

Analiza i dizajn sistema

U nastavku je potrebno definisati sve potencijalne klase koje će se koristiti u sistemu. Za određivanje klasa koje će biti neophodne za rad sistema potrebno je koristiti specifikaciju sistema i prethodno kreirane dijagrame.

Template za jednu klasu potrebno je iskopirati onoliko puta koliko je neophodno da bi se definisale sve klase u sistemu.

Definicija klasa u sistemu

Naziv klase: „Idable“ - interfejs

Funkcionalni zahtjevi u kojima klasa učestvuje:

(FZ br. 00: ime zahtjeva)

(FZ br. 00: ime zahtjeva)

(FZ br. 00: ime zahtjeva)

(FZ br. 00: ime zahtjeva)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
id	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Click or tap here to enter text.	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Click or tap here to enter text.	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: „Korisnik sistema“ – apstraktna klasa

Funkcionalni zahtjevi u kojima klasa učestvuje:

(FZ br. 00: ime zahtjeva)

(FZ br. 00: ime zahtjeva)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
ime	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
prezime	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
datumRođenja	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
brojTelefona	int	

Naziv klase: „Korisnik sa nalogom“ – nasljeđuje klasu „Korisnik sistema“ i implementira interfejs „Idable“

Funkcionalni zahtjevi u kojima klasa učestvuje:

(Rezervacija karte)
(Ostvarivanje prava na popust)
(Otkazivanje registracije)
(Mogućnost otkazivanja rezervacije)
(Mogućnost da zatraži da mu se preporuči film)
(Mogućnost pregleda sala i odabira sjedišta pri rezervaciji)
(Mogućnost online plaćanja)
(Mogućnost povrata novca)
(Mogućnost pretrage filmova)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
eMail	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
password	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Objektno Orijentisana Analiza i Dizajn

imaPravoNaPopust	bool	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
rezervisaneKarte	List<Karta>	
bankovniRačun	Račun	

Naziv klase: „Administrator“ – nasljeđuje klasu „Korisnik sistema“, implementira interfejs „Idable“

Funkcionalni zahtjevi u kojima klasa učestvuje:

(Registracija korisnika)
 (Brisanje korisničkog računa)
 (Rezervacija karata)
 (Mogućnost pretraživanja korisnika u sistemu, po rezervisanim kartama za filmove)
 (Upravljanje korisnicima i filmovima, editovanje postojećih podataka)
 (Prikaz sadržaja aplikacije)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
eMail	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
password	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Click or tap here to enter text.	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: „Film“ – implementira interfejs „Idable“

Funkcionalni zahtjevi u kojima klasa učestvuje:

(Rezervacija karte za određeni film)
 (Otkazivanje rezervacije za određeni film)
 (Pretraga filmova)

Objektno Orijentisana Analiza i Dizajn

(Preporuka filma)

(Pretraživanje korisnika po rezervisanim kartama za filmove)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
naziv	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
zanr	Zanr	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
trajanje	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
ocjena	int	
terminiProjekcija	List<String>	

Naziv klase: „Zanr“ – enumerativna klasa

Funkcionalni zahtjevi u kojima klasa učestvuje:

(Pretraga filmova po žanru)

(FZ br. 00: ime zahtjeva)

(FZ br. 00: ime zahtjeva)

(FZ br. 00: ime zahtjeva)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
Horor	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Romantika	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
SCI-FI	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Dokumentarni		<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>

Objektno Orijentisana Analiza i Dizajn

Triler	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Krimi	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Drama	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Akcioni		<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: „Karta“ – implementira interfejs „Idable“

Funkcionalni zahtjevi u kojima klasa učestvuje:

(Rezervacija karte)
 (Mogućnost online plaćanja karte)
 (Pretraga korisnika prema rezervisanim kartama)
 (Otkazivanje rezervacije karte)
 (Mogućnost ostvarivanja popusta na kartu)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
cijena	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
imeFilma	Film	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
terminProjekcije	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
brojSale	Sala	

Naziv klase: „Sala“ – implementira interfejs „Idable“

Funkcionalni zahtjevi u kojima klasa učestvuje:

(Mogućnost odabira sjedišta u sali i termina projekcije)

(FZ br. 00: ime zahtjeva)

(FZ br. 00: ime zahtjeva)

(FZ br. 00: ime zahtjeva)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
kapacitet	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
slobodnaSjedišta	List<int>	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
jeLiZa4DXProjekcije	bool	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: „Rezervacije“ – implementira interfejs „Idable“

