Oyuncu Sınıfı

```
using System;
using System.Collections;

namespace Tombala {

	/// <summary>
	/// Tombala oyunundaki her bir oyuncu
	/// </summary>
	class Oyuncu {

		/// <summary>
		/// Oyuncumuzun elindeki kart
		/// </summary>
		private ArrayList kart = new ArrayList(6);

		/// <summary>
		/// Oyuncumuzun elindeki kartın yedeği
		/// </summary>
		private ArrayList yedekKart = new ArrayList(6);

		/// <summary>
		/// Oyuncumuz oluşturulduğu anda ilk işlemler yapılmakta
		/// </summary>
		public Oyuncu() {

			kart.Add(0);
			kart.Add(0);
			kart.Add(0);
			kart.Add(0);
			kart.Add(0);
			kart.Add(0);

			yedekKart.AddRange(kart);
		}

		/// <summary>
		/// Oyuncunun elindeki kartın üzerindeki [index]'inci numarayı belirleme
		/// </summary>
		/// <param name="index">Oyuncunun elindeki [index]'inci numara</param>
		/// <param name="value">[index]'inci numaranın değeri</param>
		public void setKart(int index, int value) {

			kart[index] = value;
			yedekKart[index] = value;
		}

		/// <summary>
		/// Oyuncunun elindeki kartın [index]'inci numarayı Integer olarak almak
için
		/// </summary>
		/// <param name="index">[index]'inci numara</param>
		/// <returns>[index]'inci numaranın değeri</returns>
		public int getKartInt(int index) {

			return Convert.ToInt32(kart[index]);
		}

		/// <summary>
		/// Oyuncunun elindeki kartın [index]'inci numarayı String olarak almak
için
		/// </summary>
```

```
            /// <param name="index">[index]'inci numaranın değeri</param>
            /// <returns>[index]'inci numaranın değeri</returns>
            public string getKartString(int index) {

                  return Convert.ToString(kart[index]);
            }

            /// <summary>
            /// Oyuncunun elindeki yedek kartın [index]'inci numarayı String olarak
almak için
            /// </summary>
            /// <param name="index">[index]'inci numara</param>
            /// <returns>[index]'inci numaranın değeri</returns>
            public int getYedekKartInt(int index) {

                  return Convert.ToInt32(yedekKart[index]);
            }

            /// <summary>
            /// Oyuncunun elindeki yedek kartın [index]'inci numarayı String olarak
almak için
            /// </summary>
            /// <param name="index">[index]'inci numaranın değeri</param>
            /// <returns>[index]'inci numaranın değeri</returns>
            public string getYedekKartString(int index) {

                  return Convert.ToString(yedekKart[index]);
            }

            /// <summary>
            /// Oyuncunun elindeki [index]'inci numarasını 0 lamak için
            /// </summary>
            /// <param name="index">[index]'inci numara</param>
            public void kapat(int index) {

                  kart[index] = 0;
            }

            /// <summary>
            /// Oyuncunun elindeki kartın üzerinde x sayısı var mı?
            /// </summary>
            /// <param name="x">1 - 99 arasında bir sayı</param>
            /// <returns>x sayısı oyuncunun elindeki kartta varsa true
döner</returns>
            public bool isVar(int x) {

                  for (int i = 0; i < 6; i++)
                        if (getKartInt(i) == x)
                              return true;

                  return false;
            }

            /// <summary>
            /// Oyuncunun elindeki kartın üzerindeki x sayısının index i nedir?
            /// </summary>
            /// <param name="x">1 - 99 arasında bir sayı</param>
            /// <returns>x sayısı oyuncunun elindeki kartta varsa index döner, yoksa
-1 döner</returns>
            public int isVarInt(int x) {

                  for (int i = 0; i < 6; i++)
                        if (getKartInt(i) == x)
                              return i;
```

```
                return -1;
            }

            /// <summary>
            /// Oyuncunun elindeki kartın numaralarını sıralamak için
            /// </summary>
            public void kartSirala() {

                kart.Sort();
            }

            /// <summary>
            /// Oyuncunun elindeki yedek kartın numaralarını sıralamak için
            /// </summary>
            public void yedekKartSirala() {

                yedekKart.Sort();
            }

            /// <summary>
            /// Oyuncunun elindeki kartın üzerindeki numaraların tamamı 0 mı?
            /// </summary>
            /// <returns>0 ise true</returns>
            public bool isBitti() {

                for (int i = 0; i < 6; i++)
                    if (getKartInt(i) != 0)
                        return false;

                return true;
            }
        }
}
```

