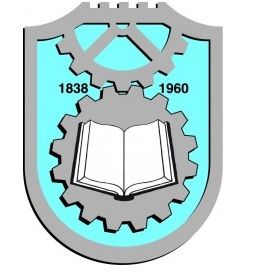
**UNIVERZITET U KRAGUJEVCU**

**FAKULTET INZENJERSKIH NAUKA**



**DOKUMENTACIJA ZA PROJEKTNI ZADATAK:**

**InfoKC**

Student: Profesor:

Anđelija Živković Dr Aleksandar Peulić

*Kragujevac, 2017.*

Sadrža

[1. Uvod 2](#_Toc483473091)

[2. Projektovanje programa 3](#_Toc483473092)

[1.Class Diagram 3](#_Toc483473093)

[2. Use Case Diagram 4](#_Toc483473094)

[3. Princip rada 5](#_Toc483473095)

[4. Implementacija koda 9](#_Toc483473096)

# Uvod

Jedna od bitnih stavki svakog projketa je da bude korišćen, efikasan i da bude od pomoći u današnjem svetu tehnologije. Vrlo dobro smo upoznati kako funkcioniše svet oko nas i baš zato imamo za cilj da sve što možemo promenimo na bolje. Svakome od nas jasan je problem bezgraničnih obaveza, čekanja u redovima, nepostavljenih informacija koje bi nam pomogle u lakšem snalaženju. Sve to, kada se nakupi na jednu gomilju problemima nema kraja kao ni izgubljenom vremenu. Stoga, u cilju nam je da realizujemo projekat za koji se nadamo da će biti od korist i pomoći u stvaranju reda i efikasnom izvršavanju poslova.

Uzimajući u obzir gore navedene probleme, osmišljen je projkat „InfoKC“, tj. informacioni Klinički centar. Ispred ordinacija uvek možemo videti velike redove u kojima stoje ljudi koji ne znaju ni da li će biti primljeni, po hodnicima lutaju i tražemesto gde treba da se obrate i još posebno kada bi se za sve te stvari raspitivali kod zaposlenih to bi njima loše uticalo na rad. Sve u svemu, ređaju se minusevi koji koče u daljem radu.

Inspirisani ovim problemima osmišljen je mali program koji korisnicima daje obaveštenje gde da idu, kome da se obrate, da li treba da zakazuju svoj pregled ili direktno da idu na isti. Sve to se radi pomoću parametara koji korisnici unose sa dobijenog uputa koji im je lekar opšte prakse prethodno dodelio. Prednost ovog projekta je to što se lako može snaći bilo koji korisnik, dovoljno je samo upisati brojeve dobijene na uputu i pažljivo pratiti korake koji im program zadaje.

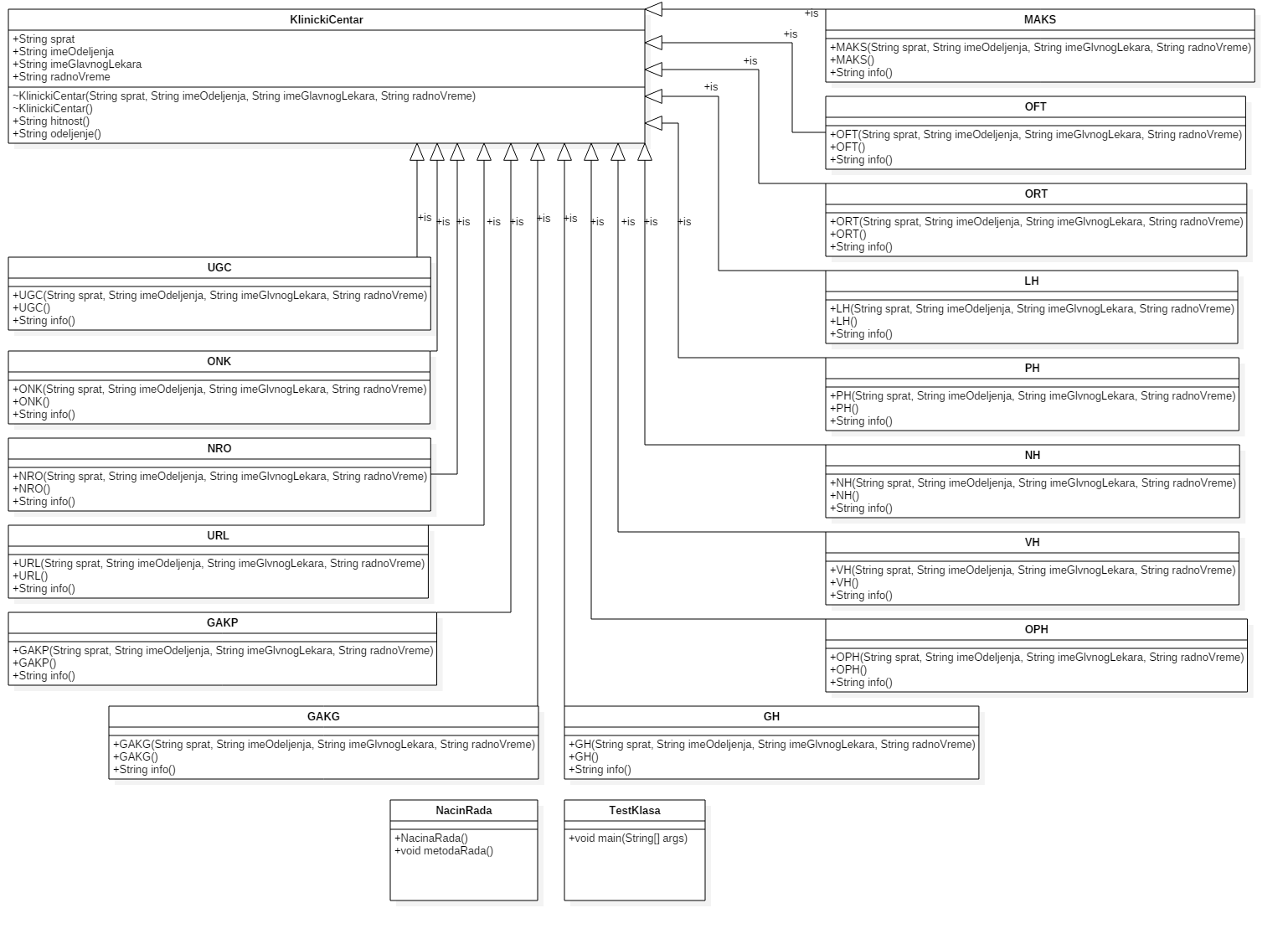
Za korišćenje ovog programa potrebno je manje od minuta, dakle brzo i lako možemo doći do informacija. Pritom ne dolazi do trošenja vremena za raspitivanje, ne bi se čekalo u redovima jer ovaj program omogućava raspored i primarnost nekog problema za koji je korisnik došao u Klinički centar.

