

Univerzitet „Džemal Bijedić“ u Mostaru
Fakultet informacijskih tehnologija
Predmet: Paralelni i distribuirani sistemi

Upravljanje radom sistema liftova
- Seminarski rad -

Studenti:

53

51

Mostar, novembar 2010.

Sadržaj

Sažetak.....	2
1. Uvod.....	3
2. Analiza scenarija rada sistema liftova	3
3. Dizajn sistema.....	4
3.1. Dijagram klasa.....	5
3.2. Logička arhitektura aplikacije.....	5
3.2.1. Simulator putnika.....	7
3.2.2. Simulator tipki	8
3.2.3. Simulator lampica	9
3.2.4. Simulator kretanja lifta	10
4. Implementacija sistema	12
5. Testiranje simulacijskog softvera.....	14
6. Korisničko uputstvo.....	15
7. Zaključak.....	Error! Bookmark not defined.
8. Literatura	Error! Bookmark not defined.
9. Prilog	Error! Bookmark not defined.
Slika 1 Dijagram „Načini korištenja“ sistemom lifta	4
Slika 2 Dijagram klasa - objekti sistema upravljanja liftom	5
Slika 3 Logička arhitektura aplikacije (prikaz paketa)	6
Slika 4 Dijagram aktivnosti za simulator putnika	8
Slika 5 Dijagram aktivnosti za simulator tipki	9
Slika 6 Dijagram aktivnosti za simulator lampica	10
Slika 7 Dijagram aktivnosti za simulator kretanja lifta	11
Slika 8 Dijagram stanja lifta.....	12
Slika 9 Prozor za odabir parametara simulacije.....	15
Slika 10 Glavni simulacijski prozor	16

Sažetak

U ovom radu je prezentiran model upravljanja sistemom liftova te je za te potrebe razvijena odgovarajuća softverska aplikacija za simulaciju prezentiranog modela. Upravljanje sistemom liftova predstavlja složen proces, prije sve što se rad jednog takvog sistema odvija u realnom vremenu. Dodatni faktor koji utiče na složenost sistema upravljanja liftova sastoji je taj što se jedan takav sistem sastoji od više podсистема gdje svaki ima svoja ograničenja i pravila, a na kraju u cjelokupnom procesu učestvuje i korisnik-čovjek koji sam po sebi unosi dio nedeterminizma u cijeli proces upravljanja. Softverski program za rješavanje jednog ovakvog kompleksnog problema real time upravljanja sistemom liftova je ostvaren uz korištenje programskog jezika Java sa uvođenjem mehanizma višenitnosti. Primjenom navedenog mehanizam višenitnosti se omogućuje izvršavanje po nekoliko zadataka istovremeno – paralelno te se te osobine pristupilo rješavanju navedenog problema.

Ključne riječi: Real time sistem, simulacija upravljanja, sistem liftova.

1. Uvod

Cilj rada je napraviti model upravljanja sistema liftova, implementirati ga i izvršiti njegovu simulaciju kroz odgovarajuću softversku aplikaciju.

Osnovi zadatak lifta jeste da dolazi na poziv putnika i prevozi ih na odabrane spratove. Složenost zadatka se povećava sa činjenicom da u sistemu upravljanja mogu da postoje n liftova, m spratova i k putnika. Na osnovu navedenog se može zaključiti da postoje slijedeći gradivi elementi modela:

- Korisnici-putnici;
- Spratovi;
- Liftovi;
- Tipke;
- Lampice;
- Upravljački sklopovi.

U modelu lifta, koji će se koristiti u radu, mnogi elementi su poprilično pojednostavljeni. Električne karakteristike raznih elektromotora, tipki i lampica se ne razmatraju. Tipke ili jesu ili nisu pritisnute; lampice svijetle ili ne. Nadalje, upravljanje elektromotorima se ne obavlja direktno već preko posebnih elektronskih naprava kojima se samo šalju naredbe ("otvori vrata", "kreni gore" i sl.).

Integracija prethodno navedenih elemenata u jednu cjelinu kao i poštivanje pojedinačnih ograničenja i specifičnosti koje imaju pojedini elementu u cjelokupnom procesu je složen proces, te je stoga nužno detaljnije razmotriti pojedine elemente.

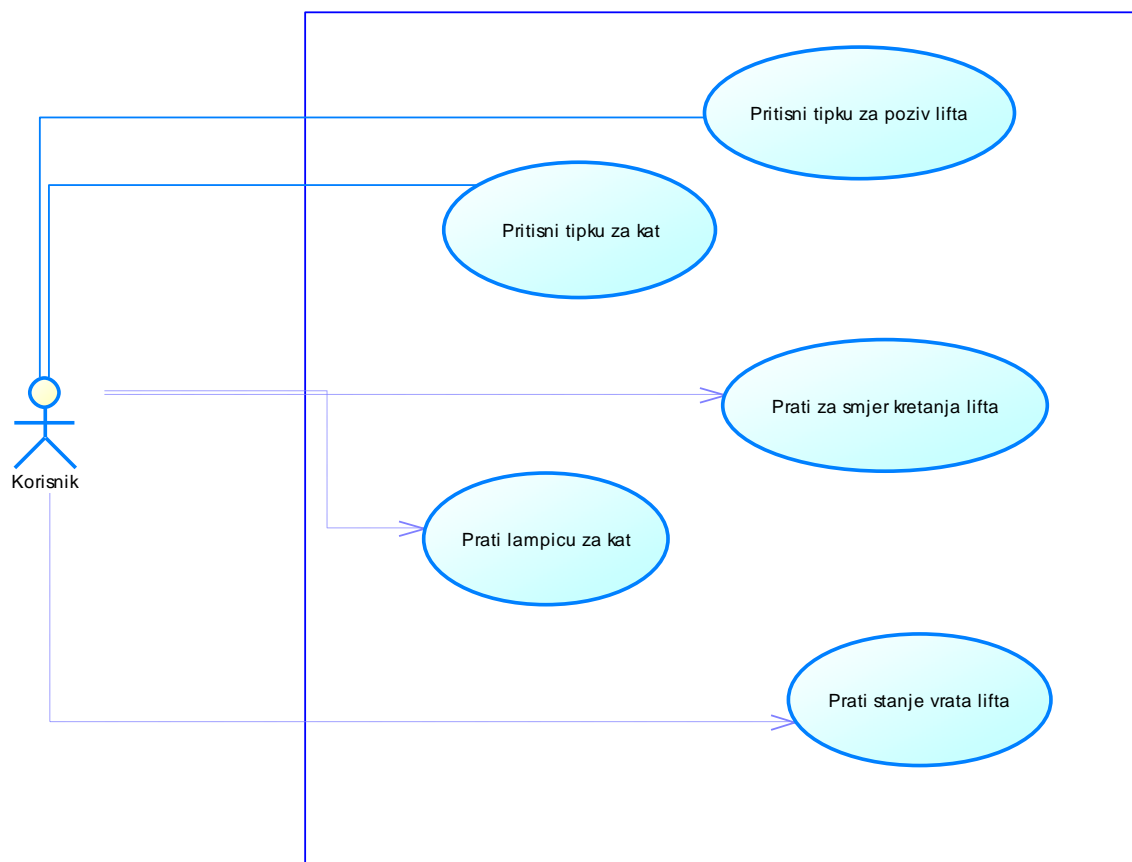
U narednom poglavlju će biti opisan scenarij rada sistema liftova.

