Opis klasa i SOLID principi

Klase potrebne za našu aplikaciju izdvojit ćemo iz opisa aplikacije.

"Pri pokretanju aplikacije korisniku će se otvoriti login prozor, tj. bit će zatraženo unošenje korisničkog imena i lozinke. Ukoliko korisnik bude prepoznat kao student bit će mu ponuđeno generisanje rasporeda za predstojeći semestar, te potom izmjena i prihvatanje rasporeda. U slučaju da se loguje student koji je prethodno generisao i prihvatio raspored, neće mu biti ponuđeno ponovno generisanje, već će biti ispisan onaj raspored koji je student potvrdio.

Ukoliko se osoba koja koristi aplikaciju prijavi kao profesor, bit će u mogućnosti da odredi predmet, ukoliko predaje više predmeta, te termine predavanja i vježbi za svoj predmet vodeći računa o dostupnosti sala. Što se tiče određivanja termina predavanja i vježbi, svaki profesor će trebati odrediti broj termina, vrijeme i salu održavanja kako bi svi student mogli prisustvovati (s obzirom na to da postoji ograničenje na broj studenata koji može biti na vježbama).

(…)

Administrator će biti u mogućnosti da dodaje nove studente, dodjeljuje im odsjek, semestar i predmete. Potom dodaje nove profesore i dodjeljuje im predmet ili više njih. U slučaju promjene stanja opreme u sali, omogućava mu se promjena kapaciteta sale ili dodavanje nove sale. U slučaju izmjene nastavnog plana i programa, administrator će moći izmijeniti, dodati ili izbrisati bilo koji od podataka baze."

Na osnovu opisa teme izdvajamo sljedeće klase: Korisnik, Student, Profesor, Predmet, Termin, Sala, Administrator.

Korisnik

> Klasa Korisnik ima attribute: ime (String), prezime (String), username (String), eMail (String), password (String) i termini (List<Termin>). Metode ove klase su, pored setter i gettera, dodajTermin (void funkcija koja kao parameter prima objekat tipa Termin) i ukloniTermin (void funkcija koja kao parameter prima objekat tipa Termin).

Student

Klasa Student izvedena je iz klase Korisnik. Dodatni atributi koje ova klasa ima su odsjek (String) i semestar (int). Atribut termini, klase Korisnik, bit će popunjeni terminima predavanja i tutorijala predmeta koji su dodijeljeni određenom studentu. Metode ove klase su, pored metoda naslijeđenih iz klase Korisnik, potrebni getteri i setter.

Profesor

Klasa Profesor izvedena je iz klase Korisnik. Dodatni atribut ove klase je predmeti (List<Predmet>), obzirom da jedan professor može predavati više predmeta. Atribut termini, klase Korisnik, bit će popunjeni terminima predavanja i tutorijala koje profesor bude kreirao za svoje predmete. Pored settera i gettera za listu predmeta, klasa Profesor ima metodu dodajTerminUBazu (void funkcija čiji je parametar objekat tipa Termin).

Predmet

Atributi klase Predmet su: nazivPredmeta (String), odsjek (String), semestar (int) i profesor (Profesor). Metode ove klase su svi potrebni setteri i getteri.

Termin

Atributi klase Termin su: predmet (Predmet), dan (moment.Day), pocetnoVrijeme (Enum), krajnjeVrijeme (Enum), sala (Sala), tipTermina (Enum). Pored gettera i settera, metoda klase Termin je jeLiPopunjenTermin (Boolean funkcija bez parametara).

Atribut tipTermina je atribut prema kojem će se razlikovati termini predavanja i tutorijala, obzirom da nema potrebe za izvođenjem klasa Predavanje i Tutorijal iz klase Termini.

Sala

Klasa Sala sadrži atribute naziv (String) i kapacitet (int), a od metoda settere i gettere za navedene atribute. Korištena je u klasi Termin i pomoću metode jeLiPopunjenTermin provjeravat će se da li je specifični termin predavanja ili tutorijala popunjen, tj. da li kapacitet sale u kojoj se termin podržava nove studente.

Administrator

Atributi klase Administrator su: profesori (List<Profesor>), studenti (List<Student>), termini (List<Termin>), sale (List<Sala>), predmeti (List<Predmet>). Pored gettera i settera za navedene atribute, klasa Administrator ima metodu dodajTerminUBazu (void funkcija čiji je parametar objekat tipa Termin).

Sve prethodno navedene klase korištene su u klasi Administrator obzirom da administrator ima uvid u sve podatke u bazi podataka i mogućnosti dodavanja novih, kao i uređivanje postojećih, profesora, studenata, termina, sala i predmeta.

U nastavku ćemo razmotriti SOLID principe

Single Responsibility Principle

Odgovornost navedenog principa je da svaka od klasa korištenih u određenoj aplikaciji ima isključivo jednu svrhu. Ukoliko neka od klasa izvršava previše različitih tipova akcija u aplikaciji, ovaj princip to tumači kao „razlog za promjenu“.

Iz prethodno navedenog opisa klasa zaključujemo da je pomenuti princip zadovoljen.

Open/Closed Principle

OC princip savjetuje dizajn sistema takav da buduće promjene ne uzrokuju mnogo modifikacija. Moguće buduće promjene su npr. dodavanje klase Asistent izvedene iz klase Korisnik, uvođenje dodatnih specifikacija koje će zahtijevati razlikovanje termina predavanja i tutorijala, samim tim izvođenje klasa Predavanje i Tutorijal iz klase Termin ili uvođenje izbornih predmeta, te izvođenje takve klase iz trenutne klase Predmet.

Sve navedene promjene neće znatno promijeniti dosadašnji dizajn Sistema te zaključujemo da je navedeni princip zadovoljen. Sve eventualne promjene desit će se u klasi Administrator koja jedino ima pristup svim podacima koji se nalaze u bazi.

Liskov Substitution Principle

Liskov Substitution princip zahtijeva da nasljeđivanje bude ispravno implementirano, odnosno da je na svim mjestima na kojima se koristi osnovni objekat moguće iskoristiti i izvedeni objekat, a da takvo nešto ima smisla.

Navedeni pricip jeste zadovoljen prema trenutim definicijama klasa obzirom da je jedino nasljeđivanje nasljeđivanje iz klase Korisnik koja je apstraktna. Međutim, obje izvedene klase, Student i Profesor, imaju upotpunosti različite pristupe i akcije u sistemu te nikada neće doći do upotrebe bilo koje od ove dvije klase na mjestima gdje je korišten osnovni objekat.

Interface Segregation Principle

Princip Interface Segregation zahtijeva da i svi interfejsi zadovoljavaju princip Single Responsibility, odnosno da svaki interfejs obavlja samo jednu vrstu akcija.

Interfejs korišten u sistemu je IBazaDodatak koji koriste klase Profesor i Administrator, a jedinja njegova akcija je ispravno dodavanje termina u bazu podataka. Zaključujemo da je navedeni princip zadovoljen.

Dependency Inversion Principle

Dependency Inversion princip zahtijeva da pri nasljeđivanju od strane više klasa bazna klasa uvijek bude apstraktna.

Jedino nasljeđivanje prisutno u sistemu je nasljeđivanje klasa Student i Profesor iz klase Korisnik koja je apstraktna, pa zaključujemo da je spomenuti princip zadovoljen.