Oyuncular Sınıfı

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Collections;
using System.Drawing;

namespace Tombala {

    /// <summary>
    /// Tombala oyunundaki oyuncularımızı bir arada tutun sınıfımız
    /// </summary>
    class Oyuncular {

        /// <summary>
        /// Oyunumuzdaki oyuncularımız
        /// </summary>
        private ArrayList oyuncular = new ArrayList(16);

        /// <summary>
        /// Oyunumuzdaki kartlarımız
        /// </summary>
        private ArrayList kartlar = new ArrayList(16);

        /// <summary>
        /// Oyunculara numara dağıtırken kullanılan liste
        /// </summary>
```

```
        private ArrayList cekilenSayilar = new ArrayList(100);

        /// <summary>
        /// Oyunda torbadan çekilmiş sayıların depolandığı liste
        /// </summary>
        private ArrayList cekilmisSayilar = new ArrayList(100);

        /// <summary>
        /// Oyuncularımızı oluştururken ilk işlemler yapılıyor
        /// </summary>
        /// <param name="x">Oyunumuzdaki oyuncu sayısı</param>
        public Oyuncular(int x) {

              for (int i = 0; i < x; i++) {

                    oyuncular.Add(new Oyuncu());
                    kartlar.Add(new FormKart());
              }
        }

        /// <summary>
        /// Output dosyasına cekilmiş sayıları vermek için kullanılan metod
        /// </summary>
        /// <returns>Oyun boyunca çekilmiş sayılar</returns>
        public string getCekilmisSayilar() {

              string s = "";

              cekilmisSayilar.Sort();

              for (int i = 0; i < cekilmisSayilar.Count; i++) {

                    s = s + (Convert.ToString(cekilmisSayilar[i]) + ", ");
              }

              return s.Substring(0, s.Length - 2);
        }

        /// <summary>
        /// Oyundaki oyuncu sayımız nedir?
        /// </summary>
        /// <returns>Oyuncu sayısı</returns>
        public int getOyuncuSayisi() {

              return oyuncular.Count;
        }

        /// <summary>
        /// Oyunumuzdaki [index]'inci oyuncu
        /// </summary>
        /// <param name="index">[index]'inci oyuncu</param>
        /// <returns>Oyuncu</returns>
        public Oyuncu getOyuncu(int index) {

              return (Oyuncu) oyuncular[index];
        }

        /// <summary>
        /// Oyunculara numaralarını dağıtmak için kullanılan metod
        /// </summary>
        public void numaralariDagit() {

              Random r = new Random();
```

```
                int s;

                for (int i = 0; i < getOyuncuSayisi(); i++) {

                        for (int j = 0; j < 6; j++) {

                                do {

                                        s = r.Next(1, 99);
                                } while (isCekilmis(s));

                                getOyuncu(i).setKart(j, s);

                                cekilenSayilar.Add(s);
                        }

                        getOyuncu(i).kartSirala();
                }
        }

        /// <summary>
        /// Oyuncuya numaralar dağıtılırken [x] sayısı daha önce verilmiş mi?
        /// </summary>
        /// <param name="x">Rasgele çekilen sayı</param>
        /// <returns>Çekildiyese true</returns>
        private bool isCekilmis(int x) {

                for (int i = 0; i < cekilenSayilar.Count; i++)
                        if (Convert.ToInt32(cekilenSayilar[i]) == x)
                                return true;

                return false;
        }

        /// <summary>
        /// Oyuncularımızın [index]'inci kartı
        /// </summary>
        /// <param name="index">[index]'inci kart</param>
        /// <returns>Kart</returns>
        public FormKart getFormKart(int index) {

                return (FormKart) (kartlar[index]);
        }

        /// <summary>
        /// Torbadan rasgele yeni sayı çekmek için
        /// </summary>
        /// <returns>Torbadan bir sayı</returns>
        public int sayiCek() {

                return getSayiCek();
        }

        /// <summary>
        /// Rasgele bir sayı çekme, çekerken daha önce çekilip çekilmediği
kontrol ediliyor
        /// </summary>
        /// <returns>Rasgele çekilen bir sayı</returns>
        private int getSayiCek() {

