YZM-2118 YAZILIM MİMARİSİ VE TASARIMI VİZE PROJE ÖDEVİ

Manisa Celal Bayar Üniversitesi / Yazılım Mühendisliği

182802006 Ahmet Sait ÇALIŞIR

192803079 Süha Aşkın GÜNDÜZ

182803001 Abdullah Obaid

2020

14.05.2020

Iki Tarih Arasi

Ahmet Sait Çalışır (182802006)

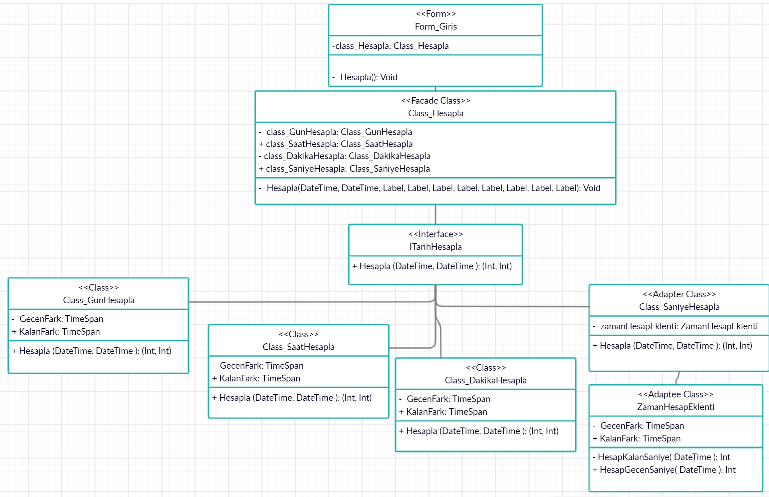
# Projenin Amacı:

Adapter pattern ve Facade patten’i bir proje içerisinde kullanmak.

# Proje Kapsamı:

Belirtilen başlangıç zamanından şimdiye kadar ne kadar zaman geçtiğini ve belirtilen son tarihe ne kadar zaman kaldığını gösteren program.

# UML Diyagramı



# Kodlar

## Form\_Giris

public Form\_Giris()

{

InitializeComponent();

}

Class\_Hesapla class\_Hesapla = new Class\_Hesapla();

private void Form\_Giris\_Load(object sender, EventArgs e)

{

dateTime\_BaslangicGun.Value = DateTime.Now.AddDays(-1);

dateTime\_SonGun.Value = DateTime.Now.AddDays(1);

dateTime\_BaslangicSaat.Value = DateTime.Now;

dateTime\_SonSaat.Value = DateTime.Now;

Hesapla();

}

private void Hesapla()

{

class\_Hesapla.Hesapla(dateTime\_BaslangicSaat.Value, dateTime\_SonSaat.Value, label\_KalanGun, label\_GecenGun, label\_KalanSaat, label\_GecenSaat, label\_KalanDakika, label\_GecenDakika,label\_KalanSaniye,label\_GecenSaniye);

}

private void dateTime\_BaslangicGun\_ValueChanged(object sender, EventArgs e)

{

if (dateTime\_BaslangicGun.Value > DateTime.Now)

{

dateTime\_BaslangicGun.Value = DateTime.Now;

}

else

{

dateTime\_BaslangicSaat.Value = dateTime\_BaslangicGun.Value;

}

}

private void dateTime\_SonGun\_ValueChanged(object sender, EventArgs e)

{

if (dateTime\_SonGun.Value < DateTime.Now)

{

dateTime\_SonGun.Value = DateTime.Now;

}

else

{

dateTime\_SonSaat.Value = dateTime\_SonGun.Value;

}

}

private void timer\_Tick(object sender, EventArgs e)

{

Hesapla();

if (dateTime\_SonGun.Value < DateTime.Now)

{

dateTime\_SonGun.Value = DateTime.Now;

}

if (dateTime\_BaslangicGun.Value > DateTime.Now)

{

dateTime\_BaslangicGun.Value = DateTime.Now;

}

}

private void dateTime\_BaslangicSaat\_ValueChanged(object sender, EventArgs e)

{

dateTime\_BaslangicGun.Value = dateTime\_BaslangicSaat.Value;

}

private void dateTime\_SonSaat\_ValueChanged(object sender, EventArgs e)

{

dateTime\_SonGun.Value = dateTime\_SonSaat.Value;

}

private void button\_Cikis\_Click(object sender, EventArgs e)

