Osnove programskog jezika C#

File, Directory, Stream
DateTime, TimeSpan,
Kontejneri, MDI

Rad sa datotekama i direktorijima

File

- Statička klasa. Sadrži samo statičke metode
  - Copy,
  - Create,
  - Delete,
  - Open,
  - OpenRead, OpenWrite,
  - Move,
  - Exists,GetCreationTime, GetLastAccessTime itd...

- Fajlovi su uvek povezani tokovima objekata pri čitanju ili upisivanju.

- Stream fs = File.Open(@"c:\temp\t1.txt",FileAccess.Read); // vraća FileStream
- File.Copy(@"c:\temp\source.txt",@"c:\temp\dest.txt");

FileInfo

- Klasa slična po nameni klasi *File*, sadrži dodatne informacije o fajlu. Ova klasa je optimizovana za višestruki pristup fajlu za razliku od klase File koja je optimizovana za jedan pristup.

```
FileInfo fr = new FileInfo(@"c:\temp\t1.txt"); // input file
StreamReader reader = fr.OpenText();  // OpenText ret StreamReader

    string file = "c:\\temp\\t1.txt";
    FileInfo f1 = new FileInfo (file);
    if (f1.Exists) {
          return;    // do something with the file
    }
    throw new FileNotFoundException("No file called:"+file);
```

Directory

- Sadrži samo statičke metode.
  - CreateDirectory, Delete, GetFiles, Exists,
  - GetCurrentDirectory, Move, SetCurrentDirectory.
- Primer:
  - `Directory.CreateDirectory(@"c:\\temp");`

DirectoryInfo

```
dirName = "c:\\temp";
DirectoryInfo dir = new DirectoryInfo(dirName);
if (!dir.Exists)
   throw new DirectoryNotFoundException
         ("ne postoji"+dirName);
   foreach( DirectoryInfo di in dir.GetDirectories() )
          lstFolders.Items.Add(di.Name);
   foreach( FileInfo fi in dir.GetFiles() )
       lstFiles.Items.Add(fi.Name);
```

Reading, Writing

- *Stream* je apstraktna klasa koja omogućava binarno čitanje i pisanje podataka iz fajlova.

- Stream je apstraktna klasa svih tokova. Jedan tok je apstrakcija sekvence bajtova, datoteke, u/i uređaja, međuprocesne komunikacije "pipe", ili TCP/IP socketa. Stream klasa i izvedene klase omogućuju generički pogled na ulaz i izlaz podataka, nezavisno od vrste toka.

- Neke važne metode ove klase su:
- `Read(), Write(), Close(), Flush().`

Primer: Binarno kopiranje fajlova

```
using System.IO;
static void Main(string[] args)  {
      int buffSize = 4096;
      Stream inStream = File.OpenRead(@"c:\temp\default.asp");
      Stream outStream = File.Open(@"d:\temp\default.asp",FileMode.OpenOrCreate);

      Byte[] buffer = new Byte[buffSize] ;  // buffer size
      int numBytesRead;
      while ((numBytesRead = inStream.Read(buffer,0,buffSize)) > 0) {
            outStream.Write(buffer,0,numBytesRead);
      }

      inStream.Close();
      outStream.Close();
}
```

Baferisani tokovi

- Pošto je pristup disku vremenski dug u odnosu na druge računarske operacije, operativni sistem čita i piše podatke u blokovima. Tako, kada imamo veću količinu podataka u malim delovima kreira se baferisani tok koji se koristi privremeno dok se podaci se sakupe da bi na kraju bili upisani na disk. Time se dobija na brzini rada.

- Kada koristite baferisani tok uvek uradite snimanje na disk **flush** pre zatvaranja toka inače će podaci biti izgubljeni.

Primer:

```
static void Main(string[] args)   {
    int buffSize = 4096;
    Stream inStream = File.OpenRead(@"c:\temp\default.asp");
    Stream outStream =
                File.Open(@"d:\temp\default.asp",FileMode.OpenOrCreate);
    BufferedStream inbuffStream = new BufferedStream(inStream);
    BufferedStream outbuffStream = new BufferedStream(outStream);

    Byte[] buffer = new Byte[buffSize] ;  // buffer size
    int numBytesRead;
    while ((numBytesRead = inbuffStream.Read(buffer,0,buffSize)) > 0)   {
          outbuffStream.Write(buffer,0,numBytesRead);
    }
    outbuffStream.Flush();
    inbuffStream.Close();
    outbuffStream.Close();
}
```

FileStream

- Podržava binarno čitanje i pisanje u fajl. Moguće je i pristupanje bilo kojoj poziciji u fajlu (random access capability for reading/writing) i podržava asinhrono čitanje i pisanje podataka bez blokiranja. Asinhrono čitanje i pisanje daje i notifikaciju kada je operacija završena.

