YZM-2118 YAZILIM MİMARİSİ VE TASARIMI FİNAL PROJE ÖDEVİ

Manisa Celal Bayar Üniversitesi / Yazılım Mühendisliği

192803079 Süha Aşkın GÜNDÜZ (Mühendislik Tamamlama - 2. Öğretim)

2020

14.05.2020

Oyuncu - Dizi Eşleştirme Uygulaması (Observer Pattern)

Süha Aşkın Gündüz (192803079)

# Projenin Amacı:

Bir nesnenin değişikliğinden farklı nesneler etkilenecek ise observer deseni kullanılması gerekmektedir. Oyuncu - dizi eşleştirme uygulamamızda, oyuncu isimleri değişse bile dizi sınıflarından aynı şekilde sorgulama yapmaya devam ediyoruz ve eşleşen oyuncu olduğunda bu dizi bilgisini ekrana basmaktayız. Bu sayede sorgulanacak veri değişikliğine gidildiğinde diğer sınıflar üzerinde revize işlemi yapmamıza gerek kalmıyor ve uygulamamızı daha sürdürülebilir bir hale getiriyoruz.

# Proje Detayı ve Kapsamı:

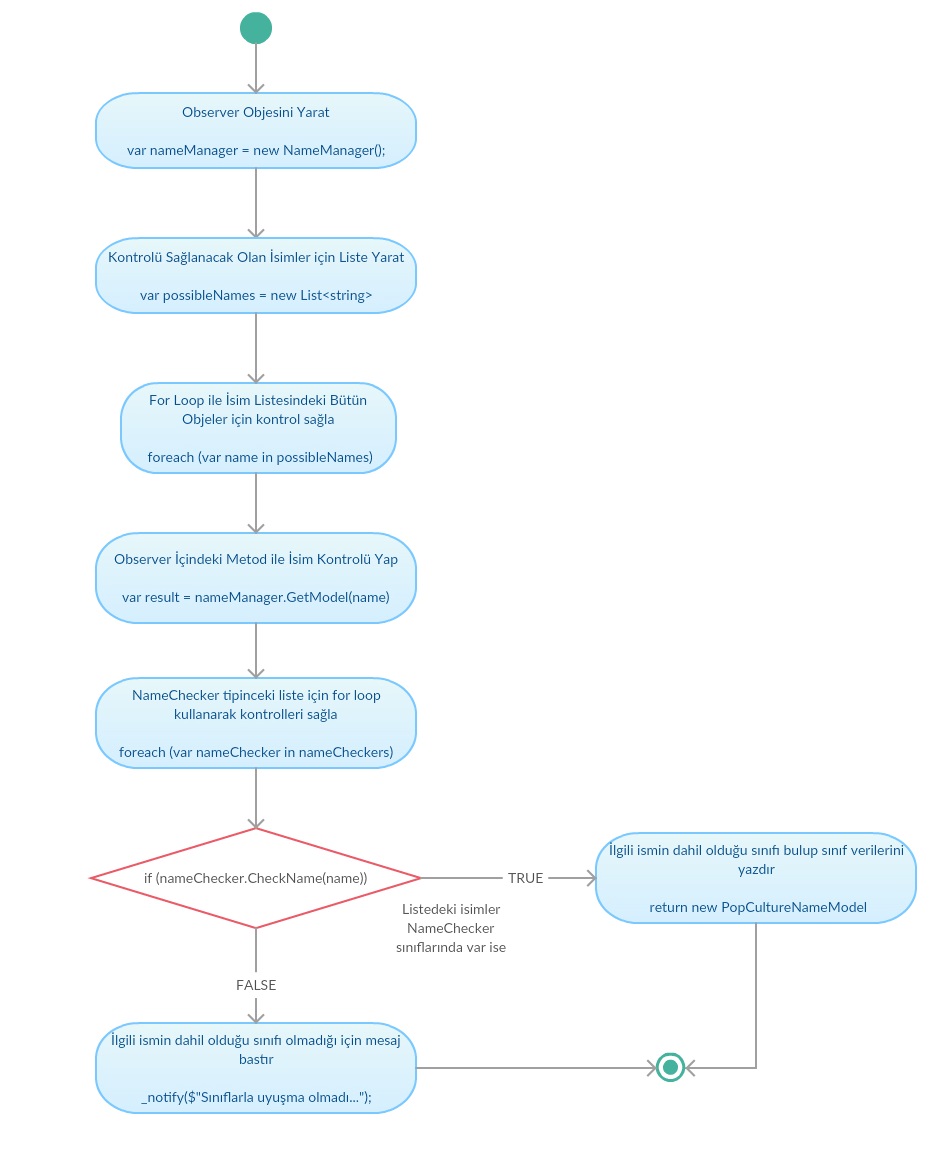
Observer (Gözlemci) Pattern bir nesnenin değişikliğinden dolayı farklı nesneler etkilenecek ise kullanılması gereken bir tasarımdır. Yani one - to - many olayını destekleyen bir tasarım desenidir. Oyuncu - dizi eşleştirme uygulamamızda bir grup oyuncu isimlerini tanımlıyoruz. Bu oyuncu isimlerini dizi sınıfları altında bulunan oyuncu isimleri ile kontrol ettirip kullanıcıya geri dönüş sağlıyoruz.

