identični, te ne zavisie od identiteta objekta.
3. Mnogo malih objekata. Razlikujiu se samo sa malim inkrementalnim dodacima.

# ZAKLJUČAK

Centar svakog poslovanja je zadovoljan korisnik. Zato je potrebno obezbediti u samom startu proizvod visokog kvaliteta koji će garantovati njegovu vrednost i pouzdanost. Kreiranje softvera se može posmatrati i kao poslovni dogovor. Softver treba da iznese klijentske potrebe bez greške. Testiranje softvera nudi dokaz za softver ispunjava poslovne zahteve klijenta.

Iako testiranje može dovesti do povećanja planiranih troškova, kašnjenja… ovo je svakako zanemarljivo u odnosu na vrednost koja se dobija od samog testiranja. Kako su bagovi popravljeni pre objave softvera, troškovi popravke bagova i održavanje softvera su svedeni na minimum.

U razvoju softvera, sve se mora duplo proveriti pre objave proizvoda. Da bi ostali u poslu kao IT kompanija, imperativ je imati tim softver testera. Proizvodi sa kritičnim bagovima, mogu prouzrokovati velike gubitke u poslovanju. Zato danas, svaka ozbiljna IT kompanija investira u svoj tim testera koji imaju zadatak da do toga ne dođe. Otkrivaju greške još u fazi razvoja, kako bi softverske performanse bile adekvatne i kako bi korisnik bio zadovoljan.

# LITERATURA

1. LAMS Nastavni materijali predmeta SE321- Obezbeđenje kvaliteta, testiranje i održavanje

Softvera - Univerzitet Metropolitan Beograd

1. <https://interrupt.memfault.com/blog/defensive-and-offensive-programming>
2. <https://dtc.rs/sr/testiranje-softvera>
3. [https://vtsnis.edu.rs/wp-content/plugins/vts predmeti/uploads/14%20Testiranje%20softvera%20i%20upravljanje%20kvalitetom%2016.03.2020.pdf](https://vtsnis.edu.rs/wp-content/plugins/vts%20predmeti/uploads/14%20Testiranje%20softvera%20i%20upravljanje%20kvalitetom%2016.03.2020.pdf)
4. <https://docplayer.rs/116320652-Testiranje-softvera.html>