ANALIZA SISTEMA

TEMA : STOMATOLOGIJA

**ZAPOSLENIK**

Atributi:

* ime (String)
* prezime(String)
* username(String)
* password(String)

Konstruktori:

* konstruktor sa četiri parametra

Metode:

* getteri
* setter
* toString
* checkPassword(String)

**DOKTOR**

-ovu klasu nasljeđuje klasa SpecDoktor.

Atributi:

* odjel(String)
* doktor(Uposlenik)

Konstruktori:

* kostruktor bez parametara
* Konstruktor sa pet parametra

Metode:

* getteri
* setter

**MEDSESTRA**

Atributi:

* medSestra(Uposlenik)

Konstruktori:

* konstruktor sa četiri parametra

Metode:

* spisakPregleda
* spisakDoktora
* getteri
* setteri

**SPECDOKTOR**

Atributi:

* odjel(String)

Konstruktori:

* konstruktor sa pet parametara

Metode:

* getteri
* setter
* toString

**DIREKTOR**

Atributi:

* direktor(Uposlenik)

Konstruktori:

* konstruktor sa četiri parametra.

Metode:

* getteri
* setter

**PACIJENT**

Atributi:

* ime(String)
* prezime(String)
* jmbg(String)
* brojKartona(Integer)
* datumRodjenja(LocalDate)
* nalazi(Nalazi)

Konstruktori:

* konstruktor sa pet parametara

Metode:

* getteri
* setter
* dodavanjePregleda(Pregled)

**PREGLED**

Atributi:

* doctor(Doktor)
* pacijent(Pacijent)
* datum(LocalDate)
* vrijeme(LocalTime)
* dijagnoza(String)

Konstruktori:

* konstruktor sa pet parametara

Metode:

* getteri
* setter

**KARTON**

Atributi:

* brojKartona(Integer)
* pacijent(Pacijent)
* usluge(ArrayList<Usluga>)

Konstruktori:

* kostruktor sa jednim parametrom
* konstruktor sa dva parametra

Metode:

* dodajPregled(Pregled)
* getteri
* setter

**USLUGA**

-Ovu klasu nasljeđuje klasa Pregled.

Atribut:

* naziv(String)
* cijena(Integer)
* opis(String)

Konstruktor:

* konstruktor sa tri parametra

Metode:

* getteri
* setteri

**ZUBEX**

Atributi:

* doktori(Array<Doktor>)
* pacijenti(Array<Pacijent>)
* usluge(Array<Usluga>)

Metode:

* dodajPacijenta(Pacijent)
* obrišiPacijenta(Pacijent)
* ažurirajPacijenta(Pacijent)
* dodajDoktora(Doktor)
* obrišiDoktora(Doktor)
* ažurirajDoktora(Doktor)
* spisakPregleda(Array<Pregled>)

Veze između klasa:

* Zaposlenik sa Doktor, Medsestra I Direktor – generalizacija, jer Doktor, Medsestra i Direktor su izvedene iz Zaposlenik
* Doktor i Specdoktor- generalizacija, jer Specdoktor je izveden iz Doktor
* Zubex i Usluga – kompozicija, jer Usluga ne može postojati bez Zubex-a
* Pacijent i Pregled – agregacija ako se Pregled izbriše, Pacijent može postojati
* Karton i Pacijent – kompozicija, jer Karton ne može postojati bez Pacijent-a
* Karton i Pregled - agregacija ako se Pregled izbriše, Karton može postojati
* Zubex i Pacijent - agregacija ako se Pacijent izbriše, Zubex može postojati
* Zubex i Doktor - agregacija, ako se Doktor izbriše, Zubex može postojati

SOLID princip

* **S**ingle Responsibility Principle – Princip pojedinačne odgovornosti

Mislimo da je ovaj princip prilično zadovoljen, jer smo pokušali da sve koncepte odvojimo u njihove vlastite klase tako da svaka klasa ima jedan i samo jedan razlog za promjenu.

* **O**pen Closed Principle – Otvoreno zatvoren princip

Ovaj princip nije narušen, jer mislimo da entiteti softvera bi trebali biti otvoreni za nadogradnju, ali zatvoreni za modifikacije.

* **L**iskov Substitution Principle – Liskov princip zamjene

Mislimo da ovaj princip nije narušen, jer nijedan podtip ne može biti zamijenjen nekim osnovnim tipom.

* **I**nterface Segregation Principle – Princip izoliranja interfejsa

Misllimo da ovaj princip nije narušen, jer korisnik nema veliki izbor metoda na raspolaganju i one su prilično dozirane njegovim potrebama. Sistem ne posjeduje “debele” klase. Tako da nije potrebno korisnika štiti od metoda koje im ne trebaju i od poznavanja implementacije objekta kojeg koristi.

* **D**ependency Inversion Principle – Princip inverzije ovisnosti

Mislimo da ovaj princip nije narušen, jer sistem ne ovisi od konkretnih klasa. Prilikom nasljeđivanja osnovne klase su apstraktne.