                Random rand = new Random();

                int s;
```

```
                do {

                        s = rand.Next(98) + 1;
                } while (cekilmisSayilar.Contains(s));

                cekilmisSayilar.Add(s);

                return s;
        }

        /// <summary>
        /// [x] sayısı oyunumuzdaki herhangi bir oyuncuda var mı?
        /// </summary>
        /// <param name="x">Bir sayı</param>
        /// <returns>[x] sayısı varsa true</returns>
        public bool isVar(int x) {

                for (int i = 0; i < getOyuncuSayisi(); i++)
                        if (getOyuncu(i).isVar(x))
                                return true;

                return false;
        }

        /// <summary>
        /// [x] sayısı varsa kaçıncı kişide var
        /// </summary>
        /// <param name="x">Bir sayı</param>
        /// <returns>[x] sayısının kaçıncı kişideyse, değilse -1</returns>
        public int isVarInt(int x) {

                for (int i = 0; i < getOyuncuSayisi(); i++)
                        if (getOyuncu(i).isVar(x))
                                return i;

                return -1;
        }

        /// <summary>
        /// Kart üzerindeki yazan sayıları yenilemek için kullanılan metod
        /// </summary>
        /// <param name="x">Hangi kişinin kartı yenilenecek</param>
        public void kartYenile(int x) {

                for (int i = 0; i < getOyuncuSayisi(); i++) {

                        getOyuncu(i).kartSirala();
                        for (int j = 0; j < 6; j++) {

                                getFormKart(i).setNumaraText(j,
getOyuncu(i).getKartString(j));
                        }
                }
        }

        /// <summary>
        /// Oyun bittiğinde kullanılan metod
        /// </summary>
        /// <param name="x">Hangi kişi oyunu kazandı</param>
        public void bitti(int x) {

                getOyuncu(x).yedekKartSirala();

                for (int i = 0; i < 6; i++) {
```

```
                        getFormKart(x).setNumaraText(i,
getOyuncu(x).getYedekKartString(i));
                        getFormKart(x).setNumaraBackColor(i, Color.Yellow);
                        getFormKart(x).setNumaraForeColor(i, Color.Blue);
                        getFormKart(x).setNumaraFont(i, new Font("verdana", 8,
FontStyle.Bold));
                  }