2. Analiza scenarija rada sistema liftova

U ovom dijelu je opisan sistem rada liftova, te je na osnovu tih ograničenja i zahtjeva implementirana odgovarajuća softverska aplikacija za simulaciju upravljanja sistemom liftova. Kao što je već naglašeno u modelu lifta, koji će se koristiti u ovom primjeru, mnogi tehnički elementi su pojednostavljeni. Scenarij je baziran na osnovnim funkcijama koje jedan takav sistem ima i to, kretanje lifta prema gore i dole, otvaranje i zatvaranje vrata, i naravno, preuzimanje putnika. Sistem liftova bi se trebao koristiti u zgradi koja ima više spratova numeriranim sa brojevima od 0 do m spratova, dok svaki sprat može imati n liftova. Svaki sprat, osim prvog sprata i potkrovlja ima dvije tipke, jedna tipka služi za poziv lifta prema gornjem spratu a drugu za poziv prema donjem spratu. Ove tipke imaju i vizualnu signalizaciju te su osvijetljenje lampicama kada su pritisnute. Unutar liftova postoji set od $(m+1)$ tipki, po jedna za svaki sprat. Kada korisnik pritisne jednu od ovih tipki upali se odgovarajuća lampica i izdaje se komanda kako bi se posjetio odabrani spratu. Lampica se gasi kada lift stigne do destinacijskog sprata. Lift se kreće brzo između spratova, ali bi trebao biti u stanju usporiti dovoljno rano da se zaustavi na željenom spratu. Kad lift nema zahtjeva, on ostaje na trenutnom spratu otvorenih vrata.

Interakcija korisnika-putnika sa sistemom lifta je prikazana na slici 1. Korisnici imaju mogućnost da pokrenu nekoliko akcija u sistemu upravljanja liftom. Pune linije na slici 1. označavaju direktnu-fizičku interakciju korisnika dok isprekidane linije predstavljaju

kontrolne akcije gdje korisnici promatraju stanje i nakon toga donose odluke za daljnje djelovanje.



Slika 1 Dijagram „Načini korištenja“ sistemom lifta

Direktne akcije koje korisnik može pokrenuti pritiskom tipke se odnose na poziv lifta, gdje na upravljačkoj tabli postoje dvije mogućnosti odabira i to ukoliko se želi koristiti lift prema gornjem ili donjem spratu. Druga akcija koju korisnik pokreće putem tipke za sprat se odnosi na odabir određene destinacije sprata. Kontrolne akcije korisnika su bazirane na lampicama odnosno na vizualnom posmatranju, gdje korisnik može da prati smjer kretanja lifta prema dole ili gore, zatim lampicu za sprata koja mu daju informaciju o trenutnoj poziciji lifta kao i stanje vrata tj. da li su vrata otvorena ili zatvorena.

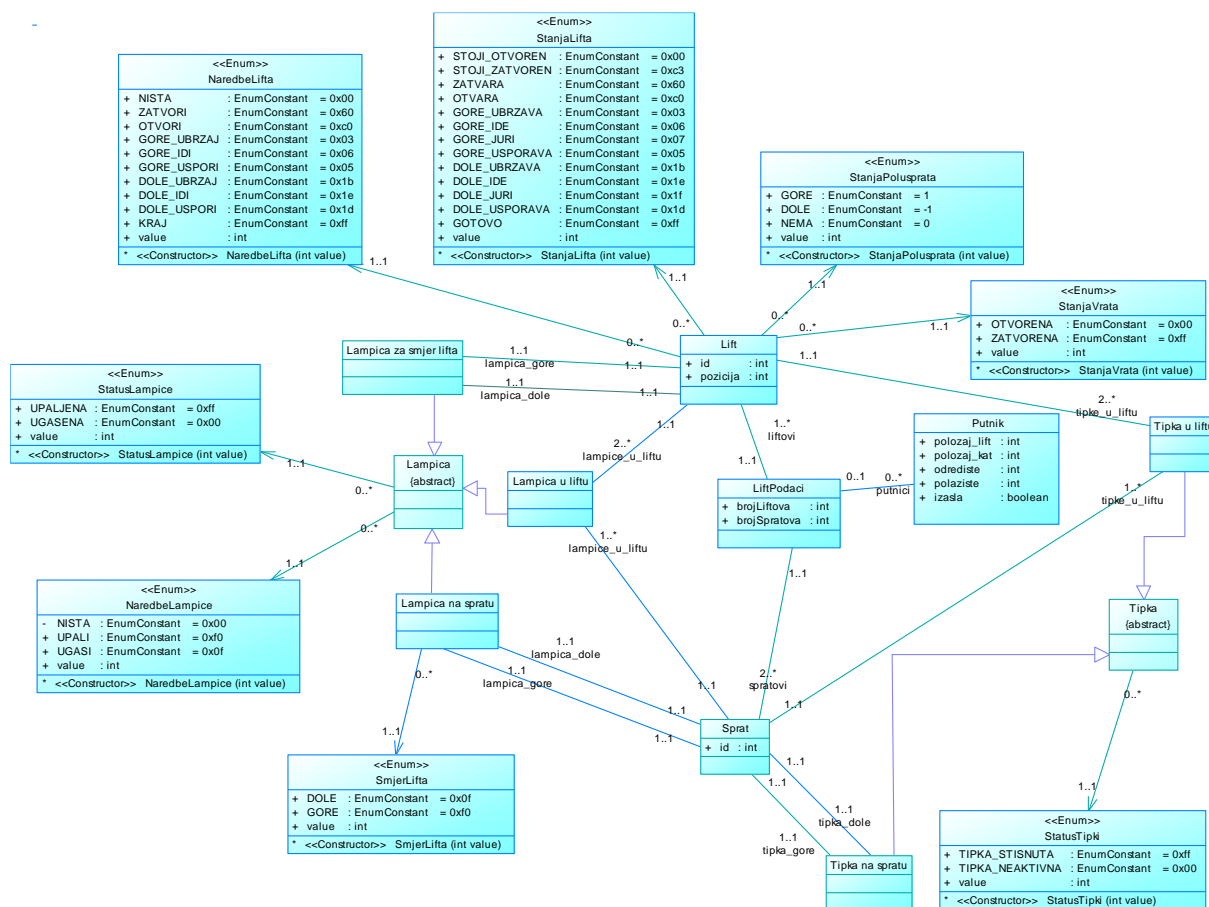
U narednom poglavlju je se predstavljen dizajn prethodno opisanog scenarija upravljanja sistemom liftova.

