SAKARYA
ÜNİVERSİTESİ

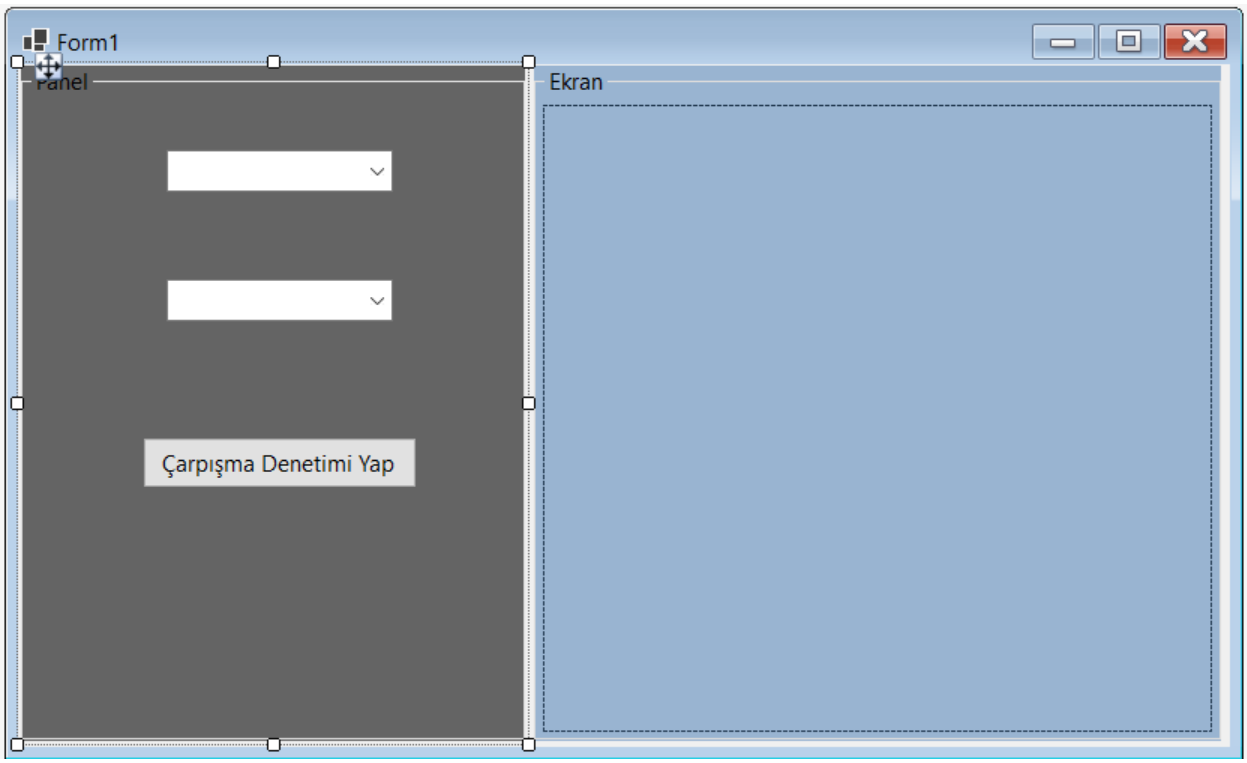
2022-2023

Nesneye Dayalı Programlama

Geometrik Nesnelerin Çarpışma Denetimi

Younes Rahebi – B221210588
BÖLÜM : BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
DERS GRUBU : B GRUBU 1.Öğretim
YOUTUBE LİNKİ : https://youtu.be/hjA1cJUikO8

Uygulama çalıştırıldığında, bir panel, iki adet combobox, bir button ve bir label oluşan bir form oluşturur. Kullanıcı, comboboxlardan iki geometrik nesne seçebilir ve panel üzerinde onları çizmek için button'a tıklayabilir.