# 2. Projektovanje programa

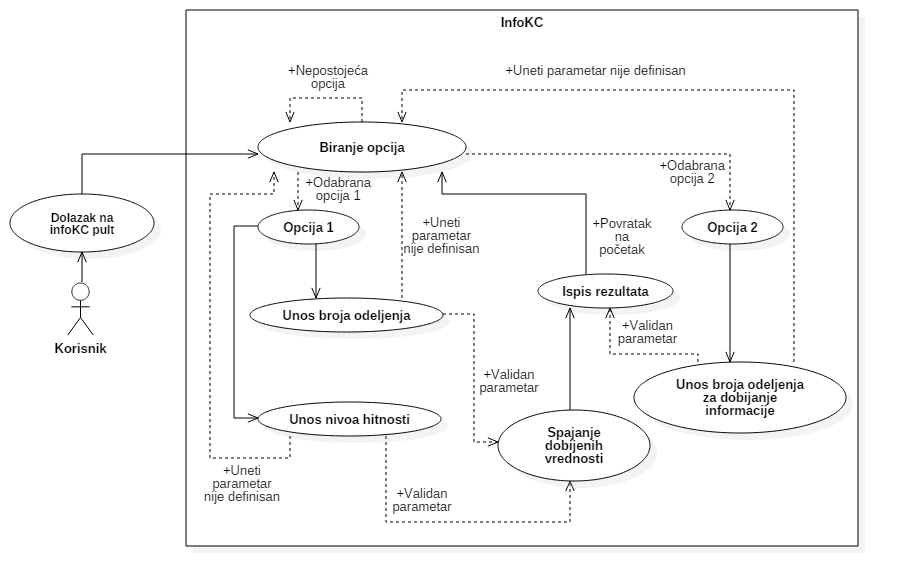
## Class Diagram

Dijagram klasa prikazuje skup klasa, interfejsa povezanih relacijama. To je graf obrazavan od temena (stvari) povezanih granama (relacijama):

1. Stvari- klase, interfejsi, tipovi, izuzetci, šabloni, saradnje, paketi,...
2. Relacije- zavisnost, generalizacija, asocijacija, realizacija

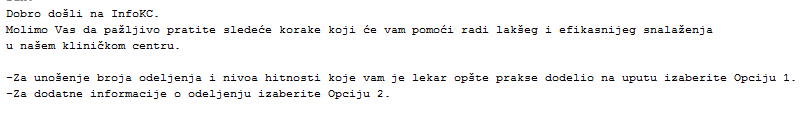
(4.1. Dijagram klasa)

## 2. Use Case Diagram



# 3. Princip rada

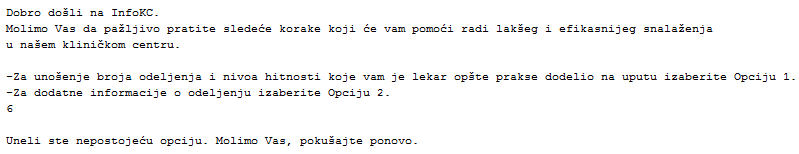
Na samom početku korisnik dobija smernice koje mu pomažu da se lakše usmeri na problem zbog kojeg je i došao. Kao što smo i prethodno naveli, korišćenje je veoma jednostavno. Dovoljno je unositi samo brojeve koje ste već prethodno dobili i shodno tome dobijate informacije koje ste želeli.

(Slika 2.1. Tekst koji se pojavljuje priliko pokretanja „infoKC“ projekta.)

Sada je najbitinije da odaberemo opciju zbog koje smo došli na ovaj info-pult:

* **Opcija 1** je namenjena prvenstveno za preglede na koje korisnik treba da ode. Već na uputu, korisnik ima zabeleženo ime ordinacije koji mu je lekar opšte prakse dodelio zbog pregleda i broj hitnosti njegovog pregleda. Taj broj hitnosti ustvari predstavlja primarnost pacijenta, da li treba da zakazuje svoj pregled ukoliko je hitnost mala ili srednja, ili odmah treba da se obrati lekaru i bude pregledan ako je hitnost velika.
* **Opcija 2** je namenjena za dodatne informacije. Ukoliko pacijent ili korisnik nije upućen gde se nalazi ordinacija, koji lekar je zadužen za preglede na izabranoj ordinaciji i takodje koje je radno vreme. To su isto veoma bitne stvari koje nam mogu sačuvati vreme a ujedno nam dati i korisne informacije.

Dovoljno je samo uneti broj opcije (1 ili 2) i tako dolazimo do sledećih pitanja koja omogućavaju konačni dolazak do informacija. Ukoliko korisnik slučajno unese pogrešan broj koji nije predviđen za opcije pojavljuje se obaveštenje o lošem unosu i početni tekst koji omogućava ponovni unos opcija.



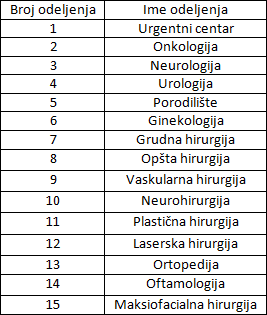
(Slika 2.2. Unos pogrešnog broja, dobija se obaveštenje o lošem unosu)

Opcija 1

Ovu opciju biramo unošenjem broja 1. Opcija 1 sadrži dva ključna pitanja radi lakšeg usmeravanja korisnika. Pokretanje ove opcije otvara prvo pitanje za koje unosimo broj odeljenja koji smo prethodno dobili na uputu. Drugo pitanje je koliko je hitan naš pregled. Taj broj je takođe dat na uputu.

Broj odeljenja

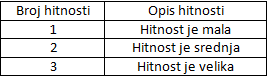
Svako odeljenje ima svoj jedinstveni broj koji ga opisuje. Taj jedinstveni broj je isti i u Opciji 1 i u Opciji 2. Na osnovu unosa tog broja korisnik može pročitati o kom odeljenju je reč. Broj i ime odeljenja dati su u sledećoj tabeli:



(Slika 2.3. Brojevi odeljenja sa imenima)

Broj hitnosti

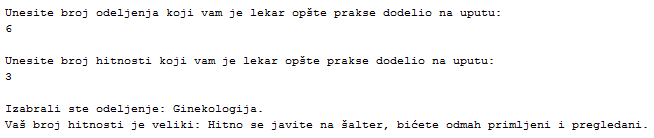
* Mali stepen hitnosti predstavlja ne toliko urgentni problem za koji pacijent može da zakaže termin kako bi bio pregledan. Ova opcija se bira unošenjem broja 1.
* Srednji stepen hitnosti predstavlja hitnost koja je malo jača od prethodne, takođe se pacijent šalje na šalter zakazivanja ali njihov termin pregleda biće pre realizovan nego pregled za mali stepen hitnosti.Ova opcija se bira unošenjem broja 2.
* Visoki stepen hitnosti predstavalja hitnost za koju je potrebno da se što pre pacijent javi na šalter i što pre bude primljen na pregled. Ova opcija se bira unošenjem broja 3.



(Slika 2.4. Broj hitnosti sa kratkim opisom)

Dakle, kada želimo da pokrenemo Opciju 1 (tj. opciju za unos brojeva datih od strane lekara opšte prakse) i da zakažemo termin pregleda, dobijaju se dva pitanja koja zahtevaju unos brojeva koji se nalaze na uputu. Prvi broj koji se nalazi na uputu je broj odeljenja i samim tim prvo pitanje Opcije 1 zahteva unos baš tog prvog broja. Drugi broj je broj hitnosti tako da drugo postavljeno pitanje je vezano za unošenje broja hitnosti. Na kraju dobije se sklopljena rečenica koja korisnika obaveštava koje odeljenje je uneo , koja hitnost mu je dodeljena i na osnovu te hitnosti da li da zakazuje ili će odmah biti primljen.