3. Dizajn sistema

Nakon što smo definisali elementi sistema upravljanje liftovima te opisan scenarij rada neophodno je da ti odgovarajuću dizajn sistema. Dizajn sistema će se predstaviti kroz nekoliko dijagrama, dijagram statičke strukture sistema tj. dijagram klase, dijagram arhitekture paketa aplikacije, dijagram stanja sistema i dijagram aktivnosti.

3.1. Dijagram klasa

Na slici 2 je data statička struktura sistema kroz prezentirani dijagram klasa. Ovaj dijagram sadrži klase koji su implementirani u simulaciji upravljanja sistemom liftova. Veze klasa jednih sa drugima predstavlja logičku povezanost između entiteta.



Slika 2 Dijagram klasa - objekti sistema upravljanja liftom

Dijagram klasa sadrži softverske klase kao što su *lift*, *sprat*, *tipka*, *lampica*, *putnik*, itd. Pored standardni klase postoje i enumeracijske klase kao što su *naredbe lifta*, *stanje lampica*, *status tipki* itd. Ove klase predstavljaju pojedina stanja čije vrijednosti mogu da poprime klase koje su u vezi sa njima. Tako npr. *klasa lampica* je u vezi enumeracijskom klasom *status lampice* može da poprime stanja kao što su *upaljena* ili *ugašena*. Još jedna specifičnost na dijagramu klasa su dvostruke veze između pojedinih klasa. Gdje između klase *sprat* i klase *tipka na spratu* postoji dvostruka veza *tipka_dole* i *tipka_gore*, postojanje ovakvih veza je neophodno za slučaj kada je potrebno pamtit (čuvati) informaciju za koji smjer je pritisnuta tipka. Ostali elementi dijagrama klasa su standardni te su razumljivi iz samog prikaza dijagrama.

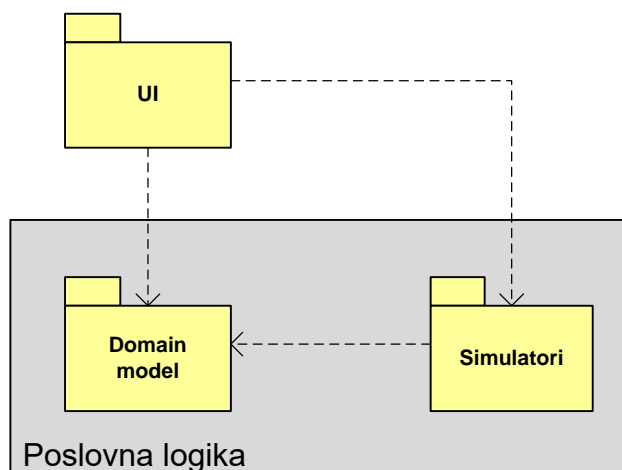
3.2. Logička arhitektura aplikacije

Na slici 3 je predstavljen dijagram logičke arhitekture aplikacije. Ovaj dijagram može biti prikazana UML dijagramima paketa, kao dio dizajn modela.

Logička arhitektura aplikacije predstavlja organizaciju softverskih klasa u pakete (ili imenske prostore), podsisteme i slojeve. Ta organizacija označava se kao logička arhitektura jer ne donosi odluke o tome kako će pojedini elementi biti raspoređeni po procesima operativnog sistema, niti po fizičkim računarima u mreži.

Sloj je grupa klasa, paketa ili podsistema koje imaju logički povezana zaduženja za jedan veći aspekt sistema. Često se koristi termin "kohezivna" zaduženja da bi se naglasila jaka međusobna povezanost tih zaduženja. Slojevi se organizuju tako da viši slojevi (npr. UI sloj) koriste usluge nižih slojeva.

U našem sistemu logičke arhitekture postoje tri paketa (slojeve) Simulator, UI i Domain model. Isprekidane strelice označavaju vezu ovisnosti (dependency) između pojedinačnih paketa. Klase iz UI paketa ovise o klasama (koriste ih) iz Domain modela i Simulator paketa. To znači da ako nešto izmijenimo u Domain model paketu ili Simulator paketu, to može uticati na klase iz UI paketa. Obrnuto ne važi, jer klase iz Domain modela ili Simulator sloja nemaju informaciju o postojanju UI sloja. Tako, ako izmijenimo UI sloj, klase iz Domain model ili Simulator sloja se ne moraju mijenjati.



Slika 3 Logička arhitektura aplikacije (prikaz paketa)

Sloj poslovne logike se sastoji od Domain logike i aplikacijske logike [1].

- Paket Domain modela implementira Domain logiku koja manipuliše Domain objektima i relacijama između njih.
- Paket Simulator sadrži aplikacijsku logiku koja je definisana na osnovu funkcionalnih pravila sistema upravljanja liftovima. Funkcionalna pravila se izvode iz predhodno prezentiranih dijagram *Načina korištenja*.

Sadržaj klasa Domain modela je prikazan u tabeli 1.

Tabela 1 Sadržaj paketa domain model

Klase Domain modela	
<ul style="list-style-type: none"> • Sprat.java • LampicaNaSpratu.java • LampicaULiftu.java • LampicaZaSmjerLifta.java • Lift.java • Lampica.java • LiftPodaci.java • Putnik.java 	Enumeracijske klase (stanja): <ul style="list-style-type: none"> • NaredbeLampice.java • NaredbeLifta.java • SmjerLifta.java • StanjaLifta.java • StanjaPolusprata.java • StanjaVrata.java • StatusLampice.java

<ul style="list-style-type: none"> • Tipka.java • TipkaNaSpratu.java • TipkaULiftu.java 	<ul style="list-style-type: none"> • StatusTipki.java
--	--