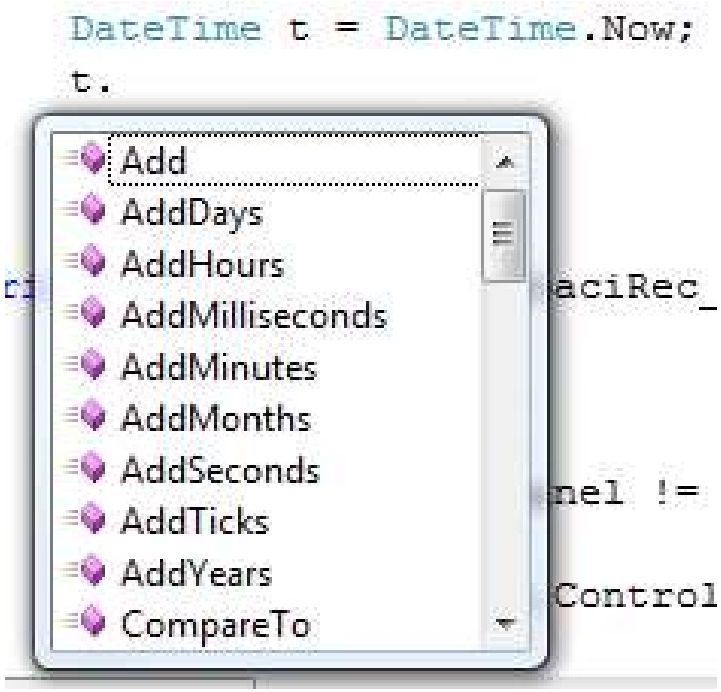
```
FileStream fsw = new FileStream(@"d:\temp\t1.txt",
                                FileMode.Create, FileAccess.Write);
byte b1 = 65;
fsw.Write(b1);

byte [] bArr = new Byte[100];
fsw.Write(bArr,0,50);

FileStream fsr = new FileStream(@"d:\temp\t1.txt",
                                FileMode.Open, FileAccess.Read);
fsr.Read(bArr,0,50);
fsr.Close();
fsw.Close();
```

Rad sa tekstualnim fajlovima

```
FileInfo fr = new FileInfo(@"c:\temp\t1.txt");
StreamReader reader = fr.OpenText();
string sline; string [] substrings;
char [] separators = {' ', ','.'\t'};
string name; int ID; float gpa;

StreamWriter writer = new StreamWriter(@"c:\temp\t2.txt",false);
sline = reader.ReadLine();

while (sline != null)
{

     substrings = sline.Split(separators,3); // 3 maximum size to split
     name = substrings[0];
     ID = int.Parse(substrings[1]);
     gpa = float.Parse(substrings[2]);
     Console.WriteLine("{0} {1} {2}", name,ID,gpa);
     writer.WriteLine("{0} {1} {2}",name, ID, gpa);
     sline = reader.ReadLine();

}

writer.Flush();
writer.Close();
```

DateTime

- Služi za merenje apsolutnog vremena.
- DateTime je struktura
  - DateTime t = DateTime.Now;



Parse, Globalization

- DateTime ima metode koje konvertuju string u **DateTime** object i obrnuto.
- ***Parse*** :

```
string MyString = "Jan 1, 2002";
DateTime MyDateTime = DateTime.Parse(MyString);
Console.WriteLine(MyDateTime);
```