Düğmeye tıklandığında, "**button_Click**" olay işleyicisi çağrılır. Bu yöntem öncelikle paneli temizler ve açılır kutulardan seçilen öğeleri alır. Daha sonra iki geometrik nesne için rastgele koordinatlar ve boyutlar üretir ve seçilen öğelere göre ilgili sınıfların örneklerini oluşturur. Her nesne üzerinde **Draw** yöntemi çağrılır ve panel üzerinde çizilir.

```
private void button_Click(object sender, EventArgs e)
{
    Graphics g = panel.CreateGraphics();
    g.Clear(panel.BackColor);

    // Combobox'tan seçilen öğeyi al
    string selectedItem = (string)comboBox.SelectedItem;
    string selectedItem1 = (string)comboBox1.SelectedItem;

    // Rastgele boyutlar ve koordinatlar üret
    int x = random.Next(0, panel.Width);
    int y = random.Next(0, panel.Height);

    Nesne nesne1 = null;

    // Seçilen geometrik nesneyi rastgele boyutlar ve koordinatlarla çiz
    if (selectedItem == "Nokta")
    {
        Nokta nokta = new Nokta(x, y);
        nokta.Draw(panel.CreateGraphics());
        nesne1 = nokta;
    }
```

Nesneler çizildikten sonra, **Carpisma** yöntemi kullanılarak çarpışmaları kontrol edilir. Eğer herhangi bir nesnenin **Carpisma** yöntemi **true** döndürürse, yani nesnelerin çarpıştığını belirtirse, etiketin metni “Çarpısma Var” olarak ayarlanır ve arka plan rengi yeşile ayarlanır. Aksi halde, etiketin metni “Çarpısma Yok” olarak ayarlanır ve arka plan rengi kırmızıya ayarlanır.

```
        // Seçilen nesnelerin kesişip kesişmediğini kontrol ed
        if (nesne1 != null && nesne2 != null)
        {
            if (nesne1.Carpisma(nesne2) || nesne2.Carpisma(nesne1))
            {
                label.BackColor = Color.Green;
                label.Text = "Çarpısma Var";
            }
            else
            {
                label.BackColor = Color.Red;
                label.Text = "Çarpısma Yok";
            }
        }
        else
        {
            label.BackColor = Color.Yellow;
            label.Text = "Nesneleri Seçiniz";
        }
```

Kod ayrıca farklı geometrik nesneleri temsil eden birkaç sınıf içerir: **Nokta**, **Dikdortgen**, **Daire**, **Kure**, **Yuzey**, **Prizma** ve **Silindir**. Her sınıf, nesnenin koordinatları için özelliklere ve nesneyi çizmek ve başka bir nesne ile çarpışmayı kontrol etmek için soyut yöntemlere sahip olan soyut bir temel sınıf olan **Nesne** sınıfından türetilmiştir. Her türetilmiş sınıf bu yöntemleri kendi yollarıyla uygular.

```
public class Dikdortgen : Nesne
{
    10 references
    public int Width { get; set; }

    10 references
    public int Height { get; set; }

    2 references
    public Dikdortgen(int x, int y, int width, int height) : base(x, y)
    {
        Width = width;
        Height = height;
    }

    3 references
    public override void Draw(Graphics graphics)
    {
        graphics.DrawRectangle(Pens.Black, X, Y, Width, Height);
    }

    10 references
    public override bool Carpisma(Nesne nesne)
    {
        if (nesne is Dikdortgen dikdortgen)
        {
            return X < dikdortgen.X + dikdortgen.Width && X + Width > dikdortgen.X && Y < dikdortgen.Y + dikdortgen.Height && Y + Height > dikdortgen.Y;
        }
        else if (nesne is Daire daire)
        {
            // Dairenin merkezine en yakın dikdörtgen noktasını bul
            int closestX = Math.Max(X, Math.Min(daire.X, X + Width));
            int closestY = Math.Max(Y, Math.Min(daire.Y, Y + Height));

            // Bu nokta ile dairenin merkezi arasındaki mesafeyi hesapla
            int distanceX = daire.X - closestX;
            int distanceY = daire.Y - closestY;
            int distanceSquared = distanceX * distanceX + distanceY * distanceY;

            // Bu mesafenin dairenin yarıçapından küçük veya eşit olup olmadığını kontrol ed
            return distanceSquared <= daire.Radius * daire.Radius;
        }
```

Örneğin, **Dikdortgen** sınıfı genişlik ve yüksekliğe sahip bir dikdörtgeni temsil eder. **Draw** yöntemi, **Graphics** sınıfının **DrawRectangle** yöntemini kullanarak panel üzerinde bir dikdörtgen çizer. **Carpisma** yöntemi ise başka bir nesnenin kendisi ile çarpışıp çarpışmadığını kontrol ederken sınırlarının üst üste gelip gelmediğini kontrol eder.