Funkcionalni zahtjevi u kojima klasa učestvuje:

(Rezervacija karte)

(Otkazivanje rezervacije)

(Mogućnost pretraživanja korisnika prema rezervisanim kartama)

(Mogućnost povrata novca za otkazivanje rezervacije na vrijeme)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
ime	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
prezime	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
idKorisnikaSaNalogom	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
karta	Karta	

Naziv klase: „Račun“ -implementira interfejs Idable

Funkcionalni zahtjevi u kojima klasa učestvuje:

(Mogućnost online plaćanja)
(Mogućnost povrata novca)
(Mogućnost ostvarivanja popusta)
(Mogućnost rezervacije karte)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
brojRačuna	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
datumIsteka	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
CSC	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
stanjeRačuna	double	

Naziv klase: „Gost“ – nasljeđuje klasu „Korisnik sistema“, implementira interfejs „Idable“

Funkcionalni zahtjevi u kojima klasa učestvuje:

(Mogućnost kreiranja računa)
(Prikaz sadržaja)
(FZ br. 00: ime zahtjeva)
(FZ br. 00: ime zahtjeva)

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
Click or tap here to enter text.	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

SOLID PRINCIPI:

Na prvi pogled primjetimo da je prema ovakvoj raspodjeli atributa narušen open-closed princip, međutim treba imati na umu da su uglavnom veze između klasa, posmatrano iz ugla ERD dijagrama, jedan na jedan ili jedan na više. Srećom, ovaj problem sa narušenosti open-closed principa se lako može riješiti, uzmimo za primjer korisnika sa nalogom koji bi trebao da ima na svom nalogu uvid za sve rezervisane karte. Pojavljivanje klase Karta u klasi Korisnik sa nalogom možemo izbjeći tako što pri stvaranju nove instance klase Karta ili eventualnom umetanju u neku bazu instance klase Karta kao jedan atribut možemo dodati idKorisnikaSaNalogom koji će biti tipa int (zato jer je int id), i koji će igrati ulogu foreign keya. Razlog tome je činjenica da ćemo onda lako moći iz baze izvući sve rezervisane karte za datog korisnika sastavljanjem poprilično jednostavnog upita ([pk] id = [fk] idKorisnikSaNalogom), onda ćemo te podatke moći koristiti da umetnemo u listu svih rezervisanih karata nekog konkretnog korisnika na njemu vidljivo mjesto u aplikaciji, također i ostali problemi ovog tipa se mogu riješiti na skoro identičan način kao opisani problem. Također primjetimo da imamo i klasu Račun, u ovom smislu ona nam je od važnosti zbog mogućnosti online plaćanja, ona neće narušavati open-closed princip jer nema puno funkcionalnosti, odnosno za sve moguće vrsta računa (goldenCard, silverCard, ...) će vrijediti apsolutno isti kod, ona će eventualno imati metode : „pretvoriStanjeUEure“, „jeLiUMinusu“ i slično, a poslužit će nam da prilikom rezervacije karte samo umanjimo atribut stanjeRačuna za cijenu karte, te nakon toga u nekoj drugoj klasi označimo kartu kao plaćenu i rezervisanu pozivajući neku metodu koja će insertovati određenog korisnika u bazu podataka. Dakle iako je na ovom mjestu konkretno narušen open-closed princip, njega ćemo pri implementaciji lako riješiti na već objašnjeni način. Primjetimo da smo slično nešto u radili u klasi Rezervacija gdje imamo atribut idKorisnikSaNalogom kao foreign key, čisto radi provjere sa unesenim podacima, a i radi pokazivanja primjera kako možemo izbjeći pojavljivanje drugih klasa u određenoj klasi, također bilo je dovoljno pozvati instancu klase Karta u Rezervaciju (i taj problem lako možemo riješiti, npr logično je da je za jednu kartu samo jedna rezervacija, pa je umjesto karta moglo biti samo idKarte tipa int...), jer su klasi karta nalaze sve informacije i koji je film i koja je sala i slično pa nije bilo potrebe za tim atributima.