{

Application.Exit();

}

## Class\_Hesapla

private Class\_GunHesapla class\_GunHesapla = new Class\_GunHesapla();

private Class\_SaatHesapla class\_SaatHesapla = new Class\_SaatHesapla();

private Class\_DakikaHesapla class\_DakikaHesapla = new Class\_DakikaHesapla();

private Class\_SaniyeHesapla class\_SaniyeHesapla = new Class\_SaniyeHesapla();

public void Hesapla(DateTime Baslangic, DateTime Son, Label KalanGun, Label GecenGun, Label KalanSaat, Label GecenSaat, Label KalanDakika, Label GecenDakika, Label KalanSaniye, Label GecenSaniye)

{

KalanGun.Text = class\_GunHesapla.Hesapla(Baslangic, Son).Item1.ToString();

GecenGun.Text = class\_GunHesapla.Hesapla(Baslangic, Son).Item2.ToString();

KalanSaat.Text = class\_SaatHesapla.Hesapla(Baslangic, Son).Item1.ToString();

GecenSaat.Text = class\_SaatHesapla.Hesapla(Baslangic, Son).Item2.ToString();

KalanDakika.Text = class\_DakikaHesapla.Hesapla(Baslangic, Son).Item1.ToString();

GecenDakika.Text = class\_DakikaHesapla.Hesapla(Baslangic, Son).Item2.ToString();

KalanSaniye.Text = class\_SaniyeHesapla.Hesapla(Baslangic, Son).Item1.ToString();

GecenSaniye.Text = class\_SaniyeHesapla.Hesapla(Baslangic, Son).Item2.ToString();

}

## ITarihHesapla

(int, int) Hesapla(DateTime Baslangic, DateTime Son);

## Class\_GunHesapla

private TimeSpan GecenFark;

private TimeSpan KalanFark;

public (int, int) Hesapla(DateTime Baslangic, DateTime Son)

{

GecenFark = DateTime.Now - Baslangic;

KalanFark = Son - DateTime.Now;

return (Convert.ToInt32(KalanFark.TotalDays), Convert.ToInt32(GecenFark.TotalDays));

}

## Class\_SaatHesapla

private TimeSpan GecenFark;

private TimeSpan KalanFark;

public (int, int) Hesapla(DateTime Baslangic, DateTime Son)

{

GecenFark = DateTime.Now - Baslangic;

KalanFark = Son - DateTime.Now;

return (Convert.ToInt32(KalanFark.TotalHours), Convert.ToInt32(GecenFark.TotalHours));

}

## Class\_DakikaHesapla

private TimeSpan GecenFark;

private TimeSpan KalanFark;

public (int, int) Hesapla(DateTime Baslangic, DateTime Son)

{

GecenFark = DateTime.Now - Baslangic;

KalanFark = Son - DateTime.Now;

return (Convert.ToInt32(KalanFark.TotalMinutes), Convert.ToInt32(GecenFark.TotalMinutes));

}

## Class\_SaniyeHesapla

private ZamanHesapEklenti zamanHesapEklenti = new ZamanHesapEklenti();

public (int, int) Hesapla(DateTime Baslangic, DateTime Son)

{

return (zamanHesapEklenti.HesapKalanSaniye(Son), zamanHesapEklenti.HesapGecenSaniye(Baslangic));

}

## ZamanHesapEklenti

private TimeSpan FarkKalan;

private TimeSpan FarkGecen;

public int HesapKalanSaniye(DateTime Son)

{

FarkKalan = Son - DateTime.Now;

return Convert.ToInt32(FarkKalan.TotalSeconds);

}

public int HesapGecenSaniye(DateTime Baslangic)

{

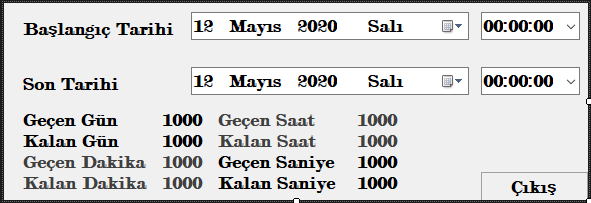
FarkGecen = DateTime.Now-Baslangic;

return Convert.ToInt32(FarkGecen.TotalSeconds);