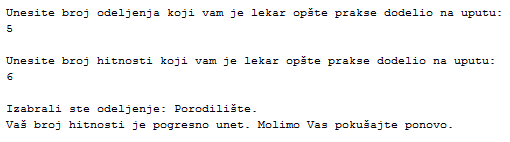
(Za demonstraciju uzećemo neke nasumične brojeve čisto radi prikaza funkcionisanja projekta):



(Slika 2.5. Prikaz unosa od strane korisnika, dobijanje informacije na osnovu unetih podataka)

Informacija koja se na kraju dobije je kratak opis šta je korisnik uneo i koji su mu sledeći koraci. Kao što se može videti, ovaj princip je vrlo lak i jednostavan za korišćenje.

Ako se u nekom slučaju desi da korisnik loše unese broj odeljenja ili broj hitnosti dobiće napisano gde je pogrešio i tu grešku može da ispravi odmah jer će se ponovo otvoriti početni tekst i mogućnost ponovnog unosa brojeva.

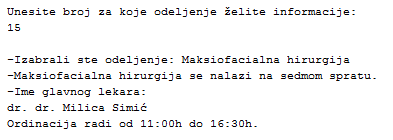


(Slika 2.6. Pogrešno unet jedan od brojeva, dobija se obaveštenje o pogrešnom unosu)

Opcija 2

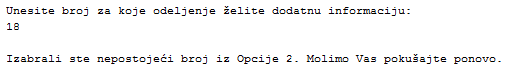
Opciju 2 možemo nazvati opciom za informisanje. Ova opcija poziva se unošenjem broja 2. Pokretanje opcije 2 povlači pitanje šta želimo da saznamo o izabranom odeljenju. Unosi se isti broj odeljenja jer je ovaj program napravljen tako da svako odeljenje ima svoj jedinstveni broj. Posle unosa broja odeljenja dobijaju se informacije koje obaveštavaju o odabranom odeljenju:

* Ime odeljenja je postavljeno na prvom mestu kako bi korisnik mogao da uporedi upisani broj sa izbačenim imenom odeljenja. Ukoliko je korisnik uneo broj koji nije namenjen za njegovo željeno odeljenje on se odmah vraća na početni tekst i tako može ponovo uneti broj i uporediti ga sa izbačenim odeljenjem.
* Sprat je druga stavka koju će ovaj program izbaciti. Tu se dobija kratak opis na kom spratu se nalazi uneto odeljenje kako bi korisnici mogli lakše da se orijentišu. To je jedna od bitnijih informacija jer pomaže korisnicima u snalaženju.
* Ime lekara je treća stavka. Tu korisnici mogu videti koji lekar će ih pregledati tj. koji lekar se nalazi na tom odeljnju.
* Radno vreme jer četvrta stavka koja je takođe od velike važnosti. Tu dobijamo kratak opis radnog vremena ordinacije. Na taj način korisnik može proveriti da li željena ordinacija radi u terminu u kom je želeo da ode i tako ne bi dolazilo do situacija nepotrebnog čekanja kada ordinacija ne radi.



(Slika 2.7. Unošenje broja odeljenja, povratne informacije o tom odeljenju)

Naravno i ovde se može desiti da korisni loše unese broj za koji nije predviđeno nijedno odeljenje. U tom slučaju korisnik opet dobija obaveštenje o pogrešnom unosu i mogućnost da ispravi grešku tako što će se ponovo otvoriti početni tekst.



(Slika 2.8. Unos pogrešnog broja, obaveštenje o lošem unosu)

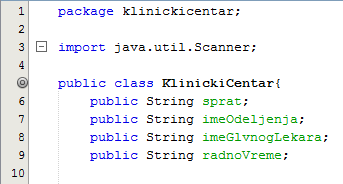
Projekat „InfoKC“ napravljen je tako da se početna strana uvek vraća kada korisnik završi sa unošenjem željenih podataka. Za ovakvu mogućnost ne postoji nikakvo dugme, već je projekat napravljen da se automatski vraća na početak. Korisnci neće morati da brinu da li je sve regularno ugašeno i da li će sledećeg korisnika opet dočekati početna strana. To se takođe može uračunati u konstataciju da je „InfoKC“ brz i lak za korišćenje.

# 4. Implementacija koda

Projekat je napravljen u okruženju NetBeans IDE 8.2. NetBeans je integrisano razvojno okruženje namenjeno razvoju Java tehnologija. Ovo okruženje može raditi na različitim platformama i veoma je korisno sredstvo za učenje i ostalih programskih jezika.

Početak kreiranja ovog projekta zasniva se prvenstveno na pravljenju početne (glavne) klase. Njoj zadajemo početne promenljive i metode koje planiramo da u narednim koracima pozovemo i da ih izvršimo.

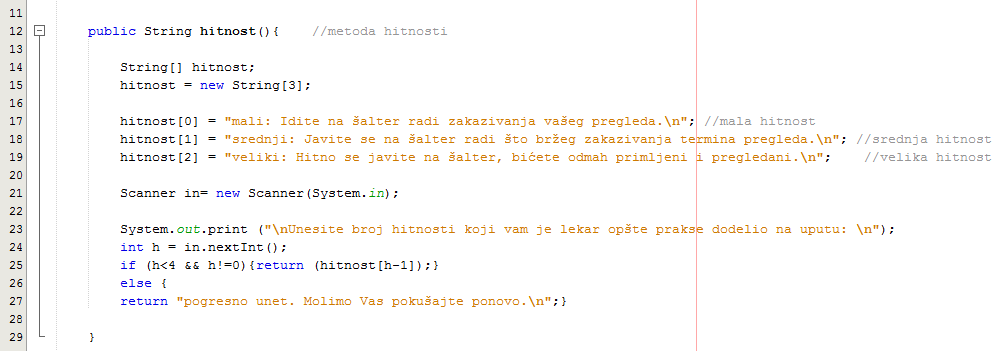
Ubacujemo java.util.Scanner jer će nam ovaj paket dosta pomoći prilikom izvršavanja programa. Java.util.Scanner paket je jednostavan tekst skener koji analizira primitivne tipove koristeći regularne izraze.



(Slika 3.1. Početak pravljenja glavne klase)

Kao što je već rečeno, napravljena je glavna klasa koja nosi naziv „KlinickiCentar“ . Glavna klasa sadrži par promenljivih koje ćemo kasnije pozvati kada budemo gradili konstruktor ove glavne klase. Svaki od ovih parametara će nam vratiti neko tekstualno rešenje pa smo odmah postavili da su ove promenljive String tipa. Naravno, svaku promenljivu i svaku metodu pisaćemo unutar tela klase (telo klase obeležava je {} zagradama).

Public je ključna reč u Javi. Može se objasniti kao javni pristup nekog člana, što žnači da svaka promenljiva ili metoda koja je definisana sa public biće vidljiva u svim drugim klasama. Baš zato smo sve ove promenljive definisali sa public jer će nam to koristiti kako bismo pozivali ove promenljive u drugim klasama.