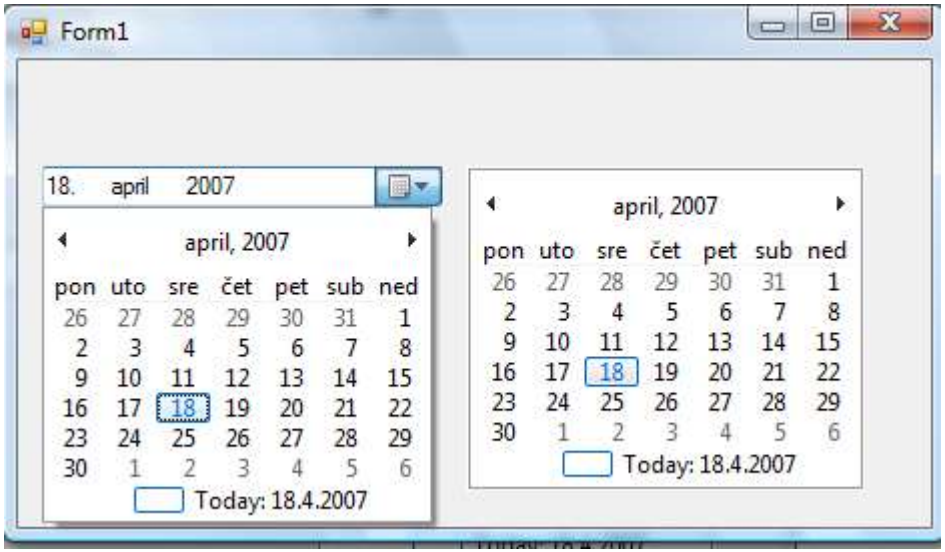
- ***ToString*** metode za formatrinje datuma.
  - `DateTime MyDate = new DateTime(2000, 1, 1, 0, 0, 0);`
  - `String MyString = MyDate.ToString("dddd - d - MMMM");`
  - `MyString = MyDate.ToString("yyyy gg");`
- Formati za ispis datuma/vremena>
  - “d”,”D” – kratki,dugi zapis datuma
  - “t”,“T” – kratki,dugi zapis vremena

Primer upotrebe parametara za globalizaciju

```
static void Main(string[] args)
{
    CultureInfo MyCultureInfo = new CultureInfo("de-DE");
    string MyString = "12 Juni 2002";
    DateTime MyDateTime = DateTime.Parse(MyString, MyCultureInfo);
    Console.WriteLine(MyDateTime);

    CultureInfo MyCultureInfo2 = new CultureInfo("en-US");
    string[] formati = MyDateTime.GetDateTimeFormats();
    Console.WriteLine(MyDateTime.ToString("d", MyCultureInfo2));
}
```