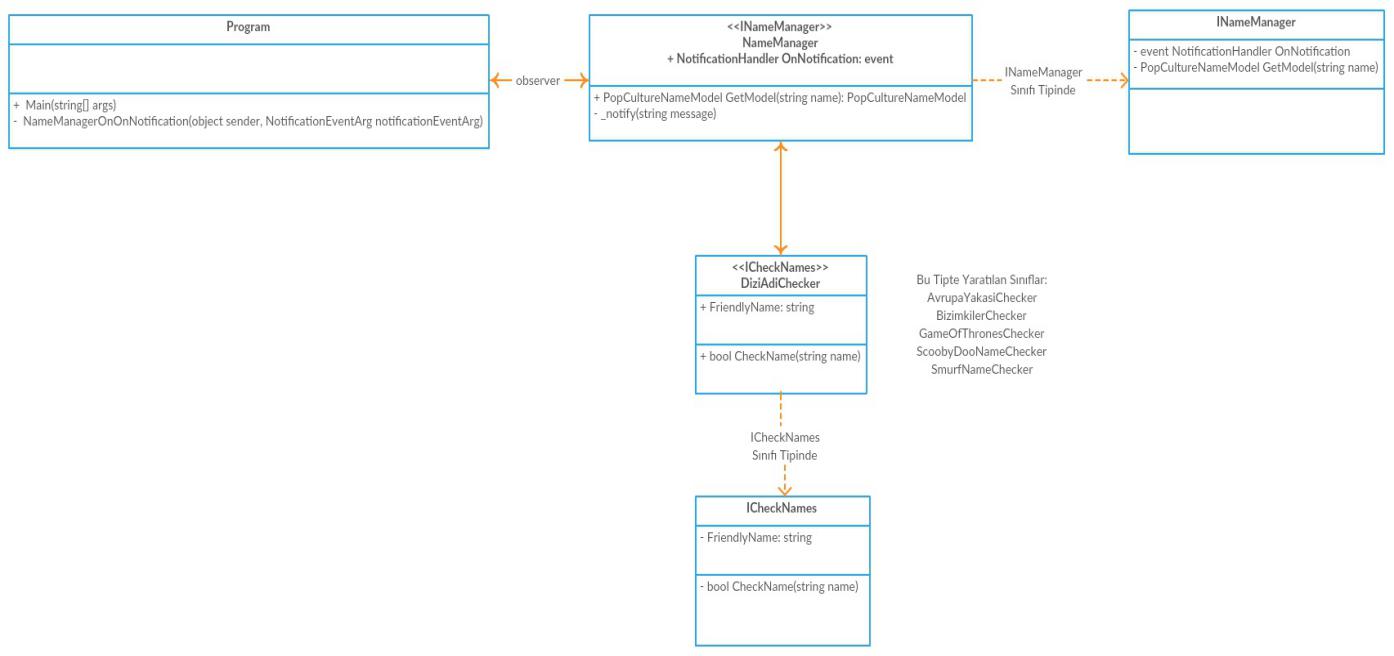
Öncelikli olarak dizi sınıfları içerisinde bu dizilere ait oyuncu karakter tanımlaması gerçekleştiriyoruz. Bu sınıfları INameChecker classımızda olduğu gibi aynı türden classla yaratıyoruz çünkü ilgili sınıfların isimlerini almamız gerekmektedir. INameChecker isimli class tipinde interface içeriği tanımlıyoruz ve bu obje kontroller esnasında true yada false dönerek bize sonucu iletiyor.