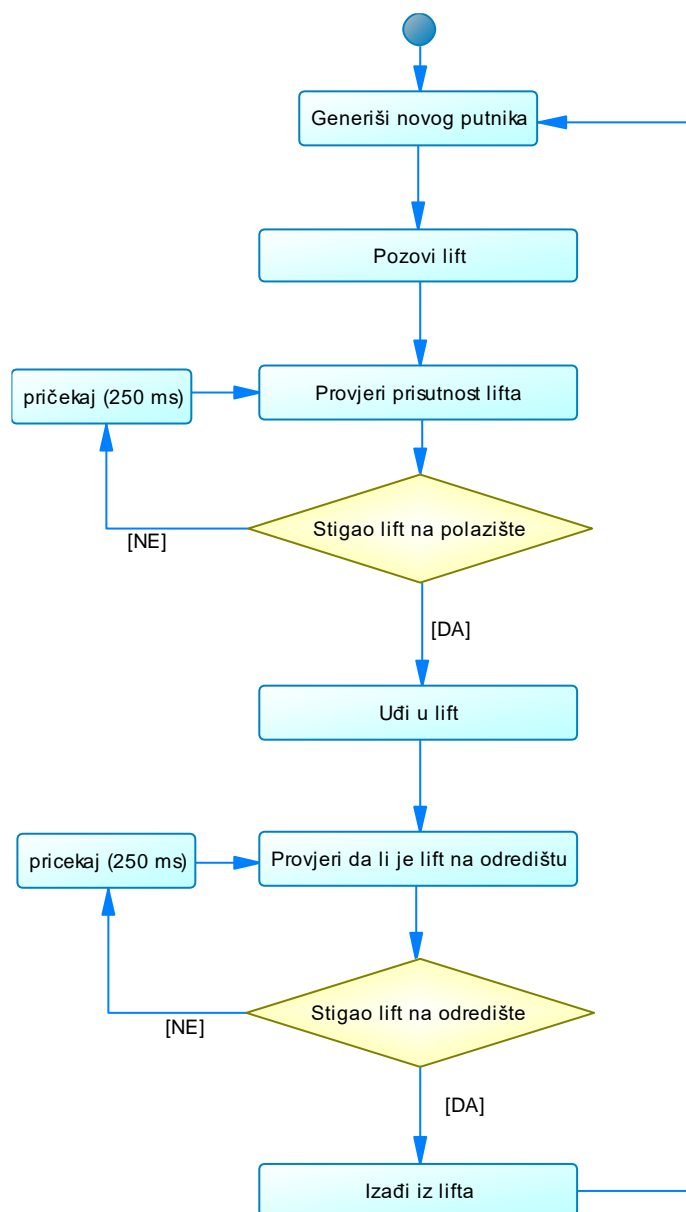
Klase aplikacijske logike definisane iz dijagrama „Načini korištenja“ su grupisane u paketu Simulator. U ovom paketu se nalaze slijedeće klase:

- SimulatorPutnika.java
- SimulatorTipki.java
- SimulatorLampica.java
- SimulatorKretanjeLifta.java

Ove klase predstavljaju odvojene module koji se izvršavaju u zasebnim nitima. U nastavku će biti prezentirani dijagramima stanja za svaku od klasa Simulator paketa.

3.2.1. Simulator putnika

Program simulator putnika je najjednostavnija modul. Zadužen je za kreiranje novog putnika koji se pojavljuju na različitim spratovima s različitim odredišnim ciljevima. Program je realiziran korištenjem klase putnik čiji svaki objekt može predstavljati jednog putnika. Osnovni program, nakon inicijalizacije podatkovne strukture, stvara novi zadatak koji periodički prolazi podatkovnu strukturu i mijenja status putnika, ako se on mijenja (npr. ulazi u lift). Početni zadatak periodički stvara novog putnika na slučajno odabranom spratu i umeće ga u podatkovnu strukturu. Simulirano ponašanje putnika prikazano je na slici 4 koja predstavlja dijagram aktivnosti za simulator putnika.



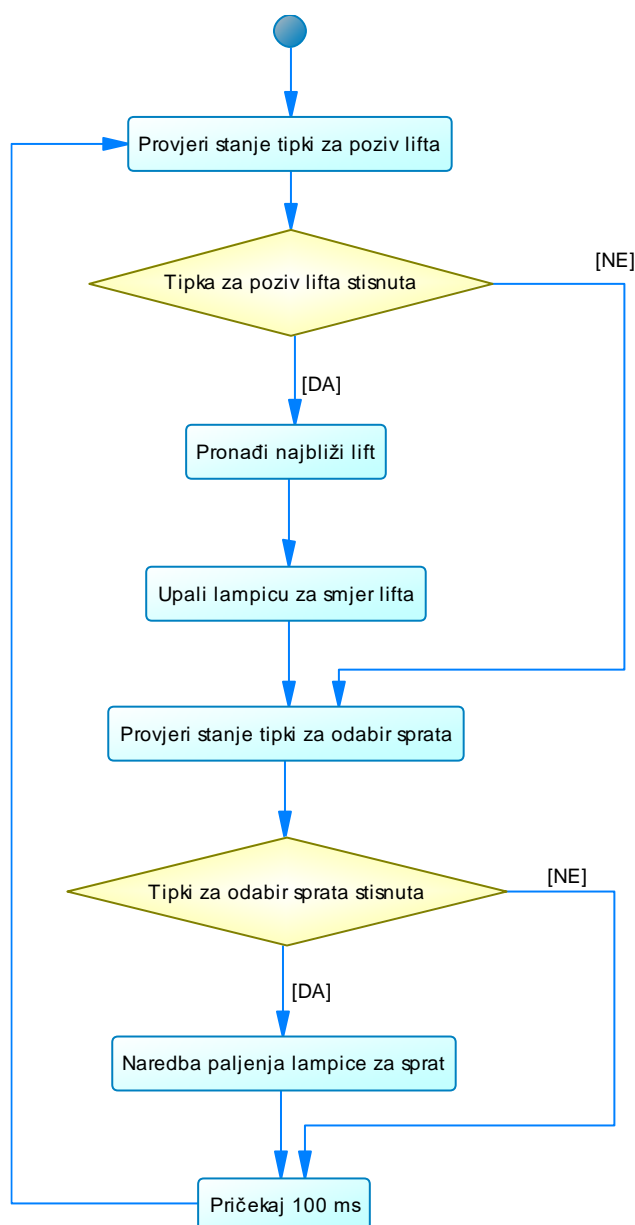
Slika 4 Dijagram aktivnosti za simulator putnika

U akciji *Pozovi lift* putnik će pritisnuti tipku za poziv, odnosno, program će u odgovarajući registar tipaka upisati oznaku pritisnute tipke. U akciji *Uđi u lift* putnik će također pritisnuti tipku odgovarajućeg sprata na kojem želi izaći. Nakon izlaska iz lifta putnik nestaje iz sistema. Da li je lift stigao i otvorio vrata i ide li lift u traženom smjeru, putnik saznaje jer vidi otvoreni lift i oznaku smjera kretanja lifta (u programu to očitava iz statusa svih liftova). Kada treba izaći iz lifta putnik također doznaje vizualno - vidi da su vrata otvorena i da je na zaslonu lifta prikazan odredišni sprat (opet se u programu to ostvaruje pregledom statusa aktuelnog lifta).

