Elektrotehnički fakultet u Sarajevu
Predmet: Objektno orijentisana analiza i dizajn
Profesorica: prof. dr dipl. ing. Dženana Đonko
Demonstratorica: Jasmina Bajramović
Akademska godina 2017/2018.

Projekat: “Av-Mau Azil grada Sarajeva”
Tim: CodeX - Ajna Zatrić, Edin Avdić, Nadir Avdagić

Refaktoring koda

(na osnovu kataloga refaktoringa i primjenom dizajn paterna)

U dokumentaciji koja slijedi pokazana je primjena tehnika restruktuiranja koda na disciplinaran način. Kôd projekta “AvMau azil” je iterativno poboljšan na osnovu kataloga refaktoringa [Fowler, 1999] sa indikacijama kada je refaktoring potreban. Korišteno je 6 modifikacija na osnovu kataloga za refaktoring i 3 refaktoringa primjenom dizajn paterna i to: Singleton (iz grupe kreacijskih paterna), Intepreter (iz grupe paterna ponašanja) i Composite (iz grupe strukturalnih paterna).

Tekst zadatka, preuzeto sa C2 OOAD- stranice predmeta : *Uraditi refaktoring koda na osnovu kataloga refaktoringa (analizirati mogućnost primjene 6 refaktoringa). Analizirati mogućnosti refaktoringa pomoću dizajn paterna. U slučaju da se u kodu ne može pronaći 3 refaktoringa u dizajn paterne (pri čemu svaki mora biti iz različite kategorije) osmisliti nove korisničke zahtjeve čija implementacija zahtjeva dizajn paterne. Napisati dokumentaciju o učinjenom refaktoringu, obrazloženje zašto je urađen i šta se postiglo.*

Indikacija 1: Klase se veoma malo razlikuju pa nije potrebno nasljeđivanje.

“Collapse hierarchy - collapse a superclass and subclass if their implementations are very similar.”

Obrazloženje zašto je refaktoring urađen i šta se time postiglo: Obzirom da je AvMau azil namijenjen isključivo dvijema vrstama životinja: mačkama i psima, i da smo primjetili da su klase “Pas” i “Mačka” izrazito slične u pogledu tipa podataka koje pohranjuju, odlučili smo ukinuti ovu hijerarhiju i osloniti se na boolean ili enum tip podataka u klasi Životinja (Enum VrstaZivotinje - {Pas, Macka}, odnosno Bolean VrstaZivotinje). Postignuto je pojednostavljenje koda.

Indikacija 2: Algoritam je isuviše kompleksan. Zamijeniti kompleksni algoritam sa jednostavnim. “Substitute a simple algorithm for a complex algorithm.”

Obrazloženje zašto je refaktoring urađen i šta se time postiglo: Do znaka upozorenja da smo došli, primjetivši da u datom dijelu koda imamo isuviše komentara. To je ukazivalo na još jedan znak za refaktoring: “Comments are used to explain difficult code (Komentari se koriste da objasne težak kod)”. U Admin.xaml.cs fajlu kod metode Slikaj_kamerom bilo je više komentara. To je nagovijestilo da se funkcija treba zamijeniti jednostavnijom funkcijom istih funkcionalnosti, što smo i uradili. Ovom modifikacijom se postiglo pojednostavljivanje koda.

```
private async void Uslikaj_kamerom(object sender, RoutedEventArgs e)
{

    var UslikajUI = new CameraCaptureUI();
    UslikajUI.PhotoSettings.Format = CameraCaptureUIPhotoFormat.Jpeg;
    UslikajUI.PhotoSettings.CroppedSizeInPixels = new Size(200, 200);
    StorageFile slika = await UslikajUI.CaptureFileAsync(CameraCaptureUIMode.Photo);


    IRandomAccessStream stream = await slika.OpenAsync(FileAccessMode.Read);
    BitmapDecoder decoder = await BitmapDecoder.CreateAsync(stream);
    SoftwareBitmap softwareBitmap = await decoder.GetSoftwareBitmapAsync();

    SoftwareBitmap softwareBitmapBGR8 = SoftwareBitmap.Convert(softwareBitmap,
    BitmapPixelFormat.Bgra8,
    BitmapAlphaMode.Premultiplied);

    SoftwareBitmapSource bitmapSource = new SoftwareBitmapSource();
    await bitmapSource.SetBitmapAsync(softwareBitmapBGR8);

    polje_za_sliku.Source = bitmapSource;

}
```