Posle definisanja promenljivih, krećemo u definisanje dve ključne metode koje se nalaze u glavnoj klasi. Prva metoda koju ćemo napisati i objasniti jeste metoda hitnosti:

(Slika 3.2. Pravljenje metode „hitnost“ unutar glavne klase, postavljanje uslova za tu metodu)

Metoda “hitnost” se nalazi u telu glavne klase. Postavljen joj je String tip jer će ova metoda ispisivati neki tekst koji će na kraju biti zadali. Planirano je da ova metoda služi za ispitivanje hitnosti koju će korisnik upisati kada ceo projekat bude spreman za korišćenje. Napravili smo niz koji sadrži 3 člana i taj niz smo nazvali takođe hitnost (kao što se i metoda zove). Taj niz će brojati od nultog člana do drugog. Svaki clan definisan je određenim tekstom koji ispisuje sledeće podatke:

1. Nulti clan niza- Kada je hitnost mala, pacijent treba da ide i da zakaže termin pregleda jer se ne nalazi u zdravstvenoj opasnosti, pacijent je male primarnosti .
2. Prvi clan niza- Kada je hitnost srednja, pacijent ima malo veću urgentnost od prethodne (male) hitnosti ali još uvek se ne svrstava u najprimarnije pacijente. Poželjno je da zakaže svoj termin pregleda i njegov pregled će biti pre pregleda pacijenata sa malom hitnošću.
3. Drugi clan niza- Kada je hitnost velika, pacijent ima najveću urgentnost, takoreći svrstava se u najprimarnije pacijente. Poželjno je da se pacijent odmah uputi ka lekaru, njegov pregled će se odmah izvršiti.

Scanner in = new Scanner(System.in)- koristimo za pozivanje klase Scanner koju smo u prethodnom obajsnili.

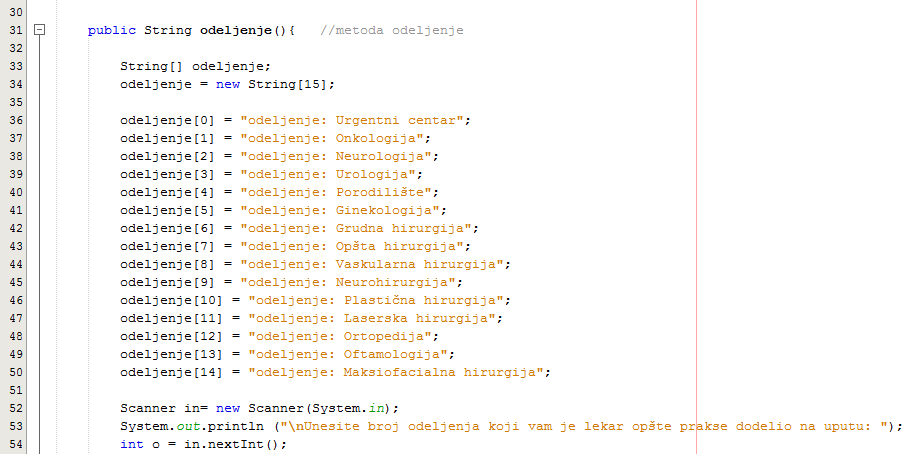
System.out.print(“ ”) –koristimo za unos nekog teksta, u ovom slučaju želimo da pacijenta obavesimo koji su mu dalji postupci u korišćenju “InfoKC”-a.U ovoj metodi nas interesuje broj hitnosti korisnika, zato postavljamo tekst koji korisniku traži unos broja hitnosti koji mu je lekar opšte prakse dodelio na uputu.

Int h = in.nextInt(); -je sledeća linija koja nam treba za čitanje broj koji je korisnik uneo. Taj broj je upisuje kao promenljiva h (nova promenljiva koja predstavlja hitnost) i ona je Int tipa.

Pošto smo dobili novu promenljivu h koja ima neku vrednost, u interesu nam je da postavimo neke određene uslove kako program ne bi izbacivao greške. Dakle, ova metoda će raditi samo za vrednosti

h =1, h =2 i h =3. U svim ostalim slučajevima unos će se računati kao loš unos jer su upisani brojevi koji nisu definisani.

Ovaj problem vrlo lako možemo rešiti unošenjem jedne If petlje. Prvi uslov ove if petlje je da promenljiva h bude manja od broja 4 i različita od broja 0. U tom slučaju metoda vraća članove niza hitnosti, tj. tekstualni opis svakog člana. Ako taj uslov nije zadovoljen, odnosto promenljiva h je jednaka 0 ili veća i jednaka od 4 program će ispisati da je broj hitnosti loše unet i da se može ponovo uneti.

Metoda “hitnost” je jedna od metoda glavne klase. Sledeća metoda koja je veoma bitna jedte metoda odeljenja:

(Slika 3.3. Pravljenje metode “odeljenje” unutar glavne klase)

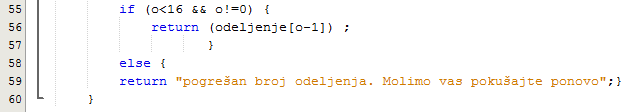
Metoda “odeljenje” nalazi se u telu glavne klase. Takođe je obležena sa String tipom jer će se njene informacije nalaziti u tekstualnom obliku i obeležena je sa public jer ćemo kasnije i ovu metodu pozivati iz različitih klasa. Korišćen je skoro isti princip kao i kod prethodne metode.

Metoda “odeljenje” sadrži niz odeljenje (isti naziv kao i metoda). Ovaj niz ima 15 članova i svaki clan tog niza ispisuje koje odeljenje je izabrano unešenjem nekog određenog broja. Naš klinički centar sadrži 15 odeljenja tako da smo svih tih 15 odeljenja povezali sa određenim članom niza.

Poslednje tri linije su iste kao i u prethodnoj metodi, pozvali smo klasu Scanner, ispisali tekst koji upućuje korisnika šta je sledeće šta treba da unese kako bi mu program izbacio željeni rezultat.

Taj broj koji će korisnik upisati u našem kodu upisuje se u promenljivu o (nova promenljiva koje označava broj odeljenja).

Sledeće bitno u ovom kodu jeste uslov koji moramo da postavimo kako program ne bi prikazao grešku u slučaju unosa parametara koji nisu definisani :

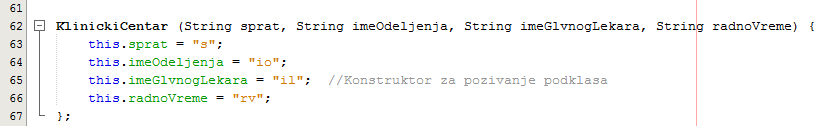


(Slika 3.4. Postavljanje uslova za metodu “odeljenje”)

Ponovo uvodimo if petlju koja u odnosu na uslov vrši određeni ispis.

Parametar o ima onu vrednost koju je korisnik upisao. Prvi uslov if petlje jeste da ta vrednost mora da bude manja od 16 i različita od 0. U tom slučaju program ispisuje tačno ono odeljenje koje je predviđeno unetim brojem korisnika. Ako prvi uslov nije zadovoljen, tj upisana vrednost je jednaka 0 ili je veća ili jednaka od 16, program će obavestiti korisnika da je lose uneo broj odeljenja i da to može ponovo upisati.

Naša glavna klasa” KlinickiCentar” ima 15 pod klasa. U tih 15 podkalsa ubrajamo odeljenja naše klinike. Kako bi te podklase nasledile neke parameter glavne klase napravili smo konstruktor sa tim parametrima. Taj konstruktor ćemo kasnije pozivati u tim podklasama.