S obzirom da nemamo metoda klasa, ne možemo puno reći o zadovoljavanju single responsibility principa, međutim primjetimo da je dosta gore opisanih klasa takvog tipa da im je glavni zadatak čuvanje podataka, pa s njima ne bi trebalo da bude problema što se tiče zadovoljavanja single responsibility principa, jedina klasa na koju bi trebalo obratiti pažnju je klasa Administrator. U klasi Administrator logično bi bilo da je moguće dodavati, editovati, brisati, preporučivati filmove, međutim od veće nam je važnosti ovdje da dodajemo, brišemo i imamo uvid u korisnike sistema, dakle da klasa Administrator ne bi imala više odgovornosti kao što je dodavanje i filmova i korisnika, možemo napraviti da se ona brine samo o korisnicima, a pošto bi bilo logično da Administrator može i dodavati filmove i brisati filmove, to lako možemo napraviti tako što kodiramo da se poseban stage u aplikaciji otvara samo određeni e-mail i password (što je e-mail i password administratora), te da se na tom stage-u nudi opcije dodavanja, brisanja i editovanja filmova čije će implemetacije biti u klasi Film, kako bi se zadovoljio single responsibility. U ovom stage-u nakon naprimjer pritiska na dugme „Dodaj“ u FilmControlleru ćemo pozvati implementirane metode za potrebne akcije (npr.

preko FilmoviManager u slučaju DaoPatterna). Dalje, ne treba očekivati od administratora neke akcije poput rezervišiKartu() i slično za takve akcije postoje pogodnije klase za njihovu implementaciju da bi se zadovoljio single responsibility princip. Sve mogućnosti administratora kojih treba da bude puno zasnivat će se na tome da se otvori poseban stage za njegove podatke. Naravno ne bi problem bio napraviti i nasljeđivanje, naprimjer da klasa administrator bude neka apstraktna a da se iz nje izvode AdministratorZaKorisnike, AdministratorZaFilmove, međutim po mom mišljenju za to nema potrebe jer je i ovo prihvatljivo rješenje. Naravno, validaciju podataka ćemo lako uraditi kroz kontrolere, ni to neće biti posao administratora, a ako bude velike potrebe za validacijom, lako možemo napraviti novu klasu Zaposlenik čija će jedina odgovornost biti validacija bankovnog računa i podataka (pod validacijom bankovnog računa podrazumjevamo i transakciju). Dakle ukoliko je problem da u klasi Administrator, koja nam jedino može biti problem (bar koliko za sada vidim) za single responsibility princip, imamo metode i za upravljanje i korisnicima i filmovima, rješenje je opisano, naravno ako nije problem puno bi lakše bilo da sve držimo u jednoj klasi te da možda zaključimo da je njen jedini responsibility upravljanje podacima. U svih slučajevima imamo rješenje za single responsibility princip.

Što se tiče Liskove zamjene, u našem primjeru neće biti takvih problema jer nemamo puno nasljeđivanja, a sva nasljeđivanja koja su izvršena imaju smisla odnosno korektna su. Iz Korisnika izvodimo vrsta korisnika pa tu nema problema.

S obzirom da nam sve klase imaju id, odnosno svaka instance bilo koje klase koja bude instancirana u bazu podataka će imati autoincrement id. Stoga smo napravili jedan interfejs Idable koji ima samo atribut id, što nam govori da je zadovoljen The Interface Segregation Principle. Znamo da je zadovoljen ovaj princip zato što ima samo jedan atribut koji je sigurno zajednički za sve one klase koje ga nasljeđuju. Također interfejs će imati metode setId i getId. Čak i u slučaju klase Račun poželjno je imati id, iako taj id nije onaj id bankovne kartice računa i slično, to je urađeno zbog evidencije u bazi svih ulaznih podataka u aplikaciju.

The Dependency Inversion Principle je zadovoljen jer imamo samo jednu apstraktnu klasu koju nema smisla nikada pozvati kao instancu te klase jer je previše apstraktna. To smo postigli izvođenjem klase Gost iz apstraktne klase Korisnik sistema. Gost nema nikakvih dodatnih atributa u odnosu na apstraktnu klase, samo je dodato da su funkcionalnosti u kojim on učestvuje prikaz sadržaja i mogućnost kreiranja računa, dok su namjerno ostavljene prazne mogućnosti za apstraktnu klasu Korisnik sistema, da bi se osigurali da tu klasu nema smisla nikada samostalno pozvati. Dakle ako želimo pozvati neku instancu klase koja nije ni registrovani korisnik ni administrator, pozvat ćemo instancu klase Gost koje je manje apstraktna od klase Korisnik sistema, ali opet ne toliko apstraktna da je ne možemo instancirati.