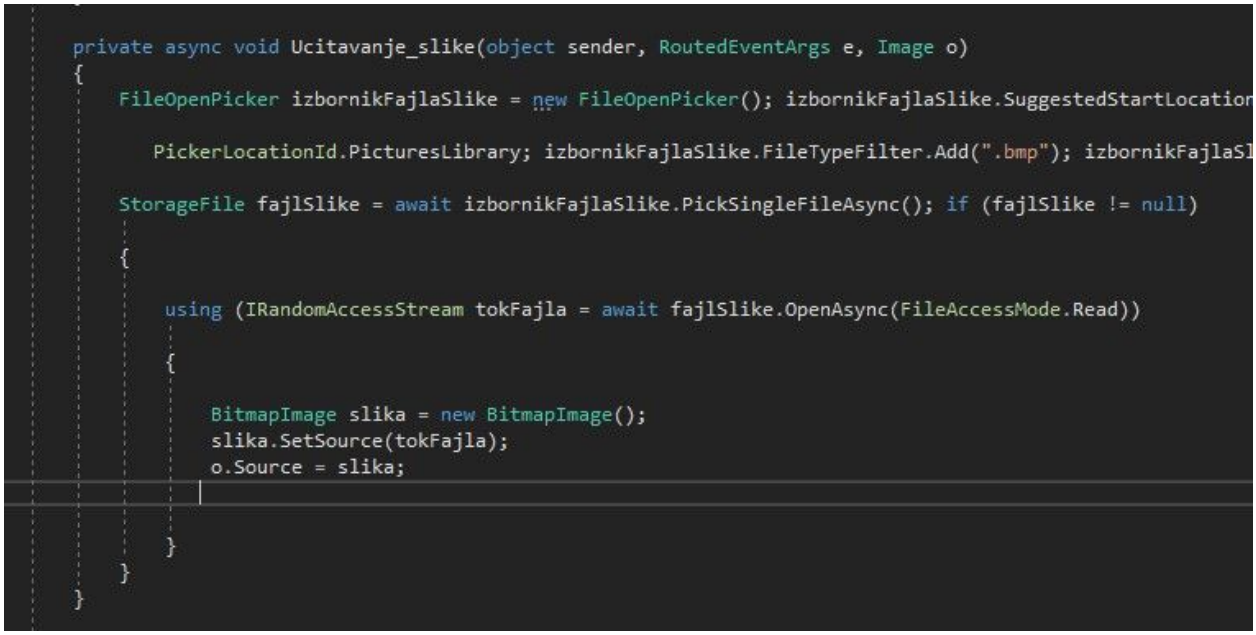
Indikacija 3: Rutina ima loše - neopisno ime. Ako rutina ima loše ime, potrebno joj je promijeniti ime u definiciji i na mjestima poziva. “A routine has a poor name”

Obrazloženje zašto je refaktoring urađen i šta se time postiglo: Analizom koda došli smo do zaključka da ima par metoda koje imaju nejasno i zbunjujuće ime u pogledu onoga što rade (na primjer, metoda “SlikajKamerom” prvobitno je imala dvosmislen naziv “Kamera”). Ime metode smo promijenili u definiciji i na mjestima poziva. Ovom modifikacijom je poboljšana dalja nadogradnja i čitljivost i razumljivost koda naročito među članovima tima koji neovisno rade na projektu.

Indikacija 4: Kod je dupliciran.”Code is duplicated”

Obrazloženje zašto je refaktoring urađen i šta se time postiglo : Imali smo dvije odvojene funkcije za dodavanje slike za životinju i dodavanje slike za korisnika. Kreirana je jedna funkcija uz minimalne modifikacije, koja se potom poziva u oba

slučaja. Funkciji smo dodali i dodatni parametar “Image Object”. Postignuto je poštovanje DRY principa i činjenice da ponavljanje koda sigurno predstavlja grešku u dizajnu.