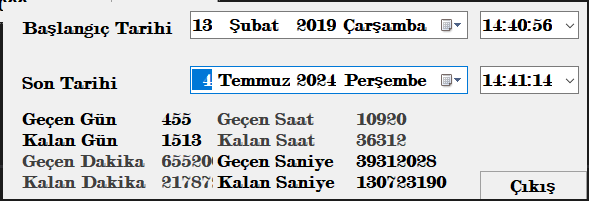
}

# Ekran Görüntüsü

## Tasarım ekran görüntüsü



## Uygulama ekran görüntüsü



# GitHub Linki:

<https://github.com/AhmetSaitCalisir/Iki-Tarih-Arasi>

Okul sınıfın yöneticisi(Birinci Proje)

Abdullah Obaid (182803001)

# Projenin Amacı:

Abstract Factory patterni kullanarak bir sınıf yönetici projesi.

# Proje Kapsamı:

Sınıfın adı, sınıfın No , sınıfın öğretmeni ve sınıfın öğrencileri eklemek ve o detayları ekrana yazdırılacak.

# UML Diyagramı

# 

# Kodlar

# Ogrenciler.java

package AbstractFactroryPackage;  
public interface Ogrenciler {  
 String getDescription();  
}

# Ogretmen.java

package AbstractFactroryPackage;  
  
public interface Ogretmen {  
 String getDescription();  
}

# SinifNo.java

package AbstractFactroryPackage;  
  
public interface SinifNo {  
   
 int getSinifNo();  
  
}

# SinifFactory.java

package AbstractFactroryPackage;  
  
public interface SinifFactory {  
  
 Ogrenciler createOgrenciler();  
  
 Ogretmen createOgretmen();  
  
 SinifNo createSinifNo();  
}

# MatematikSinif.java

package AbstractFactroryPackage;  
  
public class MatematikSinif implements SinifFactory {  
  
 @Override  
 public Ogrenciler createOgrenciler() {  
 return new MatOgrenciler();  
 }  
  
 @Override  
 public Ogretmen createOgretmen() {  
 return new MatOgretmen();  
 }  
  
 @Override  
 public SinifNo createSinifNo() {  
 return new MatSinifNo();  
 }  
}

# FizikSinif.java

package AbstractFactroryPackage;  
  
public class FizikSinif implements SinifFactory {  
  
 @Override  
 public Ogretmen createOgretmen() {  
 return new FizikOgretmen();  
 }  
  
 @Override  
 public Ogrenciler createOgrenciler() {  
 return new FizikOgrenciler();  
 }  
  
 @Override  
 public SinifNo createSinifNo() {  
 return new FizikSinifNo();  
  
 }  
}

# FizikOgretmen.java

package AbstractFactroryPackage;  
  
public class FizikOgretmen implements Ogretmen {  
 private static final String *fizikOgretment* = "Fizik Ogretmen !";  
  
 @Override  
 public String getDescription() {  
 return *fizikOgretment*;  
 }  
}

# FizikOgrenciler.java

package AbstractFactroryPackage;  
  
public class FizikOgrenciler implements Ogrenciler {  
 private static String *fizikOgrenciler* = "Fizik Ogrenciler !";  
  
 @Override  
 public String getDescription() {  
 return *fizikOgrenciler*;  
 }  
}

# FizikSinifNo.java

package AbstractFactroryPackage;  
  
public class FizikSinifNo implements SinifNo {  
 private static int *fizikSinifNo* = 109;  
  
 @Override  
 public int getSinifNo() {  
 return *fizikSinifNo*;  
 }  
}

# MatOgretmen.java

package AbstractFactroryPackage;  
  
public class MatOgretmen implements Ogretmen {  
 private static final String *MatOgretmen* = "Matematık Ogretmeni";  
  
 @Override  
 public String getDescription() {  
 return *MatOgretmen*;  
 }  
  
}

# MatOgrenciler.java

package AbstractFactroryPackage;  
  
public class MatOgrenciler implements Ogrenciler {  
 static final String *matOgrenciler* = "Matematik Ogrencileri... ";  
  
 @Override  
 public String getDescription() {  
 return *matOgrenciler*;  
 }  
}