3.2.2. Simulator tipki

Simulator tipki je zadužen za provjeru aktivnosti tipki tj. provjerava se da li je tipka pritisnuta ili ne. Nakon interpretacije stanja tipki simulator daje naredbu upravljačkom mehanizmu za pokretanje odgovarajuće akcije shodno aktiviranoj tipki. Simulator tipki je zadužen za sve vrste akcija koje se mogu pokrenuti putem pritiska tipki u sistemu upravljanja liftovima. To podrazumijeva akcije pritiska tipke za poziva lifta kao i akcije za odabir sprata.

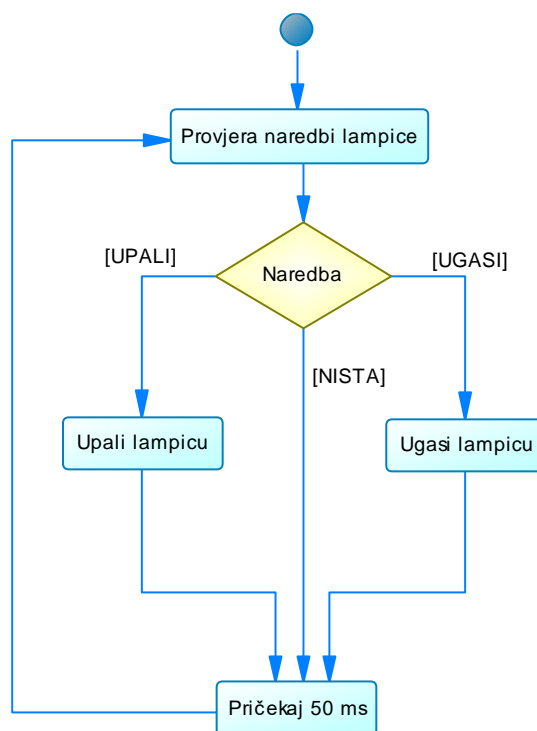
Kada je aktivirana tipka za poziv lifta simulator pronalazi najbliži lift u odnosu na trenutnu lokaciju sprata sa kojeg je pokrenuta naredba te izdaje upravljačkom mehanizmu naredbu za paljenje lampice za smjer kretanja lifta i upućuje lift na traženu lokaciju sprata. Akcija za odabir sprata se pokreće kada je aktivna tipka za odabir sprata te se ujedno i pali lampica za odabrani sprat. Prethodno navedena dva postupka provjere poziva lifta i odabira sprata se obnavlja svakih 100 ms. Na slici 5 je prikazan dijagram aktivnosti simulatora tipki.



Slika 5 Dijagram aktivnosti za simulator tipki

3.2.3. Simulator lampica

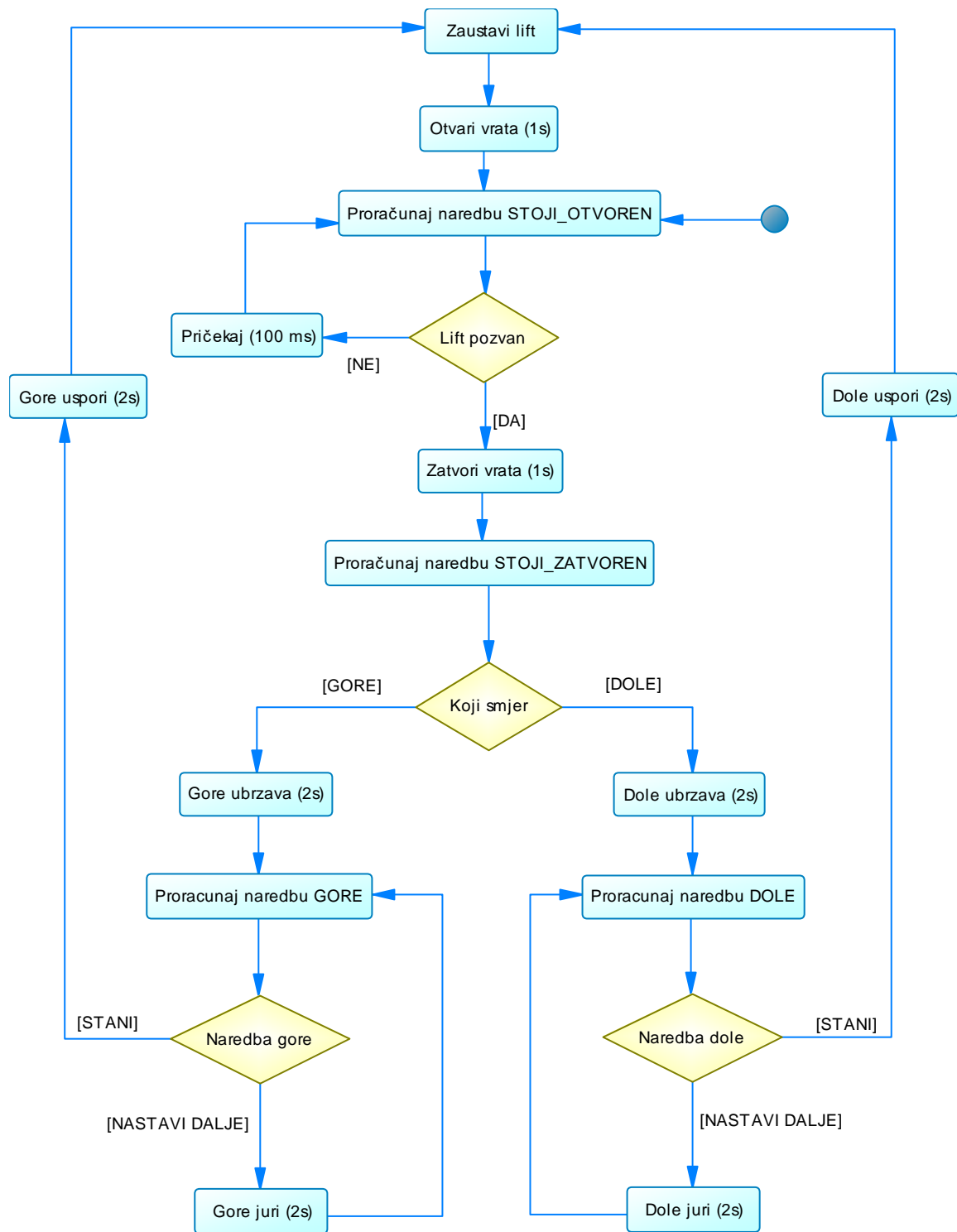
Simulator lampica predstavlja jednostavan mehanizam za provjeru naredbi lampice i izvršenju akcija nad lampicom. Dva su stanja koje lampica može poprimiti, da je upaljena ili ugašena. Simulator lampica je zadužen za sve lampice u sistemu. Provjerava se naredna lampice i ukoliko je aktivna simulator daje signal upravljačkom mehanizmu za promjenu stanja lampice ukoliko je bila upaljena ugasiti je ili obrnuto. Postupak provjere se odvija svako 50 ms. Na slici 6 je prikazan dijagram aktivnosti za simulator lampica.



