Refaktoring

Refaktoring (eng. refactoring) je Martin Fowler definirao kao "promjene interne strukture softvera da bi bio lakši za razumijevanje i jeftiniji za modificiranje bez promjene njegovog ponašanja" (Fowler 1999). Riječ "refactoring" u modernom programiranju nastala je od originalne riječi "factoring" koju je uveo inženjer Larry Constantine u strukturiranom programiranju, a koja referencira na dekompoziciju programa u konsititucione dijelove onoliko koliko je to moguće (Yourdon I Constantine 1979).

Refaktoring se može definirati i kao tehnika restrukturiranja koda na disciplinirani način. To je iterativni način poboljšanja koda.

Refaktoring na osnovu kataloga refaktoringa

1. Rename a variable with a clearer or more informative name (Preimenovati varijablu sa više informativnim imenom) - Kada ime varijable nije jasno potrebno ga je promijeniti. Isti zahtjev se primjenjuje na konstante, klase i rutine. Prošli smo još jednom kroz kod i promijenili imena varijabli gdje je to bilo potrebno tako da sve varijable imaju informativno ime . Promjene su odrađene u klasi Obrazovanje gdje je public string p reimenovano u public string ProcesiranjeZahtjeva. To nam moze biti od koristi jer iz naziva p nista ne saznajemo o navedenom atributu.

2. Replace a magic number with a named constant (Zamijeniti 'magične' brojeve sa imenovanom konstantom) – Ovaj refaktoring se koristi ako u kodu imamo neke pozate konstane ili tzv. „magične" brojeve, najčešće je to kod matematičkih ili logičkih izraza. Zbog bolje čitljivosti koda ovaj refaktoring predlaže pravljenje konstantnih tipova.

3. A class doesn't do very much (Klasa ne radi puno) – Kada klasa ne radi mnogo toga potrebno je provjeriti da li su sve odgovornosti klase dodijljene drugim klasama i eliminirati klasu u potpunosti. Klasa Biro ne radi ništa tako da smo je eliminisali pri čemu smo morali dodati atribut ZahtjevZaPosao u klasi TrzisteRada u tom atributu cemo smjestati sve zahtjeve za posao ka odedjenim poslovnim jedinicima i tako smo ispostovali refaktoring.

4. Code is duplicated (Kod je dupliciran) – Dupliciran kod predstavlja većinom prvi faktor greške u dizajnu jer zahtjeva paralelnu modifikaciju – kada se urade promjene na jednom mjestu, moramo raditi promjene i na drugom mjestu. U klasi KartonViewModel kako ne bi duplicirali kod za izbjeglicu koja svoj karton zeli prebaciti na drugo osiguranje dodana je metoda IspisiSveIzKartonaUDrugiKarton() koja ce uciniti rečeno, a tako necemo duplirati kod već pozvati samo navedenu metodu.

5. Replace Error Code with Exception – ukoliko metoda vraća poseban kod kako bi ukazala na grešku, potrebno je umjesto tog posebnog koda baciti izuzetak koji će ukazati da je došlo do greške. Veći dio koda je već bio prilagođen ovom principu. Svaka metoda koja

je imala dio koda koji vraća neku specijalnu oznaku je modifikovana tako da baca izuzetak koji ukazuje na grešku koja je nastala.

6. Remove Setting Method – ukoliko se neko polje ne smije mijenjati nakon što je inicijalno postavljena njegova vrijednost, onda je potrebno ukloniti setter za to polje. Veći dio koda je već bio prilagođen ovom principu. U nekoliko klasa su uklonjeni setteri za ona polja koja iz različitih razloga ne bi smjeli mijenjati svoju vrijednost npr. u klasi Zdravstvo uklonjen je setter za Osiguranje.

Refaktoring pomoću dizajn paterna

1. Implementiran je refaktoring pomoću Singleton paterna.Uloga Singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. Postoji više objekata koje je potrebno samo jednom instancirati i nad kojim je potrebna jedinstvena kontrola pristupa. Neki od njih su: thread pools (skupina niti), objekti koji upravljaju setovanjem registara, objekti koji se koriste za logiranje, objekti koji se koriste kao drajveri za razne uređaje kao što su printeri i grafičke kartice. Instanciranje više nego jednom navedenih objekata mogu se prouzrokovati problemi kao što su nekorektno ponašanje programa, neadekvantno korištenje resursa ili nekonzistentan rezultat. U aplikaciji je ovaj patern iskorišten u klasi Registracija prilikom instanciranja tacno jedne osobe koja popunjava registraciju izbjeglice.

2. Implementiran je refaktoring pomoću Proxy paterna. On spada u strukturalne paterne. Namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. Proxy je obično mali javni surogat objekat koji predstavlja kompleksni objekat čija aktivizacija se postiže na osnovu postavljenih pravila. Proxy patern rješava probleme kada se objekt ne može instancirati direktno (npr. zbog restrikcije pristupa). Ovaj patern je implementiran prilikom pristupa korisnicima, gdje se provjerava da li korisnik postoji ili ne. Na osnovu ovoga je urađena kontrola pristupa, jer korisnik ne može nastaviti sa korištenjem aplikacije ukoliko njegov password nije verifikovan i također je iskorišteno da sistemom moze upravljati vise korisnika koji su zaduzeni za rad sa izbjeglicama i popunavanju svih potrebnih podataka.