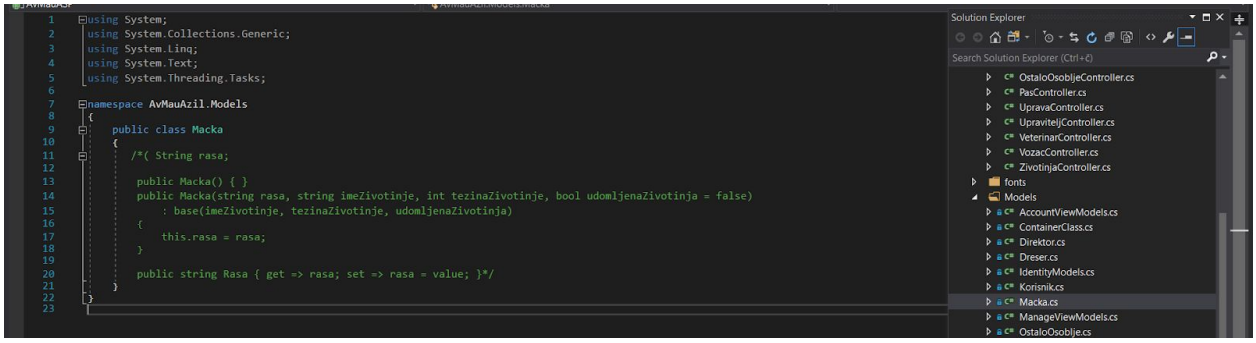
```
private async void Ucitavanje_slike(object sender, RoutedEventArgs e, Image o)
{
    FileOpenPicker izbornikFajlaSlike = new FileOpenPicker(); izbornikFajlaSlike.SuggestedStartLocation
       PickerLocationId.PicturesLibrary; izbornikFajlaSlike.FileTypeFilter.Add(".bmp"); izbornikFajlaSl
    StorageFile fajlSlike = await izbornikFajlaSlike.PickSingleFileAsync(); if (fajlSlike != null)
    {
       using (IRandomAccessStream tokFajla = await fajlSlike.OpenAsync(FileAccessMode.Read))
       {
          BitmapImage slika = new BitmapImage();
          slika.SetSource(tokFajla);
          o.Source = slika;
       }
    }
}
```

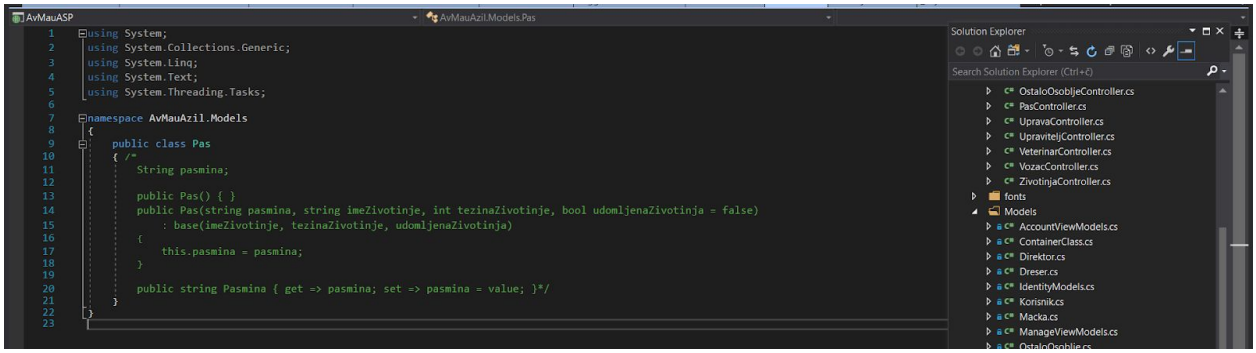
Indikacija 5: Podaci su javni. “Data members are public”.

Obrazloženje zašto je refaktoring urađen i šta se time postiglo: Metodu ne koriste drugi modeli a vidljivost je bila public. Refactoring je primjenjen tako što je metoda proglašena privatnom. Ovakvi nepotrebno javni podaci predstavljaju nejasnoću između interfejsa i implementacije. Također, oni ugrožavaju enkapsulaciju i limitiraju buduću fleksibilnost realizirane aplikacije. Ovim refaktoringom se postigla upravo dobro razgraničenje interfejsa i implementacije i buduća fleksibilnost.

Indikacija 6: Klasa ne radi puno. “ A class doesn’t do very much”

Obrazloženje zašto je refaktoring urađen i šta se time postiglo: Ponekad rezultat refaktoringa koda jeste činjenica da klasa nema mnogo da radi. U našem slučaju smo ukidanjem hijerarhije klasa Životinja -> Pas i Životinja->Mačka primjetili da su sve odgovornosti klasa Pas i Mačka dodijeljene klasi “Životinja” , pa smo ove klase eliminirali u potpunosti. (Ostavljene su u projektu i zakomentarisane kako bi se video uzrok refaktoringa).





Refaktoring u kodu primjenom dizajn paterna

(Kreacijski patern)
(Strukturni patern)
(Bihevijoralni - patern ponašanja)