Windows kontrole za rad sa datumom i vremenom

1. `DateTimePicker` – svojstvo `value`



TimeSpan

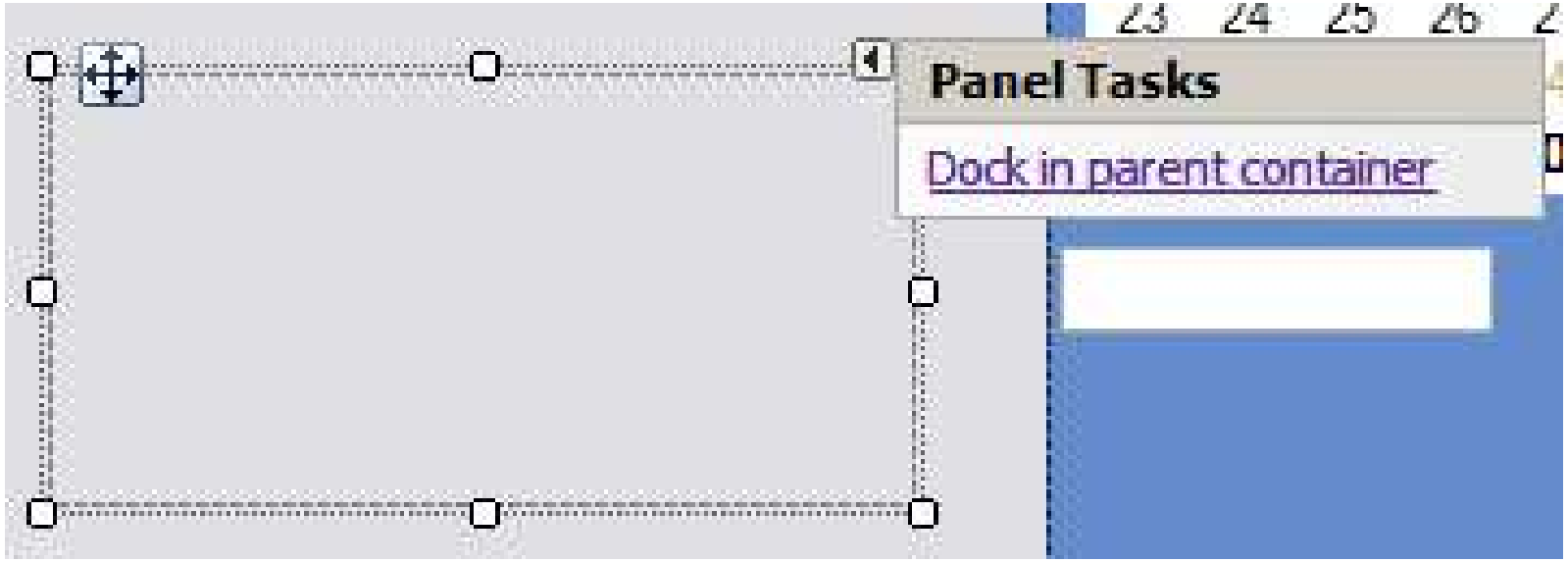
- TimeSpan predstavlja vremenski interval ili trajanje mereno kao pozitivan ili negativan broj dana, sati, sekundi ili delova sekundi. Najkrupnija jedinica je dan. (Broj dana u mesecu i godini varira).
- Vrednost TimeSpan objekta je broj tick-ova koje predstavljaju taj interval. Jedan tik je 100 nanosekundi, a opseg vrednosti TimeSpan objekta je od MinValue do MaxValue.
- TimeSpan vrednost može biti predstavljen i kao [-]*d.hh:mm:ss.ff*, gde je minus opcioni znak koji pokazuje negativni interval. *ff* su delovi sekunde.

- Primer: Na formu postaviti dve kontrole *DateTimePicker* pomoću kojih se zadaju dva datuma. Dodati jedno dugme koje će da obezbedi da se razlika vremena u te dve kontrole prebaci u jedan *ListBox*.

Kontejneri kontrola su…

- …objekti koji mogu čuvati i prikazivati više kontrola.
- Forma je jedan od kontejnera kontrola.
- Kolekcija ***Controls*** je tipična za ovu vrstu kontrola.
- Ova kolekcija nije dostupna preko prozora *Properties* u fazi dizajniranja, već automatski puni tokom dizajna kontrola na formi, ili se koristi direktno u kodu pri programskom kreiranju kontrola.
- Kontejner sadrži sve kontrole u kolekciji *Controls*
- Kontrola prisupa kontejneru preko svojstva *Parent*

Panel



- Inicijalno poprima *boju* forme ili kontejnera kome pripada.
- Svaka kontrola na Panelu ima svoja svojstva ali istovremeno su nadjačana istim svojstvima kontejnerske kontrole. Šta ovo zanači konkretno?

1. Ako je *Visible* svojstvo panela postavljeno na *false* nevidljive su sve kontrole na ovom kontejneru bez obzira na vrednost svojstva Visible kod pojedinačnih kontrola.
2. Ako je *Visible = true*, onda je vidljivost definisana vrednošću ovog svojstva kontrola koje pripadaju Panelu.