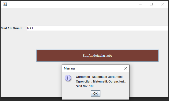
# MatSinifNo.java

package AbstractFactroryPackage;  
  
public class MatSinifNo implements SinifNo {  
  
 private static int *SinifNo* = 105;  
  
 public int getSinifNo() {  
 return *SinifNo*;  
 }  
}

# GUI.java

package AbstractFactroryPackage;  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
public class GUI {  
 JPanel panel1;  
 private JButton Button1;  
 private JTextField textField1;  
 private Ogrenciler ogrenciler;  
 private Ogretmen ogretmen;  
 private SinifNo sinifNo;  
 public void createSinif(final SinifFactory factory) {  
 setOgrenciler(factory.createOgrenciler());  
 setOgretmen(factory.createOgretmen());  
 setSinifNo(factory.createSinifNo());  
 }  
 private void setOgretmen(final Ogretmen ogretmen) {  
 this.ogretmen = ogretmen;  
 }  
 private void setOgrenciler(final Ogrenciler ogrenciler) {  
 this.ogrenciler = ogrenciler;  
 }  
 private void setSinifNo(final SinifNo sinifNo) {  
 this.sinifNo = sinifNo;}  
 public static class FactoryMaker{  
 enum SinifAdi {  
 *MAT* , *FIZ* }  
 public static SinifFactory makeFactory(GUI.FactoryMaker.SinifAdi Adi){  
 switch (Adi){  
 case *MAT*:  
 return new MatematikSinif();  
 case *FIZ*:  
 return new FizikSinif();  
 default:  
 throw new IllegalArgumentException("");}}}  
 public GUI() {  
 Button1.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {var run1 = new GUI();String button = textField1.getText();  
 if(button.equalsIgnoreCase(String.*valueOf*(GUI.FactoryMaker.SinifAdi.*MAT*)) ){  
 run1.createSinif(GUI.FactoryMaker.*makeFactory*(GUI.FactoryMaker.SinifAdi.*MAT*));  
 JOptionPane.*showMessageDialog*(null, "Ogretmen : "+ run1.ogretmen.getDescription() + "\nOgrenciler : " + run1.ogrenciler.getDescription() + "\nSinif No : " + run1.sinifNo.getSinifNo()); }  
 else if (button.equalsIgnoreCase(String.*valueOf*(GUI.FactoryMaker.SinifAdi.*FIZ*))) {  
 run1.createSinif(GUI.FactoryMaker.*makeFactory*(GUI.FactoryMaker.SinifAdi.*FIZ*));  
 JOptionPane.*showMessageDialog*(null, "Ogretmen : "+ run1.ogretmen.getDescription() + "\nOgrenciler : " + run1.ogrenciler.getDescription() + "\nSinif No : " + run1.sinifNo.getSinifNo());}  
 else {JOptionPane.*showMessageDialog*(null, "SINIF BULUNMAD!.. Lutfen Tekrar Deneyin.."); } }}); }}

# Ekran Görüntüsü



Hastane Odaların Yöneticisi (İkinci Proje)

# Projenin Amacı:

Proxy patterni kullanarak bir Hastane Odalari yönetici projesi.

# Proje Kapsamı:

Hastane odanın max kapasitesi yazarak ve girecek olan hasta saysı.

# UML Diyagramı

# 

# Kodlar

# Hasta.java

package ProxyDesignPattern;  
public class Hasta {  
 private final String hastaAdi;  
 public Hasta(String hastaAdi){  
 this.hastaAdi= hastaAdi;}  
 @Override  
 public String toString(){ return hastaAdi;}}

# HastaneOdasi.java

package ProxyDesignPattern;  
public interface HastaneOdasi { void giris(Hasta hasta);}

# Oda109.java

package ProxyDesignPattern;  
import javax.swing.\*;  
public class Oda109 implements HastaneOdasi {  
 public void giris(Hasta hasta){  
 JOptionPane.*showMessageDialog*(null,hasta+"Odaya Girdi");}}