Kontrol sürecini ise NameManager classı altında düzenliyoruz. Bu class içerisinde kontrol sağlanacak olan listeleri oluşturuyoruz. Dizi isimlerini kontrol etmek için kullanılacak objeyi yaratıp referans alınacak classları new metodu ile tek tek ekleme yapıyoruz.Ör GameOfThrones instanceını kullanmak için INameChecker interfaceinden miras almamız gerekmektedir.

Elimizdeki var olan oyuncu isimlerini for döngüsünde döndürüyoruz. NameChecker listesinde bulunan instancelarda kontrol ediyoruz ve eşleşme başarılı olduğunda name, namechecker ve friendlyname ile ekrana basıyoruz. Bu sayede oyuncu değerlerini değiştirdiğimizde dizi sınıflarına müdahele etmeden aynı işlemi sürdürmeye devam edebilmekteyiz.

# UML Diyagramı





# Kodlar

#Program.cs

|  |
| --- |
| /\*Süha Aşkın Gündüz  30.05.2020 \*/  using System;  using System.Collections.Generic;  using SuperCoolLibrary;  namespace MyApp  {  class Program  {  static void Main(string[] args)  {  //NameManager olarak yarattığımız class için bir degişken tanımlıyoruz  var nameManager = new NameManager();    //nameManager.OnNotification += NameManagerOnOnNotification;  //kontrol edilecek isimlerin listesini yaratıyoruz  var possibleNames = new List<string> {"Fred", "George","Burhan", "Serpil","Jon", "Yaprak","Daphne","Katil", "Arya","Suha"};  //Kontrol listesinde bulunan bütün objeler için kontrol yapıyoruz  foreach (var name in possibleNames)  {  //GetModel  var result = nameManager.GetModel(name);  //yukarıda yaratılan liste baz alınarak sonuçları yazdırıyoruz  if (result.Success)  {  Console.WriteLine("Listedeki " + $"{result.Name} ismi {result.FriendlyName} sınıfına aittir!");  }  else  {  Console.WriteLine($"{result.Name} ismi {result.FriendlyName}!");  }  Console.WriteLine("\n");  }  Console.ReadKey();  }  //logları yazdırmak için  private static void NameManagerOnOnNotification(object sender, NotificationEventArg notificationEventArg)  {  //Dönen mesajı bastırıyoruz  Console.WriteLine(notificationEventArg.Message);  }  }  } |

#NameManager.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using SuperCoolLibrary.NameCheckers;  namespace SuperCoolLibrary  {  public class NameManager : INameManager  {  public event NotificationHandler OnNotification;  public PopCultureNameModel GetModel(string name)  {  //İsmi kontrol eden işlem  Notify($"Checking '{name}'...");  //Kontrol için kullanılacak objeyi yaratıp referans alınacak classları tek tek ekliyoruz listeye  //Liste içinde class ekleyeceğimiz için ve class tipleri ICheckerNames olduğu için liste tipi de INameChecker oluyor  var nameCheckers = new List<INameChecker>  {  new AvrupaYakasıChecker(),  new ScoobyDooNameChecker(),  new SmurfNameChecker(),  new GameOfThronesChecker(),  new BizimkilerChecker()  };  //Kontrol listesindeki bütün objeler için işlem yapıyoruz  foreach (var nameChecker in nameCheckers)  {  //Mevcut nameChecker'ı yazdırıyoruz  Notify($"Kullanılan Checker: '{nameChecker.FriendlyName}'...");  //Programda tanımladığımız liste içindeki isimler uyuşuyor mu diye kontrol sağlıyoruz  if (nameChecker.CheckName(name))  {  //Fonksiyondan true döndüyse uyuşma sağlandı şeklinde anlıyoruz ve bunu logluyoruz  Notify($"Uyuşan bir sınıf bulundu!");  Notify($"\n");  //True dönen obje için ilgili sınıf ismini, tipini ve public değer olarak belirlediğimiz adını alıyoruz  return new PopCultureNameModel  {  Success = true,  Name = name,  NameChecker = nameChecker.GetType().Name,  FriendlyName = nameChecker.FriendlyName  };  }  //fonksiyondan false döndüyse uyuşma olmadı anlamında logu basıyoruz  else  {  Notify($"Sınıflarla uyuşma olmadı...");  Notify($"\n");  }  }  //Listede bulunan isim NameChecker listelerinde bulunamazsa farklı mesaj veriyoruz  Notify($"Bu isim listelerde mevcut değildir!");  //Herhangi bir sınıfa ait olmayan veriler için bu sınıfa atama yapıp bilinmeyen bir sınıf  //olduğuna dair mesaj bastırmak için bu verilerin atamasını kullanıyoruz  return new PopCultureNameModel  {  Success = false,  Name = name,  NameChecker = "bilinmiyor",  FriendlyName = "bilinmiyor"  };  }  private void Notify(string message)  {  if (OnNotification != null)  {  OnNotification.Invoke(this, new NotificationEventArg  {  Message = message  });  }  }  }  } |