Slika 6 Dijagram aktivnosti za simulator lampica

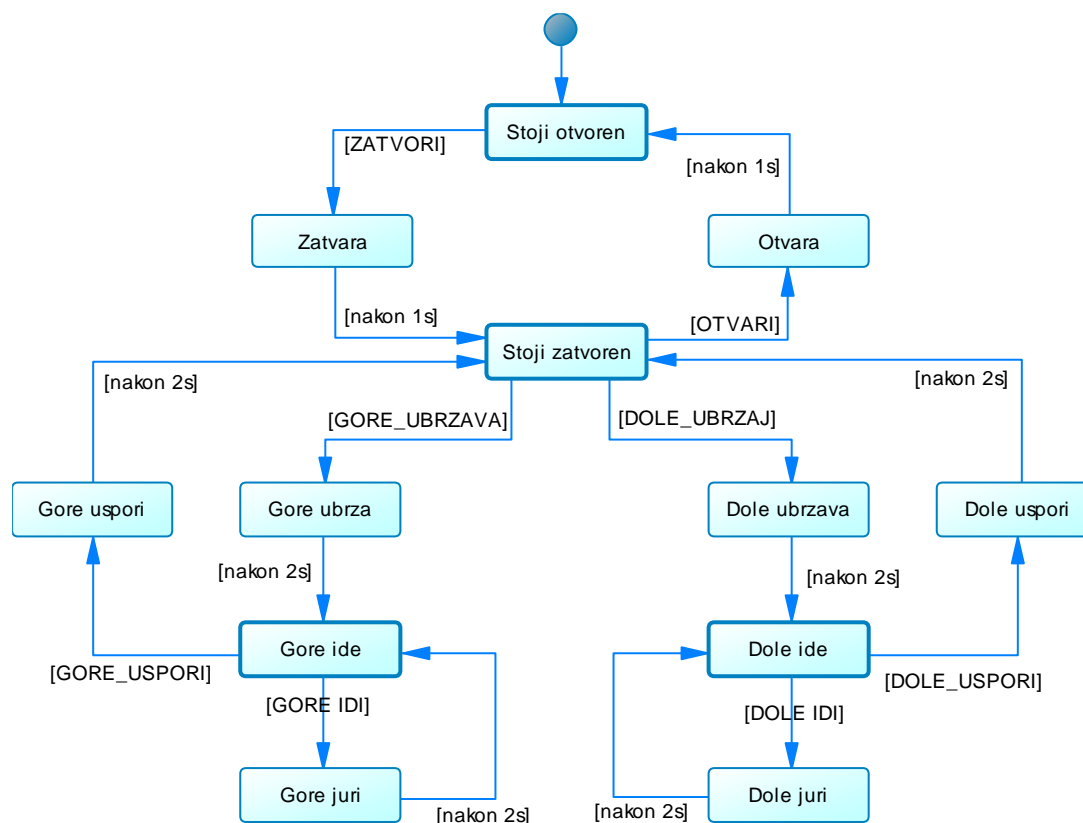
3.2.4. Simulator kretanja lifta

Simulator rada lifta interpretira naredbe upravljačkog programa koje se nalaze u zajedničkom spremniku te upravlja elektromotorima, pomičući lift gore ili dole, ili otvara ili zatvara vrata. Drugim riječima, simulator je "zamjena" za elektroniku koja bi se trebala naći uz svaki lift. Svaki lift simuliran je zasebnim zadatkom, ali je kod koji obavljaju svi ti zadaci isti. Obzirom da se u liftu (iako simuliranom) prevoze ljudi (simulirani) u sam simulator lifta ugrađene su neke sigurnosne provjere. Naprimjer, iako je lift možda dobio naredbu da otvori vrata, on to neće napraviti ako je u kretanju; ako se kreće prema gore ignorirati će naredbu kretanja prema dolje - najprije mora dobiti naredbu za zaustavljanje i slično. Ponašanje lifta koji je zadan uz problem može se prikazati dijagramom aktivnosti prikazanog na slici 7.



Slika 7 Dijagram aktivnosti za simulator kretanja lifta

Na osnovu dijagram aktivnosti za simulator kretanja lifta definiran je dijagram stanja lifta. Na slici 8 je prikazan dijagram stanja za simulator kretanja lifta.



Slika 8 Dijagram stanja lifta

Stanja u kojima lift prihvaća naredbe su: "Stoji otvoren", "Stoji zatvoren", "Gore ide" i "Dole ide" (označena su debljim okvirom). Ostala stanja prikazuju lift u obavljanju neke akcije. Tek kad akcija završi, može početi iduća. Npr. ako lift stoji, prvo mu treba zadati naredbu za ubrzanje, a tek potom za kretanje konstantnom brzinom. Ipak, s upravljačkog gledišta sva su nam stanja bitna jer odražavaju sve akcije koje lift radi te se na osnovu toga može zadati iduća naredba. Obzirom da je opis stanja prikazan dijagramom stanja, izvorni kod simulatora lifta je izgrađen oko tih stanja, tj. provjera što će lift učiniti obavlja se u odnosu na stanje u kojem se lift nalazi.

4. Implementacija sistema

Prethodno opisani model upravljanja sistemom liftova je implementiran kroz odgovarajući simulacijski softver. Za razvoj simulacijskog softvera je korišten programski jezik Java.

Za razvoj simulacijskog softvera je korišteno *Eclipse* razvojno okruženje, dok je za gradnju SWT korisničkog sučelja korišten softverski dodatak *Google Window Builder Pro*.

Eclipse predstavlja višezjezično softversko razvojno okruženje koje se sastoji od integrisanog razvojnog okruženja i proširivih plug-in dodataka. *Eclipse* je razvijan kao open source projekt te postoji mogućnost prilagodbe okruženja. Najčešće se koristi za razvoj aplikacija u Java programskom jeziku ali uz pomoću raznih *plug-in* moguća je nadogradnja za i druge programske jezike, kao što je: Ada, C, C++, COBOL, Perl, PHP, Python, Ruby. Uz nadogradnju sa odgovarajućim dodacima moguće je ovo okruženje koristiti kao alat za testiranje softvera, alata za performanse, web-alata, alat za poslovnu inteligenciju i izvješćavanja, alat za modeliranje, te za gradnju korisničkog sučelja.

