**BAHAR DÖNEMİ VERİ YAPILARI PROJE RAPORU**

Ogrenci No: 032190075

Ogrenci İsmi: Egemen Keskin

Ders kodu: BMB2006

PROJE: Bir Kıyafet Mağazasının Personel, Ürün ve Satış Kayıtlarını Tutan Bir Yazılım

Hazırladığım uygulama 4 formdan oluşmaktadır, aşağıda hepsinin ekran görüntüleri ve form bileşenleri hakkında bilgiler bulunmaktadır.

1.Form: Giriş Paneli

A screenshot of a login box

Description automatically generated with medium confidence

Bu formda giriş paneli yer almaktadır. Sisteme girmek isteyen kullanıcıların bu panele E-Mail ve Şifre bilgilerini girmesi gerekmektedir.



-Bu textbox sisteme girmek isteyen kullanıcıdan E-Mail bilgisini ister ve girilen bilgiyi tutar.



-Bu textbox sisteme girmek isteyen kullanıcıdan Şifre bilgisini ister ve girilen bilgiyi tutar.



-Bu buton üzerinde de yazıldığı üzere sisteme giriş yapılmasını sağlar, E-Mail ve Şifre textboxlarından alınan girdiler kullanıcı veritabanındaki verilerle eşleşirse sisteme giriş yapılır, eşleşmezse hata verir.

2.Form :Seçim Paneli

A screenshot of a computer

Description automatically generated with medium confidence

Bu formda, sisteme giriş yapmış olan kullanıcının ürün kayıtlarıyla mı yoksa personel kayıtlarıyla mı işlem yapacağını seçmesi gerekmektedir.

A picture containing text, font, screenshot, rectangle

Description automatically generated

-Bu buton kullanıcıyı ürün kayıtlarının ve ürün kayıt işlemlerinin bulunduğu panele yönlendirir.

A picture containing text, font, screenshot, rectangle

Description automatically generated

-Bu buton kullanıcıyı personel kayıtlarının ve personel kayıt işlemlerinin bulunduğu panele yönlendirir.

3.Form :Ürün Kayıt İşlemleri Paneli

A screenshot of a computer

Description automatically generated with medium confidence

Bu formda, kullanıcı girmiş olduğu AD, Fiyat ve ID bilgileriyle kıyafeti sisteme ekleyip görüntüleyebilir.

Sistemden silinmek istenen bir kıyafet varsa silinebilir.



-Bu buton ile sistemde mevcut olan bütün kıyafetler ve bilgileri görüntülenebilir.



-Bu buton ile ID’si girilen kıyafetin sistemden silinmesi sağlanır.

4.Form: Personel Kayıt İşlemleri Paneli

A screenshot of a computer

Description automatically generated

Bu formda, kullanıcı sisteme yeni personel ekleyebilir ve varolan bir personel kaydını silebilir.



-Bu buton ile sistemde mevcut olan bütün personellerin bilgileri görüntülenebilir.



-Bu buton ile sistemde mevcut olan bir personelin kaydının silinmesi sağlanabilir.



-Bu buton ile sistemde var olmayan bir personelin kaydı sisteme eklenebilir.

Veri Tabanında personel kaydı, ürün kaydı ve kullanıcı kaydı olmak üzere 3 adet tablo kullandım.

Ürün Kayıt Tablosu:

A picture containing text, screenshot, menu, number

Description automatically generated

Personel Kayıt Tablosu:

A picture containing text, screenshot, font, number

Description automatically generated

Kullanıcı Tablosu:



Şuanda sadece tek bir kullanıcı kayıtlı olduğu için 1 kullanıcı gözükmektedir.