#INameManager.cs

|  |
| --- |
| using System;  namespace SuperCoolLibrary  {  public delegate void NotificationHandler(object sender, NotificationEventArg e);  public interface INameManager  {  event NotificationHandler OnNotification;  PopCultureNameModel GetModel(string name);  }  } |

#NotificationEventArg.cs

|  |
| --- |
| using System;  namespace SuperCoolLibrary  {  public class NotificationEventArg : EventArgs  {  public string Message { get; set; }  }  } |

#PopCultureNameModel.cs

|  |
| --- |
| namespace SuperCoolLibrary  {  public class PopCultureNameModel  {  public bool Success { get; set; }  public string Name { get; set; }  public string NameChecker { get; set; }  public string FriendlyName { get; set; }  }  } |

#AvrupaYakasiChecker.cs

|  |
| --- |
| using System.Collections.Generic;  using System.Linq;  //Namechecker class tanımı  namespace SuperCoolLibrary.NameCheckers  {  public class AvrupaYakasıChecker : INameChecker  {  public bool CheckName(string name)  {  //Avrupa Yakası dizisnde geçen karakterlerin isimlerini bu class altında oluşturduğumuz liste içinde  //belirtiyoruz. Programın ana işlemleri esnasında karşılaştırma/kontrol için bu class kullanılacak.  //Liste içeriğini oluşturuyoruz  var list = new List<string>  {  "Burhan", "Aslı", "Volkan", "Şahika", "Yaprak"  };  //karşılaştırma yaparken karakter sorunu yaşanmaması adına liste içindeki tüm elemanları lower case yapıyoruz.  return list.Any(x => x.ToLowerInvariant() == name.ToLowerInvariant());  }  //Her yerden erişebilmek adına public değer olarak tanımlıyoruz  public string FriendlyName => "Avrupa Yakası";  }  } |

#BizimkilerChecker.cs

|  |
| --- |
| using System.Collections.Generic;  using System.Linq;  //Namechecker class tanımı  namespace SuperCoolLibrary.NameCheckers  {  public class BizimkilerChecker : INameChecker  {  public bool CheckName(string name)  {  //Bizimlkiler dizisnde geçen karakterlerin isimlerini bu class altında oluşturduğumuz liste içinde  //belirtiyoruz. Programın ana işlemleri esnasında karşılaştırma/kontrol için bu class kullanılacak.  //Liste içeriğini oluşturuyoruz  var list = new List<string>  {  "Sabri", "Katil", "Davut", "Halil", "Serpil"  };  //karşılaştırma yaparken karakter sorunu yaşanmaması adına liste içindeki tüm elemanları lower case yapıyoruz.  return list.Any(x => x.ToLowerInvariant() == name.ToLowerInvariant());  }  //Her yerden erişebilmek adına public değer olarak tanımlıyoruz  public string FriendlyName => "Bizimkiler";  }  } |

#GameOfThronesChecker.cs

|  |
| --- |
| using System.Collections.Generic;  using System.Linq;  namespace SuperCoolLibrary.NameCheckers  {  public class GameOfThronesChecker : INameChecker  {  public bool CheckName(string name)  {  //Game Of Thrones dizisnde geçen karakterlerin isimlerini bu class altında oluşturduğumuz liste içinde  //belirtiyoruz. Programın ana işlemleri esnasında karşılaştırma/kontrol için bu class kullanılacak.  //Liste içeriğini oluşturuyoruz  var list = new List<string>  {  "Jon", "Eddard", "Stannis", "Jamie", "Arya"  };  //karşılaştırma yaparken karakter sorunu yaşanmaması adına liste içindeki tüm elemanları lower case yapıyoruz.  return list.Any(x => x.ToLowerInvariant() == name.ToLowerInvariant());  }  //Her yerden erişebilmek adına public değer olarak tanımlıyoruz  public string FriendlyName => "Game of Thrones";  }  } |