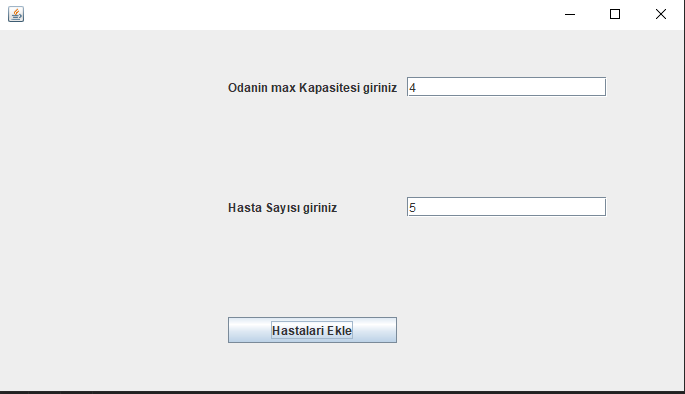
# HastaneOdasiProxy.java

package ProxyDesignPattern;  
import javax.swing.\*;  
public class HastaneOdasiProxy implements HastaneOdasi{  
 static int *maxOdaKapasitesi*;  
 static int *HastaSayisi*;  
 private static HastaneOdasi *oda*;  
 public HastaneOdasiProxy (HastaneOdasi oda ){  
 this.*oda*= oda;  
 }  
 public void giris (Hasta hasta){  
 if(*HastaSayisi* < *maxOdaKapasitesi*){  
 *oda*.giris(hasta);  
 *HastaSayisi*++; }  
 else{  
 JOptionPane.*showMessageDialog*(null,hasta +"Odaya Girilmez, Lütfen Başka Bir Odaya Götürün ");}}}

# GUI.java

package ProxyDesignPattern;  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
public class GUI {  
 HastaneOdasiProxy oda109 = new HastaneOdasiProxy(new Oda109());  
 private JTextField textField1;  
 private JTextField textField2;  
 JPanel panel1;  
 private JButton HastalariEkle;  
 public GUI() {  
 HastalariEkle.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 try{  
 int HastaSayisi = Integer.*parseInt*((textField1.getText()));  
 int MaxOdaKapasitesi = Integer.*parseInt*(textField2.getText());  
 HastaneOdasiProxy.*maxOdaKapasitesi* = MaxOdaKapasitesi;  
 for(int i =1 ; i<=HastaSayisi; i++){  
 oda109.giris(new Hasta((i) + ".Hasta ")); } }  
 catch (NumberFormatException e1){  
 JOptionPane.*showMessageDialog*(null,"lütfen Geçerli Bir Sayi Giriniz");  
 }}});}}

# Ekran Görüntüsü



# GitHub Linki: <https://github.com/AbdullahOB/Proxy-AbstractFactory>

Hesaplama ve Tarih Uygulaması (Singleton - 1. Proje)

Süha Aşkın Gündüz (192803079)

# Projenin Amacı:

Singleton pattern’i bir proje içerisinde kullanmak ve mantığını anlamak.

# Proje Detayı ve Kapsamı:

Yazılan uygulamada tanımlanan her nesne bellekte yer kaplamaktadır. Her istemci için ayrı ayrı bu nesneyi oluşturmak performans kaybına yol açacaktır. Singleton deseni üzerinden oluşturulan nesneler tek bir örnek üzerinde kalır ve her istemci bu örneği kullanır. Bir instance a kontrollü erişimi sağlar ve global değişkenler yaratmaktan kaçınmış oluruz.

Singleton deseni yapısında ki amaç, nesne global erişim noktası olarak oluşturulmaktadır. Bundan dolayı sistem çalıştığı sürece ikinci bir örnek oluşturulamaz. Bu sayede istenen nesnenin tek bir defa oluşturulmasını garanti altına almış oluruz.

Uygulamamızda ise integer ve string değerlere işlem yaptırmak dışında, current date bilgisni 3 saniye arayla çekerek aynı verinin bize return edildiğini belirlemiş, tekrardan new componenti kullarak ek tanımlama işlemi yapmaktan kurtulmuş olduk.

# UML Diyagramı

# Singleton Pattern UML

# Kodlar

#Program.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading;  using System.Threading.Tasks;  namespace \_01\_SingletonPattern  {  class Program  {  //.NET KONF  static void Main(string[] args)  {  int toplam1 = SingletonClass.Sinif.toplam(3, 2);  int carpim = SingletonClass.Sinif.carp(3, 2);  Console.WriteLine("Toplam Sonucu: "+toplam1);  Console.WriteLine("Carpim Sonucu: "+carpim);  Console.WriteLine("Toplam + Carpim Sonucu = " + (toplam1 + carpim));  Console.WriteLine(SingletonClass.Sinif.yaz());    Console.WriteLine(SingletonClass.Sinif.GetCurrentDate());  Thread.Sleep(3000);  Console.WriteLine(SingletonClass.Sinif.GetCurrentDate());  Console.ReadKey();  }  }  } |