Kullanıcı Tablosu SQL - Create Kod:

USE [projeDB]

GO

/\*\*\*\*\*\* Object: Table [dbo].[yonetici\_kayit] \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[yonetici\_kayit](

[Ad] [nvarchar](20) NOT NULL,

[Soyad] [nvarchar](20) NOT NULL,

[Mail] [nvarchar](30) NOT NULL,

[Parola] [nvarchar](30) NOT NULL,

PRIMARY KEY CLUSTERED

(

[Mail] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Ürün Kayıt Tablosu SQL Create Kod:

USE [projeDB]

GO

/\*\*\*\*\*\* Object: Table [dbo].[urun\_kayit] \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[urun\_kayit](

[UrunID] [int] NOT NULL,

[UrunAD] [nvarchar](50) NOT NULL,

[Fiyat] [money] NOT NULL,

CONSTRAINT [PK\_urun\_kayit] PRIMARY KEY CLUSTERED

(

[UrunID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Kullanıcı Tablosu SQL – Create Kod:

USE [projeDB]

GO

/\*\*\*\*\*\* Object: Table [dbo].[yonetici\_kayit] \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[yonetici\_kayit](

[Ad] [nvarchar](20) NOT NULL,

[Soyad] [nvarchar](20) NOT NULL,

[Mail] [nvarchar](30) NOT NULL,

[Parola] [nvarchar](30) NOT NULL,

PRIMARY KEY CLUSTERED

(

[Mail] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SQL SORGULARI VE AÇIKLAMALARI



Bu kod bloğunda, SqlConnection sınıfından baglanti isimli bir nesne oluşturuyoruz ve SSMS’te oluşturduğumuz veri tabanının linkini bu nesneye atıyoruz. Atama yaptıktan sonra bağlantının aktif hale gelmesi için Open() fonksiyonunu kullanıyoruz.



Bu kod bloğunda, iki adet string değişkene iki adet farklı sorgu atıyoruz. İlk sorguda yonetici\_kayit adlı tablodan, kullanıcının girdiği mail ile eşleşen parolayı çekmesini sağlıyoruz.

İkinci sorguda ise yonetici\_kayit adli tablodan bütün mailleri çekmesini sağlıyoruz.



Burada SqlCommand sınıfından iki adet nesne üretiyoruz. İlk parametreye sorgumuzu, ikinci parametreye bağlantımızı giriyoruz.



Burada, sorguda @Mail yazan kısma kullanıcının girdiği değerin yazılması sağlanıyor.



Bu kodda ikinci sorgumuzu ExecuteReader() fonksiyonu ile çalıştırıyoruz.



Bu kod bloğunda, personel\_kayit adlı tablodan bütün bilgileri çekiyoruz.

A screen shot of a computer program

Description automatically generated with low confidence

Burada, sqlDR nesnesi değer okuması yaptığı taktirde devam edecek bir while döngüsü açıyoruz.

İçerisinde personellerin bilgilerine denk gelen 5 adet string değişken tanımlıyoruz ve sırasıyla atama yapıyoruz.

A picture containing text, screenshot

Description automatically generated

Burada sql isimli değişkenimize bir sorgu ataması yapıyoruz ve SqlCommand sınıfından oluşturduğumuz komut adlı nesneyle, baglantiyi da sağlayarak bir komut oluşturmuş oluyoruz.

Komut.parameters.AddWithValue() fonksiyonlarıyla da sorguda yazdığımız değişkenler yerine, kullanıcının textboxa girdiği değerlerin yazılmasını sağlıyoruz.

Ve en sonda ise ExecuteNonQuery() fonksiyonuyla komutumuzu çalıştırıyoruz.



Burada textboxtan girilen ID ile eşleşen personelin tüm verilerini çekecek olan sorguyu secmeSorgusu değişkenine atıyoruz ve SqlCommand ile bir komut haline getiriyoruz.

A picture containing text, screenshot, font

Description automatically generated

Burada ise eşleşen personelin verilerinin silinmesini, çıkan bir onay penceresiyle onaylatıyoruz ve silinmesini sağlayacak sorguyu oluşturuyoruz ve en sonda çalıştırıp kaydın silindiğine dair bir mesaj gönderiyoruz.



Bu kod bloğunda, kullanıcının RID adlı textboxa girdiği ürün ID’si ile eşleşen ürününbütün verilerini urun\_kayit adlı tablodan çekecek olan sorguyu yazıyoruz.