(Slika 3.5. Pravljenje nepraznog konstruktora za građenje podklasa)

Konstruktor uvek nosi naziv klase od koje je napravljen. Tako ovaj konstruktor ima nazina naše glavne klase (KlinickiCentar).U prvoj zagradi () napisano je koje parametre ovaj konstruktor sadrži sa deklarisanim tipom parametara.

This je takođe jedna od ključnih reči koje se mogu videti u Java programiranju. Koristi se unutar neke metode ili, kao u ovom slučaju, konstruktoru klase. Funkcioniše kao referenca na tekući objekat koji se poziva u metodi ili konstruktoru.Može se korititi da označi bilo koji član tekućeg objekta.

Uz pomoć ključne reči this, pozivamo parametre koje konstruktor sadrži i dajemo ima određene vrednosti. Sva ta pozivanja vršimo u telu konstruktora.

Ovaj konstruktor sadrži 4 promenljive:

1. Sprat- promenljiva koja treba da sadrži napisano na kom je spratu željeno odeljenje. Pošto očekujemo tekstualni odgovor ovaj parametar postavljamo kao String, odnosno odvajamo navodnicima (“ ”). Pod tim navodnicima upisujemo bilo koji karakter da bi posle mogli da izvšimo upisivanje kada budemo pravili podkase.
2. Ime odeljenja- promenljiva koja opisuje odeljenje koje je korisnik odabrao (odnosno koji broj odeljenja je korisnik uneo). Takođe postavljamo da bude String tipa jer očekujemo teksualni odgovor. Pod navodnike stavljamo bilo koji karakter kako bi smo kasnije u podklasama vršili upis.
3. Ime glavnog lekara- promenljiva koja ispisuje ime lekara za odabrano odeljenje.
4. Radno vreme- promenljiva koja obaveštava kada radi željeno odeljenje.

Ispisivanjem ovih parametara završili smo konstruktor namenjen za pozivanje u podklasama. Sad će nam trebati još jedan konstruktor koji će biti prazan a pozivaćeno ga u test klasi.

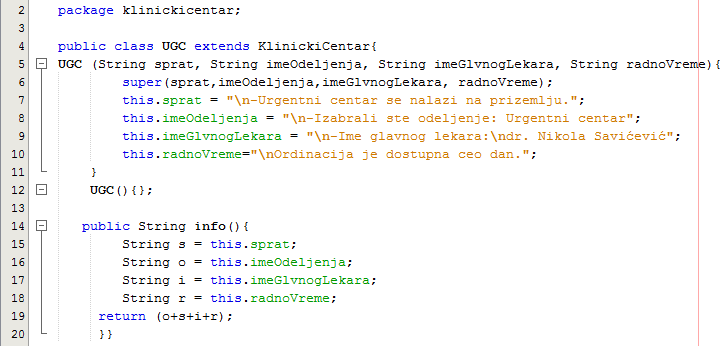


(Slika 3.6. Pravljenje praznog konstruktora)

Time završavamo celu glavnu klasu “KlinickiCentar” koja sadrži dovoljno podataka koji je opisuju.

Naša glavna klasa sadrži 15 podklasa koje predstavljaju odeljenja na klinici. U Javi, podklase su klase koje nasleđuju neku metodu ili metode iz superklase.

Sve naše podklase koje predstavljaju odeljenja, rađene su na isti način. Iskoristićemo jednu radi objašnjavanja koda:



(Slika 3.7.1. Rađeno za: Urgentni centar)

UGC je podklasa glavne klase “KlinickiCentar”. Ova klasa rađena je za odeljenje Urgenti centar.

Odmah na početku napravljen je konstruktor “UGC” koji sadrži parametre sa određenim tipom. Ti parametri su isti kao i kada smo pravili neprazan konstruktor “KlinickiCentar”. Sadrži parametre:

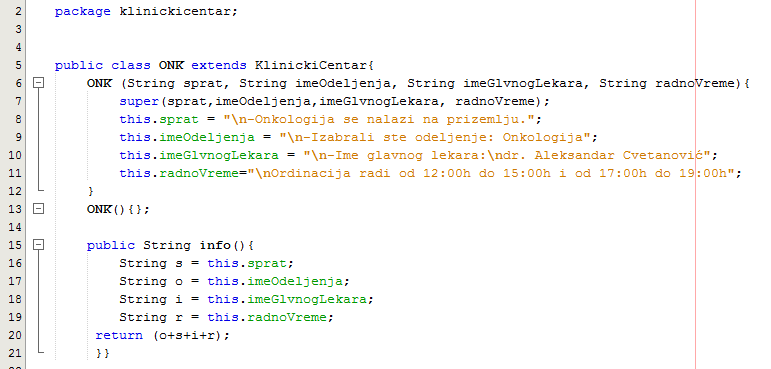
1. Sprat
2. Ime odeljenja
3. Ime glavnog lekara
4. Radno vreme

Iste te parametre nasledio je od gore pomenutog konstruktora. To se može izvršiti pomoću Java ključne reči super (). Super se koristi unutar podklase i poziva metod definisan u superklasi. U našem slučaju, iz napravljenog “UGC” konstruktora pozivamo “roditeljski” konstruktor (KlinickiCentar). U super upisujemo parametre koji se nasleđuju da bi mogli da ih definišemo.

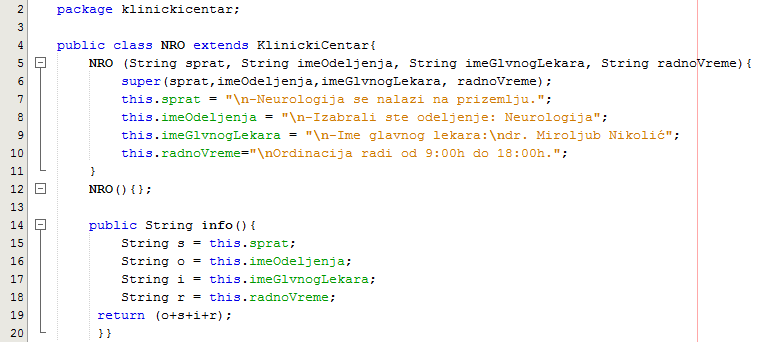
Sada nasleđene parametre ispisujemo kako bi oni opisali o kom odeljenju je reč. Dakle, za sprat smo odgovorili na pitanje gde se to odeljenje nalazi, za ime odeljenja odgovaramo na pitanje o kom odeljenju je reč, za ime glavnog lekara odgovaramo na pitanje koji lekar vrši preglede na unetom odeljenju i za radno vreme odgovaramo na pitanje od kad pa do kad uneto odeljenje radi.

Podklasa “UGC” ima jednu metodu info() koja će tačno po određenom redosledu rasporediti ove informacije . Na početak smo postavili ime odeljenja radi bolje preglednosti, odmah posle smo postavili sprat, zatim ime glavnog lekara i na kraju radon vreme odeljenja.