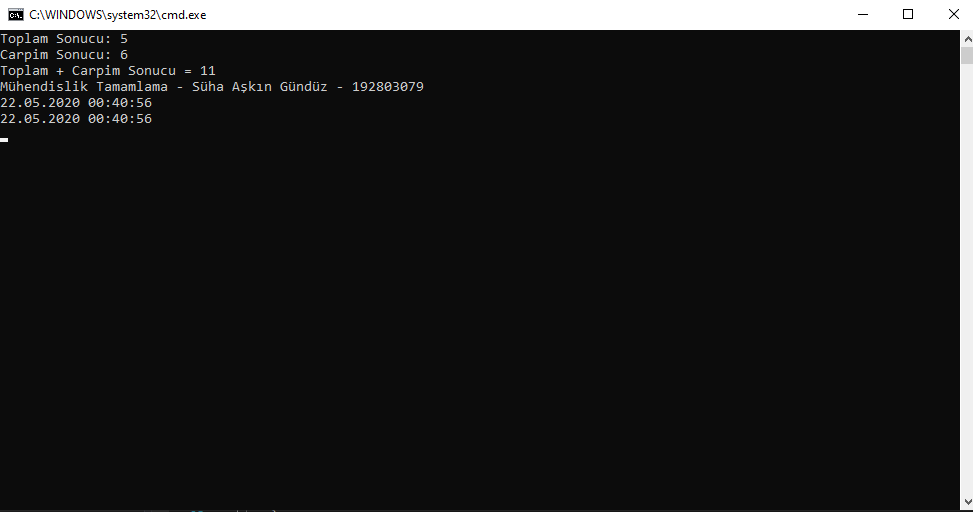
#SingletonClass.cs

|  |
| --- |
| namespace \_01\_SingletonPattern  {  class SingletonClass  {  //1) private static ve geriye class tipinde dönen bir field tanımlanır.  private static SingletonClass \_tekNesne;  //2) constructor private veya protected yapılarak erişim newleme yapılması engellenir  private SingletonClass()  {  }  //3) newleme işleminin yapıldığı geriye class tipinde dönen bir metot / property oluşturulur. Bu yapı içerisinde nesne önceden üretilmiş mi kontrol edilir üretilmediyse üretililir üretildiyse aynı nesne geri gönderilir.  public static SingletonClass Sinif  {  get  {  if(\_tekNesne == null)  {  \_tekNesne = new SingletonClass();  }  return \_tekNesne;  }  }  private DateTime now = DateTime.Now;  public DateTime GetCurrentDate() => now;  public int toplam(int s1, int s2)  {  return s1 + s2;  }  public int carp (int s1, int s2)  {  return s1 \* s2;  }  public string yaz()  {  return ("Mühendislik Tamamlama - Süha Aşkın Gündüz - 192803079");  }  }  } |

#MultiThreadSingletonClass.cs

|  |
| --- |
| namespace \_01\_SingletonPattern  {  class MultiThreadSingletonClass  {  private static MultiThreadSingletonClass \_tekNesne;  private static Object lockObject = new Object();  private MultiThreadSingletonClass()  {  }  public static MultiThreadSingletonClass Sinif  {  get  {  if (\_tekNesne == null)  {  lock(lockObject)  {  if (\_tekNesne == null)  {  \_tekNesne = new MultiThreadSingletonClass();  }  }  }  return \_tekNesne;  }  }  }  } |

# Ekran Görüntüsü



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

Araç Üretim Otomasyonu (Abstract Factory- 2. Proje)

Süha Aşkın Gündüz (192803079)

# Projenin Amacı:

Abstract Factory pattern’i bir proje içerisinde kullanmak ve mantığını anlamak.

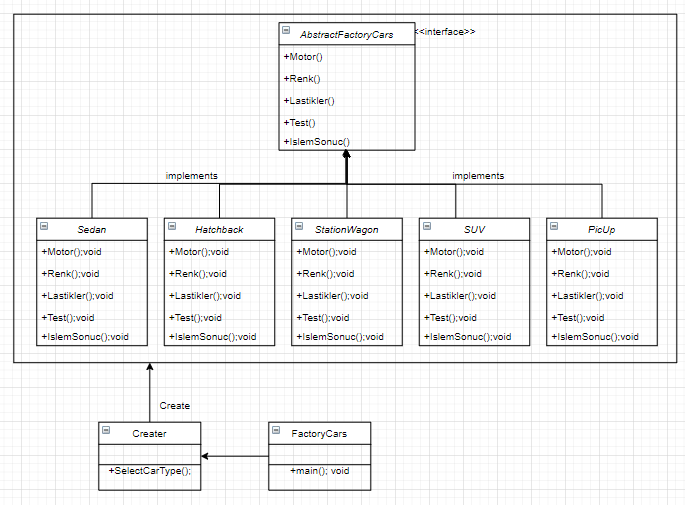
# Proje Detayı ve Kapsamı:

Abstract Factory , ürün ailesinin her sınıfından nesne oluşturmak için arayüz sağlar. Birden fazla ürün ailesi ile çalışmak durumunda kaldığımızda , ürün ailesi ile istemci tarafını soyutlamak için kullanılır. Böylece nesneleri üreten sınıf, hangi nesnenin üretileceği bilmeden bu nesnelerin oluşturulmasını yönetebiliriz.

Bu pattern yardımıyla;  
- Fabrikadan aldığımız ürünlerin birbirleriyle uyumlu olduğundan emin olabiliriz.  
-Open/Closed Principle : Mevcut kodunu bozmadan yeni ürün versiyonları üretebiliriz.   
-Single Responsibility Principle : Ürün oluşturma kodunu tek bir yerde yaparak kodun daha kolay beslenmesini sağlayabiliriz .  
-Yeni ekleme yapılması gerektiğinde kod tekrarı olmadan ugulamaya ekleme yapılabilir.

Araç üretim fabrikasında farklı kasalarda üretilen araç otomasyonu uygulamasının tanıtımı yapılmıştır. Bu uygulada implemente edilen soyut değerler her bir class altına tekrar yazma gereği duymadan abstractan bu verileri çekebilmekteyiz. Yeni bir kasa eklenmesi çıkarılması durumlarında soyut özellikleride tekrar etmeye gerek kalmadan kullanılabilmektedir.

# UML Diyagramı



# Kodlar

#Program.cs

|  |
| --- |
| namespace FactoryPattern  {  static class Program  {  /// <summary>  /// Uygulamanın ana girdi noktası.  /// </summary>  [STAThread]  static void Main()  {  Application.EnableVisualStyles();  Application.SetCompatibleTextRenderingDefault(false);  Application.Run(new Form1());  }  }  } |

#AbstractFactoryCars.cs

|  |
| --- |
| namespace FactoryPattern  {  public abstract class AbstractFactoryCars  {  public abstract void Motor();  public abstract void Renk();  public abstract void Lastikler();  public abstract void Test();  public abstract string IslemSonuc();  }  } |

#AllCars.cs

|  |
| --- |
| namespace FactoryPattern  {  enum AllCars  {  Sedan,  Hatchback,  StationWagon,  PicUp,  Suv  }  } |

#Creater.cs

|  |
| --- |
| namespace FactoryPattern  {  class Creater  {  public AbstractFactoryCars FactoryMethod(AllCars CarType)  {  AbstractFactoryCars abstractFactoryCars = null;  switch (CarType)  {  case AllCars.Sedan:  abstractFactoryCars = new Sedan();  break;  case AllCars.Hatchback:  abstractFactoryCars = new Hatchback();  break;  case AllCars.StationWagon:  abstractFactoryCars = new StationWagon();  break;  case AllCars.Suv:  abstractFactoryCars = new Suv();  break;  case AllCars.PicUp:  abstractFactoryCars = new PicUp();  break;  }  return abstractFactoryCars;  }  }  } |

#Hatchback.cs

|  |
| --- |
| namespace FactoryPattern  {  class Hatchback:AbstractFactoryCars  {  public override void Lastikler()  {  MessageBox.Show("Lastikler takıdı.");  }  public override void Motor()  {  MessageBox.Show("Motor takıdı.");  }  public override void Renk()  {  MessageBox.Show("Araç boyandı");  }  public override void Test()  {  MessageBox.Show("Araç test edildi.");  }  public override string IslemSonuc()  {  return "Hatchback Araç Tamamlandı, satışa hazır.";  }  }  } |

#PicUp.cs

|  |
| --- |
| namespace FactoryPattern  {  namespace FactoryPattern  {  class PicUp: AbstractFactoryCars  {  public override void Lastikler()  {  MessageBox.Show("Lastikler takıdı.");  }  public override void Motor()  {  MessageBox.Show("Motor takıdı.");  }  public override void Renk()  {  MessageBox.Show("Araç boyandı");  }  public override void Test()  {  MessageBox.Show("Araç test edildi.");  }  public override string IslemSonuc()  {  return "PicUp Araç Tamamlandı, satışa hazır.";  }  }  } |