A picture containing text, screenshot, font

Description automatically generated

Burada, kullanıcının silme işlemini onaylaması durumunda gerçekleşecek olan sorguyu yazıyoruz. Burada, kullanıcının textboxa girdiği ID ile eşleşen ürünün silinmesini sağlayacak sorguyu yazıyoruz ve ExecuteNonQuery() fonksiyonu ile çalıştırıyoruz. En sonda ise silindiğine dair bir mesaj gösteriyoruz.

***Form 1: PERSONEL KAYIT İŞLEMLERİ***

***Zaman karmaşıklığı analizi:***

***button4\_Click metodu:*** personel\_kayit tablosundaki verileri seçer ve bir SqlDataReader kullanarak okur. Ardından, okunan verileri bir richTextBox2 kontrolüne yazdırır. Bu işlem, veri tabanı tablosundaki tüm verileri okumak için bir döngü kullanır ve her veriyi richTextBox2 kontrolüne ekler. Bu döngü, veri tabanının boyutuna bağlı olarak O(n) zaman karmaşıklığına sahip olabilir, burada n veri tabanındaki kayıt sayısıdır.

***button5\_Click metodu:*** personel\_kayit tablosuna yeni bir kayıt ekler. Bu işlem, SqlCommand nesnesini kullanarak veri tabanına bir sorgu gönderir ve komut.ExecuteNonQuery() çağrısı ile sorguyu çalıştırır. Bu işlem, tek bir sorgu çalıştırdığı için O(1) zaman karmaşıklığına sahiptir.

***button6\_Click metodu****:* personel\_kayit tablosundan belirli bir kaydı siler. Bu işlem, SqlCommand nesnesini kullanarak silme sorgusunu yürütür ve silKomutu.ExecuteNonQuery() çağrısı ile sorguyu çalıştırır. Bu işlem de tek bir sorgu çalıştırdığı için O(1) zaman karmaşıklığına sahiptir.

***Bellek Karmaşıklığı Analizi:***

***button4\_Click metodu:****Bir SqlConnection nesnesi oluşturuluyor.SqlConnection nesnesi Open metodu ile veritabanına bağlanıyor.Bir SqlCommand nesnesi oluşturuluyor ve bir SQL sorgusu tanımlanıyor.Sorgu sonucunda elde edilen veriler bir SqlDataReader nesnesine aktarılıyor. SqlDataReader üzerinde döngü oluşturularak veriler richTextBox2 kontrolüne ekleniyor.Son olarak SqlConnection nesnesi kapatılıyor.Bellek Karmaşıklığı: O(1) (Sabit)*

***button5\_Click metodu:***

*Bir SqlConnection nesnesi oluşturuluyor.SqlConnection nesnesi Open metodu ile veri tabanına bağlanıyor.Bir SQL INSERT sorgusu tanımlanıyor ve bir SqlCommand nesnesi oluşturuluyor.Parametreler ekleniyor. SqlCommand nesnesi ExecuteNonQuery metodu ile sorgu çalıştırılıyor. SqlConnection nesnesi kapatılıyor. Bellek Karmaşıklığı: O(1) (Sabit)*

***button6\_Click metodu:***

*Bir SqlConnection nesnesi oluşturuluyor.SqlConnection nesnesi Open metodu ile veritabanına bağlanıyor.Bir SQL SELECT sorgusu tanımlanıyor ve bir SqlCommand nesnesi oluşturuluyor.Parametre ekleniyor.SqlDataReader nesnesi oluşturuluyor ve sorgu sonucu verileri almak için kullanılıyor.Eğer sorgu sonucunda veri bulunursa, silme işlemi için bir onay mesajı gösteriliyor ve eğer kullanıcı onaylarsa bir SQL DELETE sorgusu çalıştırılıyor. SqlConnection nesnesi kapatılıyor. Bellek Karmaşıklığı: O(1) (Sabit)*

*Bellek karmaşıklığı çoğunlukla sabittir ve girişe bağlı olarak değişmez.*

***FORM 2: ÜRÜN KAYIT İŞLEMLERİ***

***Zaman Karmaşıklığı Analizi***

**button2\_Click Metodu (Ürün Ekleme):**

SqlConnection oluşturma işlemi: SqlConnection sınıfının oluşturulması bir sabit zaman işlemidir (O(1)).