#ScoobyDooNameChecker.cs

|  |
| --- |
| using System.Collections.Generic;  using System.Linq;  //Namechecker class tanımı  namespace SuperCoolLibrary.NameCheckers  {  public class ScoobyDooNameChecker : INameChecker  {  public bool CheckName(string name)  {  //ScoobyDoo dizisnde geçen karakterlerin isimlerini bu class altında oluşturduğumuz liste içinde  //belirtiyoruz. Programın ana işlemleri esnasında karşılaştırma/kontrol için bu class kullanılacak.  //Liste içeriğini oluşturuyoruz  var list = new List<string>  {  "Fred", "Daphne", "Velma", "Shaggy", "Scooby"  };  //karşılaştırma yaparken karakter sorunu yaşanmaması adına liste içindeki tüm elemanları lower case yapıyoruz.  return list.Any(x => x.ToLowerInvariant() == name.ToLowerInvariant());  }  //Her yerden erişebilmek adına public değer olarak tanımlıyoruz  public string FriendlyName => "Scooby Doo";  }  } |

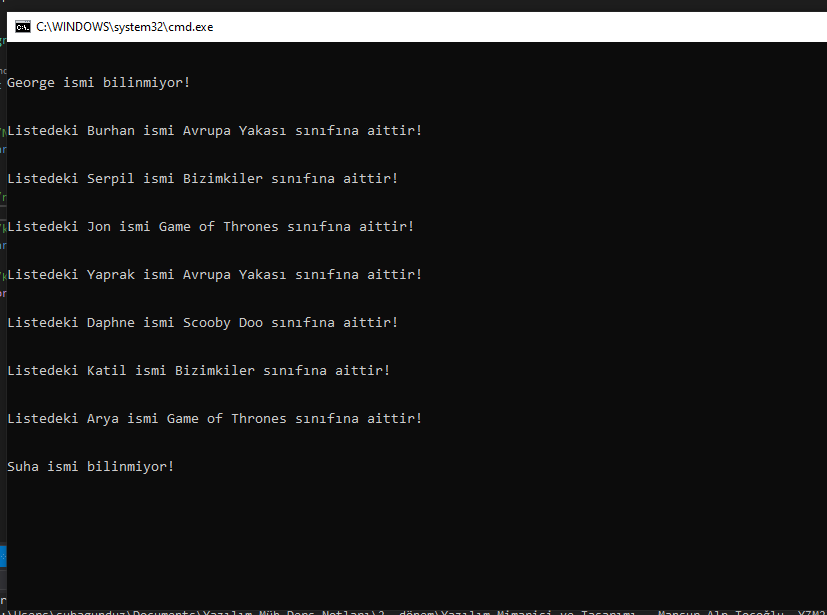
#SmurfNameChecker.cs

|  |
| --- |
| using System.Collections.Generic;  using System.Linq;  namespace SuperCoolLibrary.NameCheckers  {  public class SmurfNameChecker : INameChecker  {  public bool CheckName(string name)  {  var list = new List<string>  {  "Papa", "Brainy", "Hefty", "Smurfette"  };  return list.Any(x => x.ToLowerInvariant() == name.ToLowerInvariant());  }  public string FriendlyName => "The Smurfs";  }  } |

#INameChecker.cs

|  |
| --- |
| namespace SuperCoolLibrary.NameCheckers  {  //class tiplerinde kullandığımız interface içeriğini tanımlıyoruz  public interface INameChecker  {  //Bu obje kontroller esnasında true ya da false dönecek  //Yani eğer ana programdaki listede girilen isim ilgili class içindeki listede var ise true dönecek  bool CheckName(string name);  //Bu obje kontroller esnasında true dönen ismin class adını yazdıracak  //Class tanımlarının sonunda belirtilen public classname objesini döndürecek  string FriendlyName { get; }  }  } |

# Ekran Görüntüsü:



# GitHub Linki: <https://github.com/suhagunduz/ObserverPattern>