SOLID PRINCIPI

PRINCIP S

⇒ Single Responsibility Principle: Svaka klasa treba imati samo jednu ulogu.

Svaka klasa u aplikaciji ima samo jednu odgovornost tj. ima samo jedan razlog za promjenu, čime je zadovoljen ovaj princip. Jedina klasa koja bi mogla predstavljati problem jeste klasa *Donor*. Naime, ova klasa je u početku imala mnogo više metoda od ostalih klasa i od trenutne klase *Donor* koja je sada predstavljena. Mnogo smo se trudile da sve te metode jasno razdvojimo, kako u interfejse tako i u ostale klase. Upravo iz tih razloga klasa *Donor* implementira najveći broj interfejsa i zbog toga je uvedena i klasa *ZdravstevnaHistorijaIzvještaj*. Njeno postojanje omogućava promjenu forme izvještaja, što ne bismo mogli uraditi da je ostala kao metoda u klasi *Donor*, gdje bi prouzrokovalo još jednu promjenu tj. promjenu u klasi *ZdravstvenaHistorija*. Metode koje su ostale u ovoj klasi služe za promjenu njenih atributa i unapređenje klase, što ne narušava ovaj princip. Ostale klase imaju jednu do dvije metode.

PRINCIP O

⇒ Open/Closed Principle: Klasa treba biti otvorena za nadogradnje, ali zatvorena za modifikacije.

Princip nije narušen, iako u aplikaciji imamo mnogo klasa koje koriste preostale klase preko atributa i metoda. Međutim, svaka upotreba druge klase od strane prve je čisto informativnog tipa tj. u svrhu dobavljanja informacija te klase, a ne njene promjene.

PRINCIP L

⇒ Liskov Substitution Principle: Svaka osnovna klasa treba biti zamjenjiva svim svojim podtipovima bez da to utječe na ispravnost rada programa.

U aplikaciji imamo veoma malo nasljeđivanja, zbog čega nije bilo teško zadovoljiti ovaj princip. Klase *ObavijestDonor* i *ObavijestBolnica* su izvedene iz klase *Obavijest*, koja je apstraktna klasa. S obzirom da je klasa *Obavijest* apstraktna, nju je moguće na svakom mjestu zamijeniti njenim podtipovima, a da se ne narušava ispravnost rada programa.

PRINCIP I

⇒ Interface Segregation Principle: Bolje je imati više specifičnih interfejsa, nego jedan generalizovani.

Ovaj princip je zadovoljen. U aplikaciji imamo 5 interfejsa, gdje svaki predstavlja zasebnu funkcionalnost sa svojim operacijama. Jedan od glavnih razloga uvođenja interfejsa u našu aplikaciju jeste da se ne bi narušio prvi princip, čime postižemo da su zadovoljena oba principa.

PRINCIP D

⇒ Dependency Inversion Principle: Sistem klasa i njegovo funkcionisanje treba ovisiti o apstrakcijama, a ne o konkretnim implementacijama.

U našoj aplikaciji klase ne ovise od konkretnih klasa, nego od apstrakcija i interfejsa i time je ovaj princip zadovoljen. To smo postigle tako što smo klasu *Obavijest* proglasile apstraktnom klasom i dodale interfejse koje realiziraju određene klase.