SQL sorgusu çalıştırma işlemi: SqlCommand sınıfı kullanılarak veri tabanına bir sorgu gönderiliyor. Bu işlem de bir sabit zaman işlemidir (O(1)).

SqlConnection kapatma işlemi: Baglanti.Close() işlemi, SqlConnection nesnesini kapatır ve kaynakları serbest bırakır. Bu işlem de bir sabit zaman işlemidir (O(1)).

Toplam zaman karmaşıklığı: O(1)

**button1\_Click Metodu (Ürünleri Listeleme):**

SqlConnection oluşturma işlemi: SqlConnection sınıfının oluşturulması bir sabit zaman işlemidir (O(1)).

SqlCommand sorgusu çalıştırma işlemi: SqlCommand sınıfı kullanılarak veri tabanına bir sorgu gönderiliyor. Bu sorgu tüm kayıtları getireceği için, veri tabanındaki kayıt sayısıyla doğru orantılı olarak zaman alır. Diyelim ki kayıt sayısı n olsun, bu durumda zaman karmaşıklığı O(n) olur.

SqlDataReader ile kayıtları okuma işlemi: SqlDataReader sınıfı, sorgudan dönen sonuç kümesini tek tek okuyarak işler. Veri okuma işlemi sırasında her bir kayıt için bir kez döngü yapılır ve bu döngüdeki işlemler sabit zamanlıdır. Bu nedenle, zaman karmaşıklığı O(n) olur.

Verileri richTextBox'a yazma işlemi: RichTextBox'a yazma işlemi, metnin uzunluğuna bağlı olarak zaman alır. Ancak, bu aşamada metnin uzunluğu, kayıt sayısından daha önemsizdir ve sürekli olarak üzerine yazma yapıldığından, zaman karmaşıklığını etkilemez. Dolayısıyla, zaman karmaşıklığı O(n) olur.

SqlConnection kapatma işlemi: SqlConnection nesnesinin kapatılması bir sabit zaman işlemidir (O(1)).

Toplam zaman karmaşıklığı: O(n)

**button3\_Click Metodu (Ürün Silme):**

SqlConnection oluşturma işlemi: SqlConnection sınıfının oluşturulması bir sabit zaman işlemidir (O(1)).

**SqlCommand sorgusu çalıştırma işlemi:** SqlCommand sınıfı kullanılarak veritabanına bir sorgu gönderiliyor. Bu işlem bir sabit zaman işlemidir (O(1)).

SqlDataReader ile kaydı kontrol etme işlemi: SqlDataReader sınıfı, sorgudan dönen sonuç kümesini tek tek okuyarak işler. Bu işlem, sonucun tek bir kayıt döndürdüğü durumlarda bile sadece bir kez gerçekleşir. Dolayısıyla, zaman karmaşıklığı O(1) olur.

**MessageBox gösterme işlemi:** MessageBox.Show() metodu, bir mesaj kutusu görüntülemek için kullanılır ve sabit zamanlı bir işlemdir (O(1)).

Silme işlemi (DELETE): DELETE sorgusu, veri tabanında bir kaydın silinmesi için kullanılır ve sabit zamanlı bir işlemdir (O(1)).

**SqlConnection kapatma işlemi:** SqlConnection nesnesinin kapatılması bir sabit zaman işlemidir (O(1)).

Toplam zaman karmaşıklığı: O(1)

Diğer olay işleyicileri ve metotlar genellikle tek bir işlem yapar ve bu işlemler sabit zamanlıdır (O(1)).

Özetle, button1\_Click metodu dışındaki diğer metotlar ve olay işleyicileri sabit zamanlıdır. button1\_Click metodu, kayıt sayısına bağlı olarak zaman karmaşıklığı O(n)'dir.