                  getFormKart(x).setBaslik("Kazanan: " + (x + 1));
            }
      }
}
```

FormAna Sınıfı

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace Tombala {

      public partial class FormAna : Form {

            public FormAna() {

                  InitializeComponent();
            }

            private Oyuncular oyuncularim;

            private bool oyunBasladi = false;
            private bool oyunBitti = true;

            private Oyuncular getOyuncularim() {

                  return oyuncularim;
            }

            private void tiklaOyunaBasla(object sender, EventArgs e) {

                  if ((oyunBasladi) || (!oyunBitti)) {

                        hataVer("Öncelikle aktif oyunu bitirmelisiniz!");
                  } else {

                        oyunaBasla(Convert.ToInt32(nudOyuncuSayisi.Value), false);
                        oyunBasladi = true;
                        oyunBitti = false;
                  }
            }

            private void hataVer(string hata) {

                  MessageBox.Show(hata, "Hata", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
```

```
            private void bilgiVer(string bilgi) {

                  MessageBox.Show(bilgi, "Bilgi", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }

            private void oyunaBasla(int x, bool yeniOyun) {

                  if (!yeniOyun) {


                  }

                  oyuncularim = new Oyuncular(x);

                  oyuncularim.numaralariDagit();

                  kartlariDagit();

                  tlpOyunAlani.Visible = true;
            }

            private void kartlariDagit() {

                  for (int i = 0; i < getOyuncularim().getOyuncuSayisi(); i++) {

                        sayilariKartlaraYaz(i);
                        tlpOyunAlani.Controls.Add(getOyuncularim().getFormKart(i));
                  }
            }

            private void sayilariKartlaraYaz(int x) {

                  getOyuncularim().getFormKart(x).setBaslik((x + 1) + ". Oyuncu");

                  for (int i = 0; i < 6; i++)

                        getOyuncularim().getFormKart(x).setNumaraText(i,
getOyuncularim().getOyuncu(x).getKartString(i));
            }

            private void tiklaOyunuBitir(object sender, EventArgs e) {

                  if ((oyunBasladi) || (!oyunBitti)) {

                        if (MessageBox.Show("Oyunu bitirmek istesiğinizden emin
misiniz?", "Tombala Oyunu", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes) {

                              for (int i = 0; i < getOyuncularim().getOyuncuSayisi();
i++) {


      tlpOyunAlani.Controls.Remove(getOyuncularim().getFormKart(i));
                              }

                              tlpOyunAlani.Visible = false;
                              oyunBasladi = false;
                              oyunBitti = true;
                        }
                  } else {

                        hataVer("Öncelikle oyuna başlamalısınız.");
                  }
```

```
            }

            private void tiklaHakkinda(object sender, EventArgs e) {

                  FormHakkinda fhHakkinda = new FormHakkinda();
                  fhHakkinda.Show();
            }

            private void tiklaTimerBilgi(object sender, EventArgs e) {

                  Random rand = new Random();

                  int r = rand.Next(150);// +105;
                  int g = rand.Next(150);// + 105;
                  int b = rand.Next(150);// + 105;

                  Color c = Color.FromArgb(r, g, b);

                  lblBilgi.BackColor = c;
            }

            private void tiklaSayiCek(object sender, EventArgs e) {

                  if ((oyunBasladi) && (!oyunBitti)) {

                        int s = getOyuncularim().sayiCek();

                        MessageBox.Show("Çekilen Sayı: " + s, "Yeni Sayı Çekildi",
MessageBoxButtons.OK);

                        int kacinciOyuncu = getOyuncularim().isVarInt(s);

                        if (kacinciOyuncu == -1) {

                              bilgiVer(s + " sayısı hiç bir oyuncuda yok");
                        } else {

                              int kacinciSayi =
getOyuncularim().getOyuncu(kacinciOyuncu).isVarInt(s);

                              bilgiVer(s + " sayısı, " + (kacinciOyuncu + 1) + ".
oyuncunun " + (kacinciSayi + 1) + ". sayısıdır.");


      getOyuncularim().getOyuncu(kacinciOyuncu).kapat(kacinciSayi);

                              getOyuncularim().kartYenile(kacinciOyuncu);


      getOyuncularim().getFormKart(kacinciOyuncu).setKapali();

                              if
(getOyuncularim().getOyuncu(kacinciOyuncu).isBitti()) {

                                    bilgiVer("Oyunu " + (kacinciOyuncu + 1) + ".
oyuncu kazanmıştır.");

                                    getOyuncularim().bitti(kacinciOyuncu);

                                    oyunBitti = true;
```

```
                                    if (MessageBox.Show("Sonuçları kaydetmek ister
misini?", "Sonuçları Kaydet", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes) {

                                          sonuclariKaydet(kacinciOyuncu);
                                    } else {


                                    }
                              }
                        } else {

                              hataVer("Yeni oyun başlatın.");
                        }
                  }

                  private void sonuclariKaydet(int kazanan) {

                        SaveFileDialog sfdSonuclariKaydet = new SaveFileDialog();

                        if (sfdSonuclariKaydet.ShowDialog() == DialogResult.OK) {

                              sfdSonuclariKaydet.Title = "Kaydedilecek yer seçiniz";
                              StreamWriter sw = File.CreateText(sfdSonuclariKaydet.FileName
+ ".txt");

                              sw.WriteLine("Sonuçlar");
                              sw.WriteLine("");
                              sw.WriteLine("Oyunu " + (kazanan + 1) + ". oyuncu
kazanmıştır");
                              sw.WriteLine("");
                              sw.WriteLine("Kazanan Sayılar");
                              sw.WriteLine("---------------");
                              for (int i = 0; i < 6; i++) {

                                    sw.Write(" | " +
getOyuncularim().getOyuncu(kazanan).getYedekKartString(i));
                              }
                              sw.WriteLine(" | ");
                              sw.WriteLine("");
                              sw.WriteLine("Oyunda çekilen sayılar");
                              sw.WriteLine("----------------------");
                              sw.WriteLine(getOyuncularim().getCekilmisSayilar());
                              sw.Close();
                        }
                  }
            }
}
```

FormHakkinda Sınıfı

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;
using System.Reflection;

namespace Tombala
{
    partial class FormHakkinda : Form
    {
        public FormHakkinda()
```

```
        {
            InitializeComponent();

            //  Initialize the AboutBox to display the product information from the
assembly information.
            //  Change assembly information settings for your application through
either:
            //  - Project->Properties->Application->Assembly Information
            //  - AssemblyInfo.cs
            this.Text = String.Format("{0}", AssemblyTitle);
            this.labelProductName.Text = AssemblyProduct;
            this.labelVersion.Text = String.Format("Versiyon {0}", AssemblyVersion);
            this.labelCopyright.Text = AssemblyCopyright;
            this.labelCompanyName.Text = AssemblyCompany;
            this.textBoxDescription.Text = AssemblyDescription;
        }

        #region Assembly Attribute Accessors

        public string AssemblyTitle
        {
            get
            {
                // Get all Title attributes on this assembly
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute),
false);
                // If there is at least one Title attribute
                if (attributes.Length > 0)
                {
                    // Select the first one
                    AssemblyTitleAttribute titleAttribute =
(AssemblyTitleAttribute)attributes[0];
                    // If it is not an empty string, return it
                    if (titleAttribute.Title != "")
                        return titleAttribute.Title;
                }
                // If there was no Title attribute, or if the Title attribute was the
empty string, return the .exe name
                return
System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);
            }
        }

        public string AssemblyVersion
        {
            get
            {
                return Assembly.GetExecutingAssembly().GetName().Version.ToString();
            }
        }

        public string AssemblyDescription
        {
            get
            {
                // Get all Description attributes on this assembly
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribu
te), false);
                // If there aren't any Description attributes, return an empty string
                if (attributes.Length == 0)
                    return "";
                // If there is a Description attribute, return its value
```