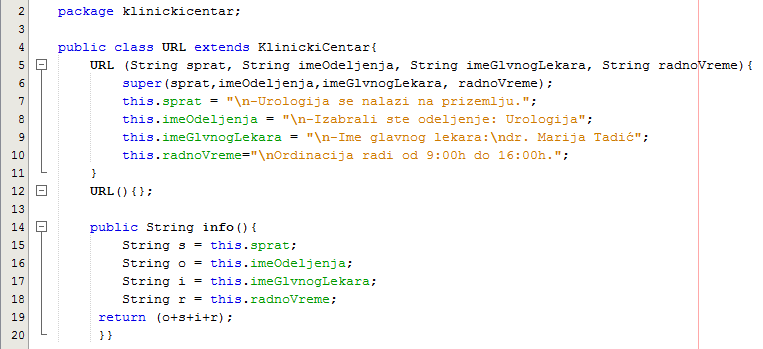
Kao što smo pomenuli, na isti način urađena je svaka sledeća podklasa koja definiše odeljenje tako da ćemo sada prikazati ostale podklase ali izostavićemo objašnjenja :



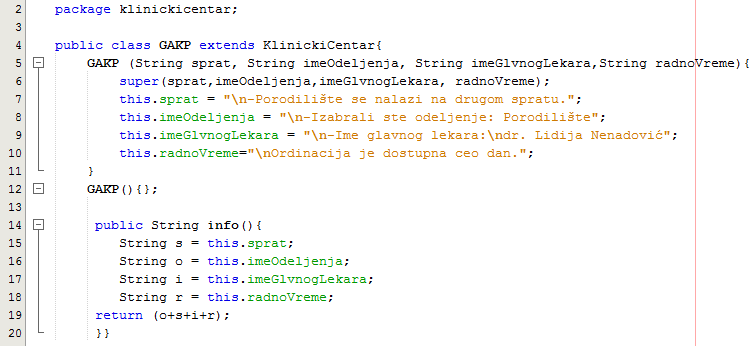
(Slika 3.7.2. Rađeno za odeljenje: Onkologija)



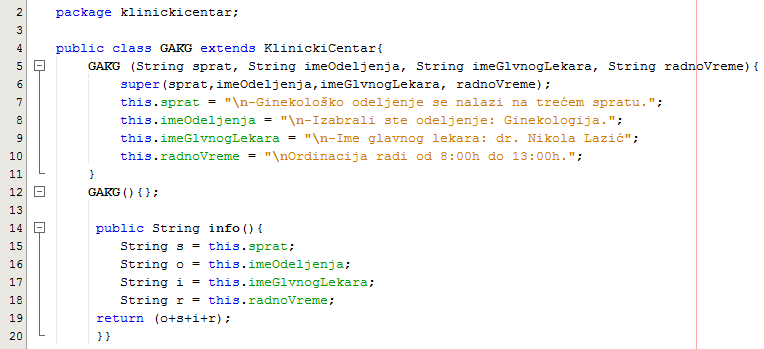
(Slika 3.7.3. Rađeno za odeljenje: Neurologija)



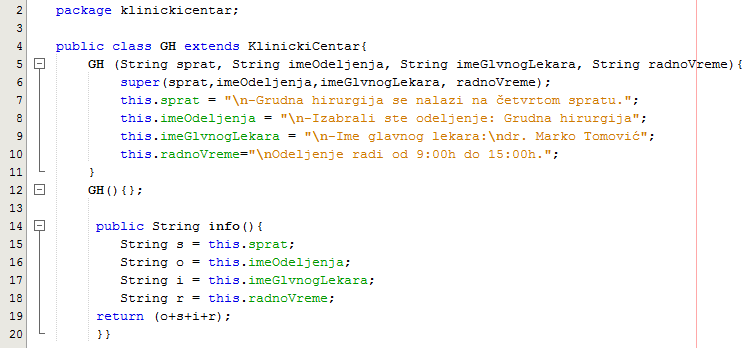
(Slika 3.7.4. Rađeno za odeljenje: Urologija)



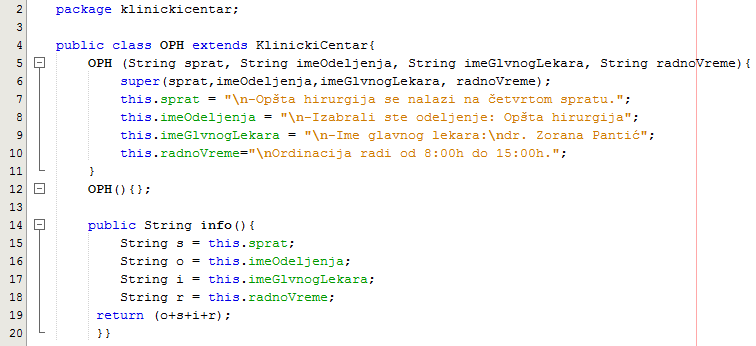
(Slika 3.7.5.Rađeno za odeljenje: Porodilište)



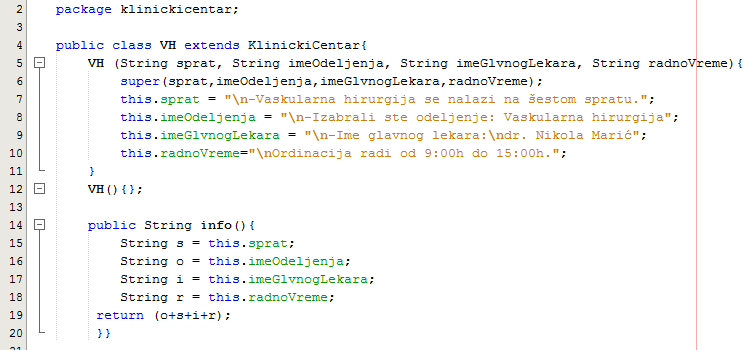
(Slika 3.7.6. Rađeno za odeljenje: Ginekologija)



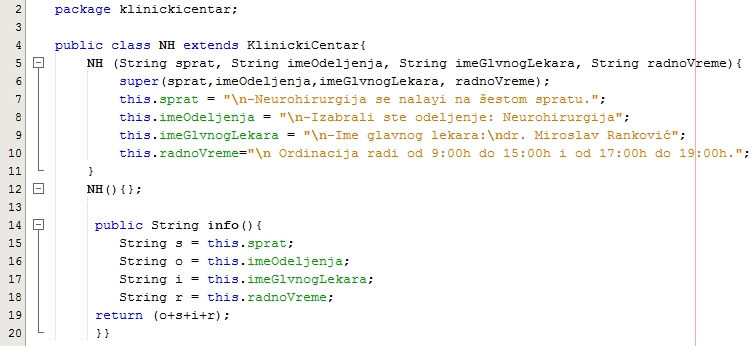
(Slika 3.7.7.Rađeno za odeljenje: Grudna hirurgija)



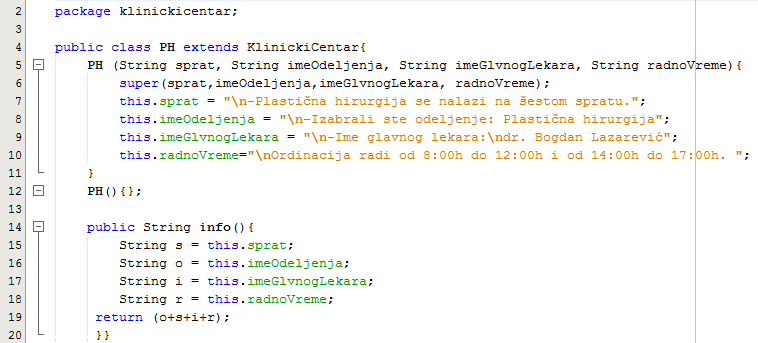
(Slika 3.7.8. Rađeno za odeljenje: Opšta hirurgija )