***Bellek Karmaşıklığı Analizi:***

**Değişkenler:**

**SqlConnection baglanti;:** SqlConnection nesnesini temsil eder. Sabit miktarda bellek kullanır (O(1)).

**string sql:** SQL sorgusu için bir dize. Sabit miktarda bellek kullanır (O(1)).

**SqlCommand komut:** SqlCommand nesnesini temsil eder. Sabit miktarda bellek kullanır (O(1)).

**SqlDataReader sqlDR:** SqlDataReader nesnesini temsil eder. Okunan kayıtların bellekte tutulduğu yerdir. Bellek kullanımı, okunan kayıt sayısına bağlı olarak artar (O(n)).

**Form bileşenleri**:

**ID, AD, Fiyat, RID gibi TextBox nesneleri:** Girilen metinleri temsil eder. Metin uzunluğuyla doğru orantılı olarak bellek kullanır (O(1)).

**richTextBox1:** Ürün listesini temsil eden RichTextBox nesnesi. Listelenen kayıtların toplam metin uzunluğuna bağlı olarak bellek kullanır. Eğer kayıt sayısı n ise bellek kullanımı O(n) olabilir.

**Veritabanı Bağlantısı:**

Veritabanı bağlantısı için bir SqlConnection nesnesi oluşturulur. Bu nesne bellekte bir bağlantıyı temsil eder ve sabit miktarda bellek kullanır (O(1)).

***FORM: GİRİS***

Giris sınıfının yapıcı metodu (public Giris()) içerisinde formun başlatılması ve bileşenlerin oluşturulması işlemleri yer almaktadır. Bu kısım, sabit bir sürede çalışır ve O(1) zaman karmaşıklığına sahiptir.

button1\_Click adlı olay işleyicisi metodu, "Giriş" butonuna tıklandığında çalışır. Bu metot içerisinde aşağıdaki işlemler gerçekleştirilir:

SqlConnection nesnesinin oluşturulması ve veritabanına bağlanması (baglanti.Open()). Bu işlem, veritabanı bağlantısının kurulması ve O(1) zaman karmaşıklığına sahiptir.

İki adet SQL sorgusu tanımlanması (secmeSorgusu ve secmeSorgusu2). Bu işlem, sabit bir sürede gerçekleşir ve O(1) zaman karmaşıklığına sahiptir.

Kullanıcı tarafından TextBox'lara girilen e-posta (mail) ve şifre (sifre) değerlerinin alınması. Bu işlem, sabit bir sürede gerçekleşir ve O(1) zaman karmaşıklığına sahiptir.

SqlCommand nesnelerinin oluşturulması ve parametrelerin ayarlanması (cmd ve cmd2). Bu işlem, sabit bir sürede gerçekleşir ve O(1) zaman karmaşıklığına sahiptir.

İlk SQL sorgusunun çalıştırılması (cmd.ExecuteReader()) ve SqlDataReader nesnesinin oluşturulması (dr2). Bu işlem, veritabanından dönen sonuçları okumak için kullanılır ve sonuç kümesinin boyutuna bağlı olarak O(n) zaman karmaşıklığına sahiptir. Burada n, yonetici\_kayit tablosundaki satır sayısıdır.

SqlDataReader ile veritabanından alınan sonuçların bir diziye kopyalanması (while (dr2.Read())). Bu işlem, veritabanından dönen sonuçları bir diziye kopyalama işlemidir. Sonuç kümesinin boyutuna bağlı olarak O(n) zaman karmaşıklığına sahiptir.

İkinci bir döngü aracılığıyla dizideki değerlerin kontrol edilmesi (for (int j = 0; j < list.Length; j++)). Bu işlem, dizideki değerleri kontrol ederek e-postanın doğruluğunu kontrol eder.

Kontrol işlemi (if (list[j] == mail)) için her bir dizi elemanıyla e-postanın karşılaştırılması yapılır. Bu işlem, dizi boyutuna bağlı olarak O(n) zaman karmaşıklığına sahiptir.

Eğer e-posta doğrulanırsa (sonuc = 1), kontrol döngüsünden çıkılır.

Eğer e-posta doğrulanamazsa (sonuc = 0), "E-Mail veya Parola Hatalı" mesajı görüntülenir.