```
                return ((AssemblyDescriptionAttribute)attributes[0]).Description;
            }
        }

        public string AssemblyProduct
        {
            get
            {
                // Get all Product attributes on this assembly
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute),
false);
                // If there aren't any Product attributes, return an empty string
                if (attributes.Length == 0)
                    return "";
                // If there is a Product attribute, return its value
                return ((AssemblyProductAttribute)attributes[0]).Product;
            }
        }

        public string AssemblyCopyright
        {
            get
            {
                // Get all Copyright attributes on this assembly
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute
), false);
                // If there aren't any Copyright attributes, return an empty string
                if (attributes.Length == 0)
                    return "";
                // If there is a Copyright attribute, return its value
                return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;
            }
        }

        public string AssemblyCompany
        {
            get
            {
                // Get all Company attributes on this assembly
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute),
false);
                // If there aren't any Company attributes, return an empty string
                if (attributes.Length == 0)
                    return "";
                // If there is a Company attribute, return its value
                return ((AssemblyCompanyAttribute)attributes[0]).Company;
            }
        }
        #endregion

        private void tiklaKapt(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

FormKart Sınıfı

```
using System.Drawing;
using System;
namespace Tombala
{
    partial class FormKart
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Component Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.baslik = new System.Windows.Forms.Label();

            for (int i = 0; i < 6; i++)
            {
                this.numaralar[i] = new System.Windows.Forms.Label();
                this.numaralar[i].BackColor =
System.Drawing.SystemColors.ControlLightLight;
                this.numaralar[i].BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
                this.numaralar[i].Name = "numara" + i;
                this.numaralar[i].Size = new System.Drawing.Size(32, 32);
                this.numaralar[i].TabIndex = i + 1;
                this.numaralar[i].TextAlign =
System.Drawing.ContentAlignment.MiddleCenter;
                this.Controls.Add(this.numaralar[i]);
            }

            this.numaralar[0].Location = new System.Drawing.Point(8, 40);
            this.numaralar[1].Location = new System.Drawing.Point(48, 40);
            this.numaralar[2].Location = new System.Drawing.Point(88, 40);
            this.numaralar[3].Location = new System.Drawing.Point(8, 80);
            this.numaralar[4].Location = new System.Drawing.Point(48, 80);
            this.numaralar[5].Location = new System.Drawing.Point(88, 80);

            this.SuspendLayout();
            //
            // kullaniciBilgisi
            //
```

```
            this.baslik.BackColor = System.Drawing.SystemColors.ActiveCaption;
            this.baslik.Dock = System.Windows.Forms.DockStyle.Top;
            this.baslik.Font = new System.Drawing.Font("Verdana", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.baslik.ForeColor =
System.Drawing.SystemColors.GradientActiveCaption;
            this.baslik.Location = new System.Drawing.Point(0, 0);
            this.baslik.Name = "kullaniciBilgisi";
            this.baslik.Size = new System.Drawing.Size(126, 32);
            this.baslik.TabIndex = 0;
            this.baslik.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
            //
            // FormKart
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.Controls.Add(this.baslik);
            this.Name = "FormKart";
            this.Size = new System.Drawing.Size(126, 118);
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Label[] numaralar = new
System.Windows.Forms.Label[6];

        private System.Windows.Forms.Label baslik;

            private void setKapali(int index) {

                  this.numaralar[index].BackColor = Color.Black;
                  this.numaralar[index].ForeColor = Color.White;
            }

            public void setKapali() {

                  for (int i = 0; i < 6; i++) {

                        if (Convert.ToInt32(this.numaralar[i].Text) == 0) {

                              this.setKapali(i);
                        }
                  }
            }

            public void setBaslik(string s) {

                  baslik.Text = s;
            }

            public void setNumaraText(int index, string s) {

                  numaralar[index].Text = s;
            }

            public void setNumaraBackColor(int index, Color c) {

                  numaralar[index].BackColor = c;
            }

            public void setNumaraForeColor(int index, Color c) {
```

```
                numaralar[index].ForeColor = c;
            }

            public void setNumaraFont(int index, Font f) {

                numaralar[index].Font = f;
            }
        }
}
```