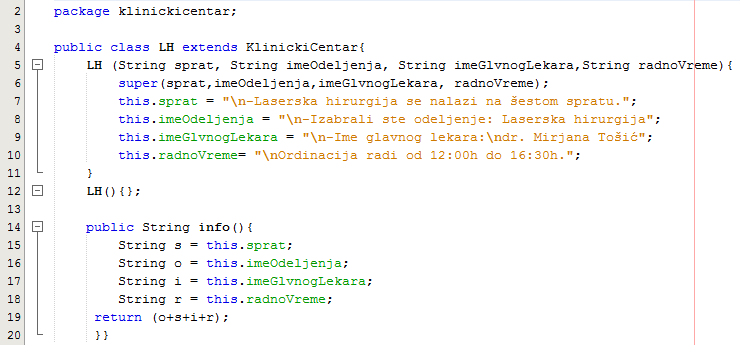
(Slika 3.7.9. Rađeno za odeljenje: Vaskularna hirurgija )



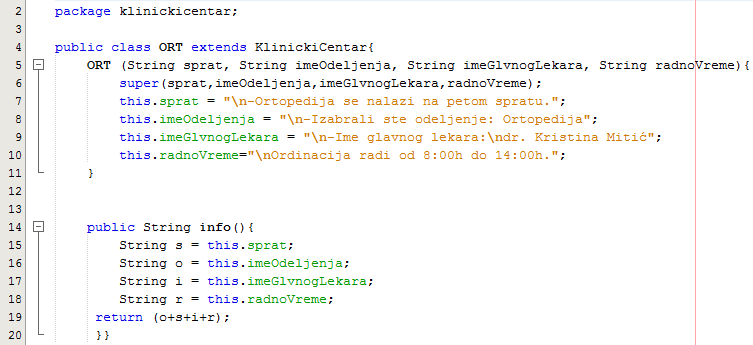
(Slika 3.7.10. Rađeno za odeljenje: Neurohirurgija )



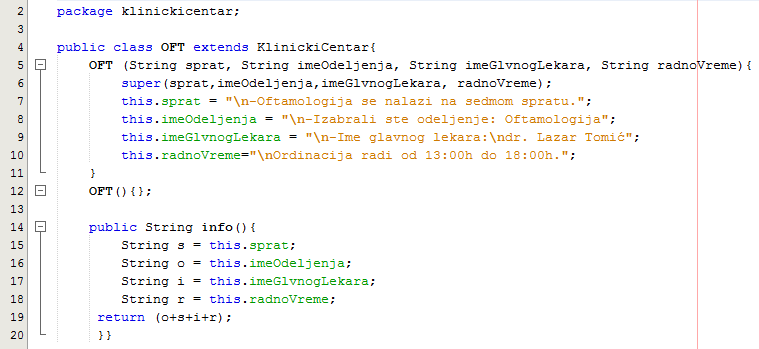
(Slika 3.7.11. Rađeno za odeljenje: Plastična hirurgija )



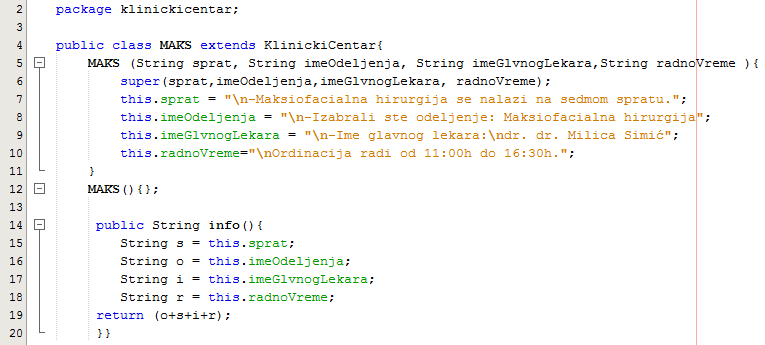
(Slika 3.7.12. Rađeno za odeljenje: Laserska hirurgija )



(Slika 3.7.13. Rađeno za odeljenje: Ortopedija )



(Slika 3.7.14. Rađeno za odeljenje: Oftamologija )

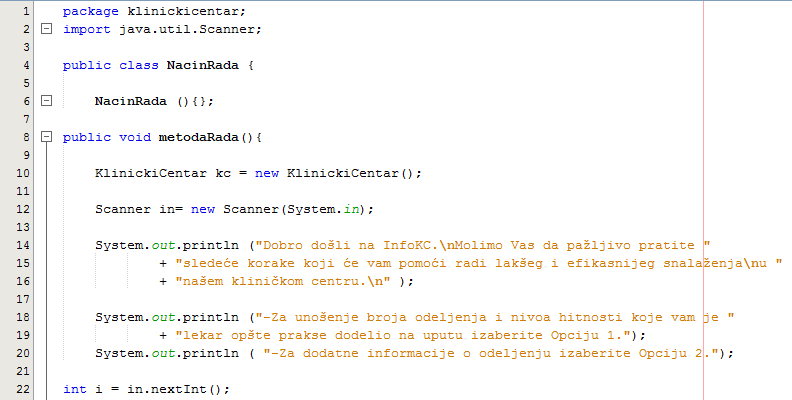


(Slika 3.7.15. Rađeno za odeljenje: Maksiofacialna hirurgija )

Na ovaj način završena je glavna klasa sa svojim podklasama.

Imali smo za ideju da napravimo jednu klasu koja će sadržati metodu koja obavlja većinu programa. Cilj nam je bio da ovu klasu posle pozivemo u tesk klasi i tako napravimo program koji će se iznova i iznova izvršavati, tj. kada god završimo unošenje i ne želimo više da koristimo program, on će se ponovo pokrenuti da bi dočekao narednog korisnika.

Napravljena je nova klasa kojoj smo dali naziv NačinRada, jer ona na neki način definiše kako će program funkcionisati.

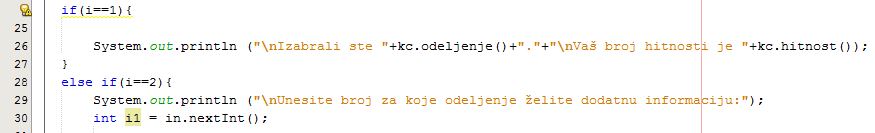


(Slika 3.8. Pravljenje nove klase “NacinRada” i metode “metodaRada”)

Napravljen je prazan konstruktor koji ćemo kasnije takođe pozivati u test klasi.

Ova klasa sadrži metodu koja je nazvana “metodaRada”.U toj metodi smo pozvali prazan konstruktor iz glavne klase “KlinickiCentar”. Sada krećemo u sređivanje našeg programa.

Zamišljeno je da početni ekran prikazuje korisniku uvodni tekst i obaveštava ga o mogućim opcijama koje on može da izabere.