İlk SQL sorgusundan dönen sonuçları okumak için SqlDataReader nesnesi kullanılır (dr). Bu işlem, veritabanından dönen sonuç kümesinin boyutuna bağlı olarak O(n) zaman karmaşıklığına sahiptir.

SqlDataReader ile parolanın doğruluğu kontrol edilir (while (dr.Read())). Bu işlem, veritabanından dönen sonuçları okuyarak doğru parolanın olup olmadığını kontrol eder. Sonuç kümesinin boyutuna bağlı olarak O(n) zaman karmaşıklığına sahiptir.

Parola doğrulanırsa, "Giriş Başarılı" mesajı görüntülenir ve Secim formu oluşturularak gösterilir. Bu işlem sabit bir sürede gerçekleşir ve O(1) zaman karmaşıklığına sahiptir.

Parola doğrulanamazsa, "E-Mail veya Parola Hatalı" mesajı görüntülenir.

**Bellek Karmaşıklığı Analizi:**

`Giris` sınıfı, bir Windows Forms formu temsil eder. Bu form, form elemanları ve gerekli nesnelerin tanımlanmasıyla birlikte bir bellek alanı oluşturur. Bu bellek alanı, formun oluşturulduğu sürece kullanılır ve form kapatıldığında bellekten serbest bırakılır. Bu bellek kullanımı, form elemanları ve diğer tanımlanan nesnelere bağlı olarak değişebilir.

`button1\_Click` adlı olay işleyicisi metodu, "Giriş" butonuna tıklandığında çalışır. Bu metodun içinde aşağıdaki nesnelerin bellekte kullanımı vardır:

SqlConnection nesnesi (baglanti): Veritabanı bağlantısını temsil eder ve bellekte bir alan kaplar.

SqlCommand nesneleri (`cmd` ve `cmd2`): SQL sorgularını temsil eder ve bellekte bir alan kaplar.

`SqlDataReader` nesneleri (`dr` ve `dr2`): Veritabanından dönen sonuçları okumak için kullanılır ve bellekte bir alan kaplar.

string türündeki değişkenler (`secmeSorgusu`, `secmeSorgusu2`, `mail`, `sifre`): Kullanıcı giriş bilgilerini ve SQL sorgularını temsil eder ve bellekte bir alan kaplar.

string türündeki dizi (`list`): Veritabanından dönen sonuçların depolandığı dizi, bellekte bir alan kaplar.

***FORM: SEÇİM***

**Zaman Karmaşıklığı Analizi:**

**Secim sınıfının tanımlanması:**

Bu sınıf, Windows Forms uygulamasının ana formunu temsil eder. Sınıfın tanımlanması zaman karmaşıklığına etki etmez.

Secim formunun oluşturulması ve bileşenlerinin eklenmesi:

Secim sınıfının bir örneği oluşturulur ve bileşenleri (düğmeler) bu form üzerine eklenir. Bu adımda, formun oluşturulması ve bileşenlerin eklenmesi işlemleri sabit zamanda gerçekleşir ve zaman karmaşıklığına etki etmez.

**button1\_Click olay işleyicisinin tanımlanması:**

Bu metot, "button1" düğmesine tıklandığında çalışacak olan olay işleyicisini temsil eder. Form2 sınıfından bir örnek oluşturulur ve bu örnek bir iletişim kutusu olarak görüntülenir. ShowDialog metodu, diyalog kutusunun kullanıcı tarafından kapatılana kadar beklemesini sağlar. Bu adımda, bir örnek oluşturma ve diyalog kutusu görüntüleme işlemleri sabit zamanda gerçekleşir.

**button2\_Click olay işleyicisinin tanımlanması:**

Bu metot, "button2" düğmesine tıklandığında çalışacak olan olay işleyicisini temsil eder. Form1 sınıfından bir örnek oluşturulur ve bu örnek bir iletişim kutusu olarak görüntülenir. ShowDialog metodu, diyalog kutusunun kullanıcı tarafından kapatılana kadar beklemesini sağlar. Bu adımda, bir örnek oluşturma ve diyalog kutusu görüntüleme işlemleri sabit zamanda gerçekleşir ve zaman karmaşıklığına etki etmez.