Google Window Builder Pro je softverski dodatak za Eclipse razvojno okruženje koje se koristi za izradu korisničkih sučelja baziranih na Java tehnologijama. Ovaj dodatak podržava vizualni dizajn sučelja koristeći SWT i Swing biblioteke. Ovaj paket je izvorno razvijen od strane firme Instantiation. Nedavno ga je otkupila firma Google i time je paket *Window Builder Pro* postao besplatan kao svi ostali Google softverski proizvodi.

Kako se radi o upravljanju sistemima u realnom vremenu korištenje Java programskog jezika je bilo pogodno. Sistemi u realnom vremenu zahtijevaju konkurentno izvršavanje što je također bilo moguće ostvariti sa Java programskim jezikom [4]. Da bi se ostvarila neophodna konkurentnost u implementaciji simulacijskog softvera korištena je paradigma paralelnog programiranja uz korištenje mehanizama višenitnosti.

Paralelno programiranje je grana programiranja gdje se koriste posebne metode za razdvajanje algoritama ili problema na njegove osnovne dijelove i izvode se istovremeno na mnogo računar ili procesorskih jezgri - tj. paralelno. Ovim načinom izvođenja programa moguće je smanjiti vrijeme koje je potrebno za pronalaženje rješenja. Paralelno programiranje se mnogo koristi u industriji, financijskim institucijama, astronomiji itd. [2]

Višenitnost je tehnika koja se koristi za implementaciju programa paralelnog programiranja. S obzirom da se jedna nit izvršava samo na jednoj procesorskoj jezgri, ostale procesorske jezgre unutar iste hardverske platforme ne mogu biti korišteni za izvršavanje zadataka. Navedeni opis važi za programe koji su pisani na takav način da ne postoji mogućnost dijeljenja glavnog zadatka u odvojene niti. Koncept višenitnosti je baziran na ideji dekompozicije zadatka na više odvojenih cjelina - podzadataka, nakon čega se vrši dodjeljivanje podzadataka pojedinim nitima te se tako omogućuje operativnom sistemu, koji radi na višejezgrenim platformama, iskorištavanje procesorskih resursa koji nisu zauzeti [3].

Za implementaciju paketa Simulator svaki od simulatora je implementiran kao zaseban proces jedne niti. Dijelovi implementiranog koda su dati u prilogu. U nastavku su data dva glavna algoritma koja su korištena za pronalazak najbližeg lifta u odnosu na trenutnu poziciju putnika pri pozivu lifta. Ovaj algoritam je opisan u pseudo jeziku i prikazan je u tabeli 2.

Tabela 2 Pseudo kod za pronalaženje najbližeg lifta

Algoritam A1

Pseudo kod za pronalaženje najbližeg lifta u odnosu na trenutnu poziciju putnika prilikom poziva lifta

```

S = broj sprata koji pozva lift
NAJBЛИЗИ_LIFT_DUZINA = 2 * BROJ_SPRATOVA
za svaki (lift L iz kolekcije LIFTOVI)
    ako (L.STANJE_VRATA == OTVORENO)
        računaj daljinu D od trenutne pozicije lifta L i sprata S
    inače
        ako (lift L planira proći sprat S) // izvršavanje Algoritma A2
            računaj daljinu D od trenutne pozicije lifta L i sprata S
        inače (D < NAJBЛИЗИ_LIFT_DUZINA)
            NAJBЛИЗИ_LIFT_DUZINA = D
            NAJBЛИЗИ_LIFT_BR = L.BrojLifta
rezultat NAJBЛИЗИ_LIFT_BR
    
```

Algoritam A2 koji je predstavljen u tabeli 3 služi za ispitivanje da li lift L koji je trenutno u vožnji, prolazi pored sprata S na kojem je došlo do poziva lifta.

Tabela 3 Pseudo kod za algoritam koji ispituje prolaska lifta

Algoritam A2
Pseudo kod za algoritam koji ispituje da li će lift L proći pored sprata S
<pre>ako (L.SMJER == GORE) ako (L.pozicija >= S) rezultat false // već je prošao sprat M = pronađi najviši sprat koji je označen u liftu L ako (M < S) rezultat false // nije planirao doći do sprata S return true inače ako (L.pozicija <= S) rezultat false // već je prošao sprat M = pronađi najniži sprat koji je označen u liftu L ako (M > S) rezultat false // nije planirao doći do sprata S rezultat true</pre>

Ostali dijelovi implementacije kao i sam kod su dati u prilogu ovog dokumenta.

5. Testiranje simulacijskog softvera

Proces testiranja simulacijskog softvera je izvršen na način da se višestrukim pokretanjem simulacije i zadavanjem različitih vrijednosti ulaznih parametara nadgledao rad sistema. U prvom testnom scenariju su zadati sljedeći parametri:

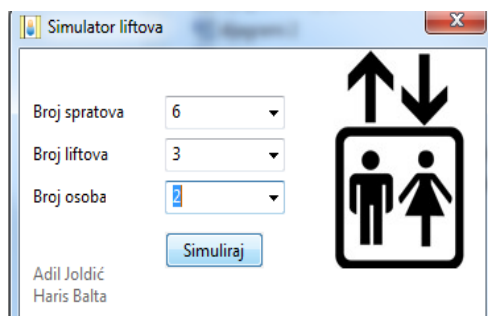
- broj spratova 10;
- broj liftova 3;
- broj putnika 3;

Nakon promatranja simulacije uočeno je slijedeće:

- da se nije jedan lift nije kretao "u prazno".
- da su svi putnici stigli na odredišni sprat.
- da niti jedan putnik nije čekao na dolazak lifta a da je bilo u isto vrijeme slobodnih liftova (tj. liftova u stanju stajanja na spratu duže od 2 s).

6. Korisničko uputstvo

Za pokretanje simulacijskog softvera neophodno je da računar na kojem se pokreće simulacija ima instaliranu Java Virtual Mashine (JVM)¹. JVM platforma omogućava izvršavanje Java kompajliranog byte koda nezavisno od platforme na kojoj se pokreće [5]. Nakon instalacije moguće je pokrenuti simulacijski softver. Pri pokretanju programa otvara se prozor (slika 8) gdje korisnik ima mogućnost da odabere vrijednosti za tri parametra simulacije. Parametri koje korisnik definiše su: broj spratova, broj liftova i broj putnika.