#StationWagon.cs

|  |
| --- |
| namespace FactoryPattern  {  class StationWagon : AbstractFactoryCars  {  public override void Lastikler()  {  MessageBox.Show("Lastikler takıdı.");  }  public override void Motor()  {  MessageBox.Show("Motor takıdı.");  }  public override void Renk()  {  MessageBox.Show("Araç boyandı");  }  public override void Test()  {  MessageBox.Show("Araç test edildi.");  }  public override string IslemSonuc()  {  return "Station Wagon Araç Tamamlandı, satışa hazır.";  }  }  } |

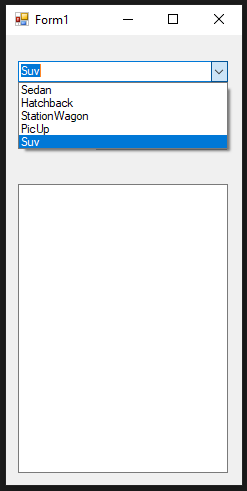
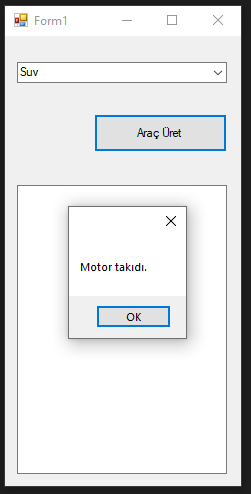
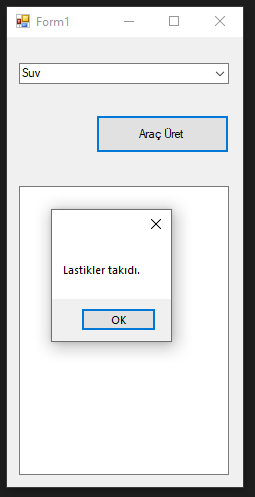
#StationWagon.cs

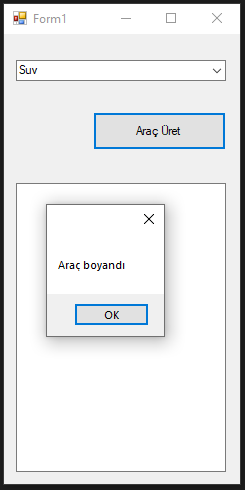
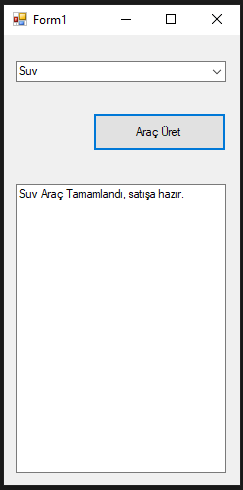
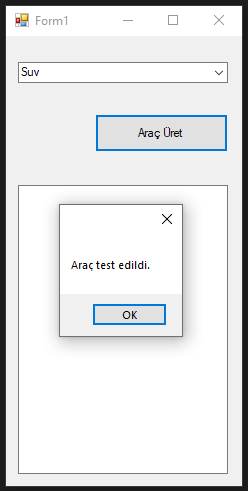
|  |
| --- |
| namespace FactoryPattern  {  class Sedan : AbstractFactoryCars  {  public override void Lastikler()  {  MessageBox.Show("Lastikler takıdı.");  }  public override void Motor()  {  MessageBox.Show("Motor takılmadı");  }  public override void Renk()  {  MessageBox.Show("Araç boyandı");  }  public override void Test()  {  MessageBox.Show("Araç test edildi.");  }  public override string IslemSonuc()  {  return "Sedan Araç Tamamlandı, satışa hazır.";  }  }  } |

#StationWagon.cs

|  |
| --- |
| namespace FactoryPattern  {  class Suv : AbstractFactoryCars  {  public override void Lastikler()  {  MessageBox.Show("Lastikler takıdı.");  }  public override void Motor()  {  MessageBox.Show("Motor takıdı.");  }  public override void Renk()  {  MessageBox.Show("Araç boyandı");  }  public override void Test()  {  MessageBox.Show("Araç test edildi.");  }  public override string IslemSonuc()  {  return "Suv Araç Tamamlandı, satışa hazır.";  }  }  } |

# Ekran Görüntüsü:

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