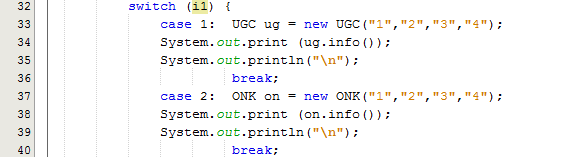
Za unošenje broja na osnovu koje će se odabrati Opcija 1 ili Opcija 2 uvodimo novi parameta i koji će biti Int tipa jer uzima vrednost broja. Kako bi taj parametar i odlučivao koja opcija je u pitanju postavljamo određeni uslov:

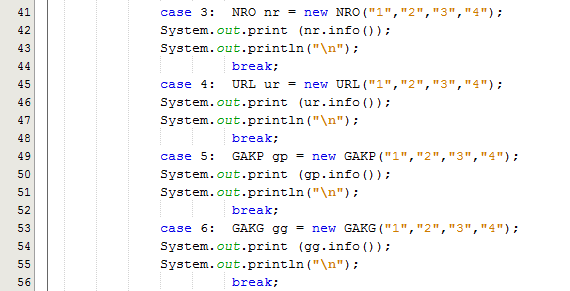
(Slika 3.9. Uslov za odabir navedenih opcija)

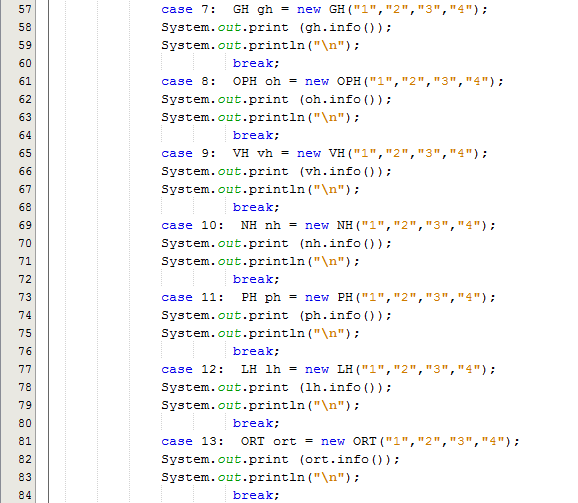
Ako parametar i dobije vrednost 1 tada će program krenuti da radi Opciju 1. Za Opciju 1 pozivamo metodu odeljenje iz klase KlinickiCentar i metodu hitnost isto iz klase KlinickiCentar. Sve ovo je složeno tako da se na ekranu (u zavisnosti od unosa) ispisuje puna rečenica koja obaveštava korisnika šta je uneo i da li zakazuje termin pregleda ili ne.

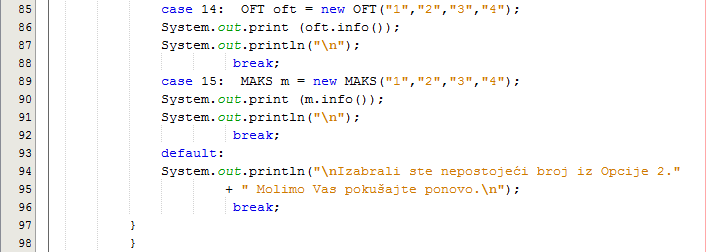
Ako parametar i dobije vrednost 2 tada će program izvršavati Opciju 2. Opcija 2 predstavlja informacionu opciju. Ovde je potrebno da korisnik unese broj željenog odeljenje kako bi dobio informacije o tom odeljenju. Pomoću System.out. println obavešavamo korisnika da nam je potreban broj odeljenja kako bi program izbacio potrebne informacije.

Broj koji korisnik unese dodeliće se promenljivoj i1. Ta promenljiva imaće onoliko vrednosti koliko imamo odeljenja na tom Kliničkom centru, dakle 15. Predstavljanje ovih odeljenja urađeno je case izjavom, tj. switch. Za takav izbor smo se opredelili najviše zbog preglednosti:









(Slika 3.10. Pravljenje 15 slučajeva za svako odeljenje iz Opcije 2 )

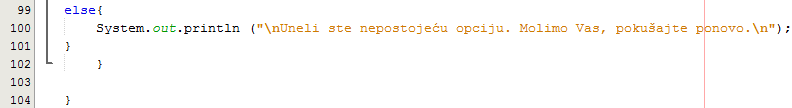
Napravljeno je 15 slučajeva. Svaki slučaj određuje odeljenje pod zadatim brojem.

Najpre smo morali da pozovemo konstruktor određenog odeljenja. Taj konstruktor je napravljen u podklasi glavne klase. Pošto ovde nismo zvali prazan konstruktor već konstruktor koji je sadržao parametara uneli smo bilo koja 4 karaktera String tipa kako bi naglasili da se radi o nepraznom konstruktoru.

Sledeća stavka je da se ispišu vrednosti koje se nalaze u metodi info(). Ova metoda je različita za svako odeljenje i pošto je String tipa ispisaće informacije za svako odeljenje.

U ovim slučajevima postavljen je i default koji, kada se uneti broj ne poklapa sa ovi slučajeva, ispisuje da je izabran nepostojeći broj odeljenja iz Opcije 2 I da korisnik može ponovo uneti broj.

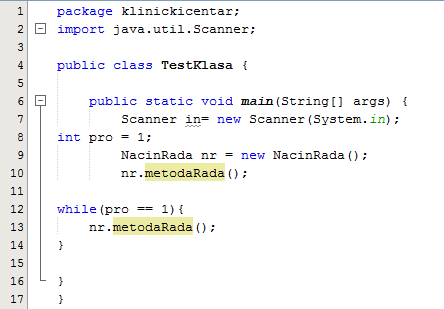
Sve ovo do sada je ispitivalo šta će se desiti kaka je početni parameta i jednak 1 ili 2, ali, kako ne bi došlo do pojavljivanja greške, uzeli smo u opticaj I slučaj da opcija nije dobro uneta, I to je onaj drugi deo if petlje:



(Slika 3.11. Ispisivanje kako je unet loš broj opcije)

Tada korisnik dobija obaveštenje da je uneo pogrešan broj opcije i da može uneti ponovo.

Time je završena klasa NacinRad. Jedino što nam je ostalo kako bi naš program dobro funkcionisao jesta da sada napravimo Test klasu u kojoj ćemo pozivati klasu NacinRada. Na taj način program će raditi iznova i iznova. Korisnici neće biti u obavezi da gase svoje podatke i otvaraju početni ekran za naredne korisnike. Ovaj program napravljen je da to resave sam:



(Slika 3.12. Pravljenje test klase sa glavnom metodom)

Naša test klasa sadrži glavnu metodu.

Da bi program počeo sa radom on mora da odnekud počne da se izvršava. To se radi u glavnoj (main) metodi. Nijedna aplikacija ne može raditi ako ne poseduje glavnu metodu.

Test klasa nam najviše znači kako bi napravili da se program vrti u krug i iznova izbacuje početni tekst.

Zato, ovde je postavljena nova promenljiva kojoj je odmah na početku dodeljen broj 1 (pro=1).

Pozvan je konstruktor klase NacinRada kako bi mogli da pozovemo metodu metodaRada koja ima ispisano izvršavanje celog programa. Ovde smo koristili while petlju i postavili joj uslov da ako je pro=1 metodaRada će se izvršavati. Pošto smo inicializovali promenljivu pro da je uvek jednaka 1 ova while petlja će se stalno izvršavati.

Ovde je kraj u ispisivanju kodova za ovaj projekat. Kod je ispisan veoma jednostavno i shvatljivo tako da se nadamo da razumevanje ovog koda nije bilo toliko komplikovano.