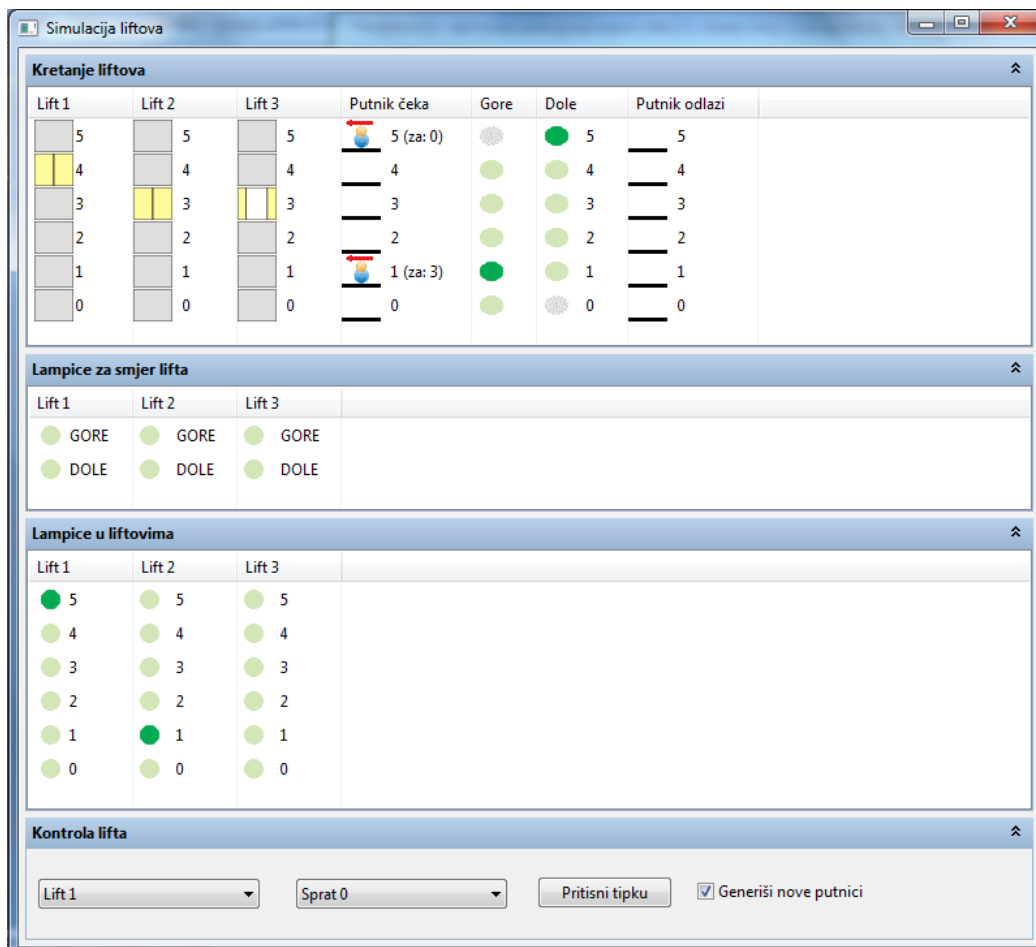


Slika 9 Prozor za odabir parametara simulacije

Parametri se prosljeđuju kao ulazne vrijednosti na osnovu kojih se generiše glavni prozor simulacijskog softvera. Potrebno je napomenuti da se ovo odvija u vremenu izvršavanja simulacije. Glavni prozor simulacije je prikazan na slici 9. Kao što se sa slike vidi postoje četiri odvojena sekcije simulacije, a to su:

1. Kretanje liftova;
2. Lampice za smjer liftova;
3. Lampice za spratove u liftovima;
4. Kontrola lifta;

¹ JVM paket se može besplatno preuzeti sa <http://java-virtual-machine.net/download.html>





Slika 10 Glavni simulacijski prozor

Sekcija 1: Kretanje liftova








Ovdje je prikazan glavni dio simulacije. Sastavne komponente su prikazani u tabeli 4.

Tabela 4. Komponente sekcije "Kretanje liftova"

Komponenta	Opis
Liftovi	Moguća stanja lifta i njihov grafički prikaz je dat u tabeli 5.
Spratovi	Broj spratova je definiran u prozoru za odabir parametara simulacije.
Dolazeći putnici	Ovdje su prikazani putnici koji na polazišnom spratu čekaju dolazak lifta. Pokraj svakoj putnika je prikazan njegov odredišni sprat.
Lampice	Ove lampice koje pokazuju odabrani smjer od strane putnika na spratovima gdje putnici čekaju dolazak lifta. Na svakom spratu se nalaze po dvije lampice (za smjer gore i smjer dole) osim prvog i posljednjem spratu. Stanje lampica su: lampica ugašena:  lampica upaljena: 
Odlazeći putnici	Ovdje su prikazani putnici koji su stigli na odredište i napuštaju sistem. Ovi putnici se prikazuju 2 sekunde.

Stanja lifta i njihov grafički prikaz u simulacijskom softveru su dati u tabeli 5.

Tabela 5. Grafički prikaz različitih stanja lifta

Opis stanja lifta	Grafički prikaz
nema lifta (lift nije spratu)	
lift stoji otvoren bez putnika	
lift stoji otvoren sa putnikom	
lift zatvoren bez putnika	
lift zatvoren sa putnikom	
lift se nalazi u poluspratu (gore ili dole) bez putnika	
lift se nalazi u poluspratu (gore ili dole) sa putnikom	

Iako grafički nije prikazano, moguće je da se više od jedne osobe generiše na istom spratu i da se više od jedne osobe nalazi u jednom liftu.

Sekcija 2: Lampica za smjer lifta

Ovdje su prikazana stanja lampica koje označavaju smjer kretanja lifta. One se aktiviraju nakon što se zatvore vrata i izvrši kalkulacija kretanja, a deaktiviraju se u pri otvaranja vrata lifta. Za svaki lift su prikazane po dvije lampice (za smjer gore i smjer dole).

Sekcija 3: Lampice u liftovima

Ovdje su prikazana stanja lampica koje označavaju odabrani sprat na kojim lift treba stati. One se aktiviraju pritiskom tipke za odabir sprata koje se nalaze u svakom liftu.

Sekcija 4: Kontrola liftova

Kontrole lifta koji se nalazi na dnu prozora omogućava ručno zadavanje komandi za odabir spratova u pojedinim liftovima i njihovo upućivanje na njih. Ova mogućnost je ostavljena tako da se može unijeti dio neizvjesnosti u simulaciju.

Aktiviranjem opcije *Generiši nove putnike* daje se mogućnost ponavljanje simulacije sa istim parametrima.