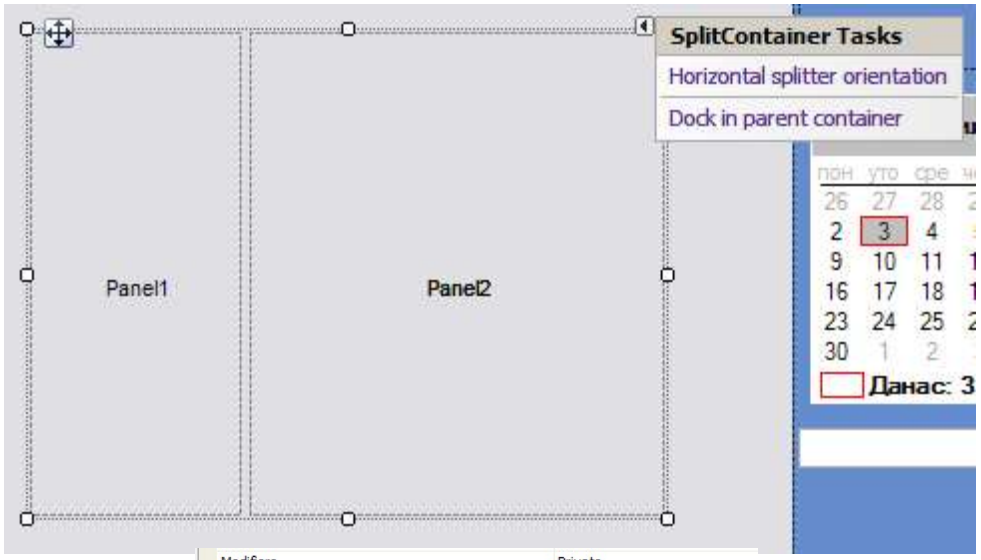
- *Location:* Svaki kontejner definiše svoj koordinatni sistem koji je isto orijentisan, ali sa početnom tačnom (0,0) koja je gornji levi ugao te kontejnerske kontrole.
- Sve kontrole u kontejneru *imaju koordinate definisane u odnosu na kontejner.*
- Promena pozicije kontejnera znači promenu pozicije svih kontrola na istom.
- Kontrola ima svojstvo *Parent* koje pokazuje na kontejner kome ista pripada!

GroupBox

- Sličan Panel kontroli.
- Uobičajeno se koristi pri grupisanju radio dugmadi.
- Sadrži okvir oko kontrola, a može sadržati i tekst.

SplitContainer

- Sadrži dve kontrole Panel
  - Svaki od panela se posebno podešava
- Sadrži jedan Spliter između panela

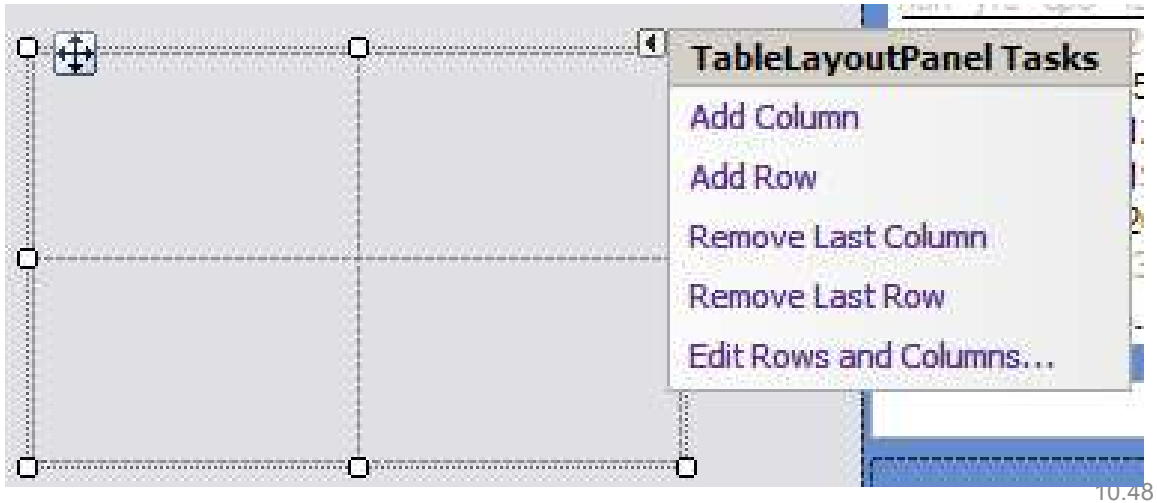


Modifiers	Private
Orientation	Vertical
Panel1	splitContainer1.Panel1
(ApplicationSettings)	
(DataBindings)	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScroll	False
AutoScrollMargin	0; 0
AutoScrollMinSize	0; 0
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayout	Tile
CausesValidation	True
ContextMenuStrip	(none)
Cursor	Default
Enabled	True
Font	Microsoft Sans Serif; 8.25pt

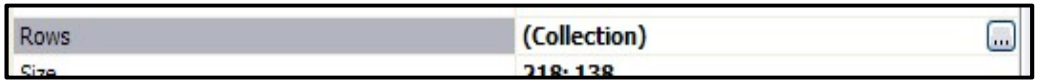
FlowLayoutPanel

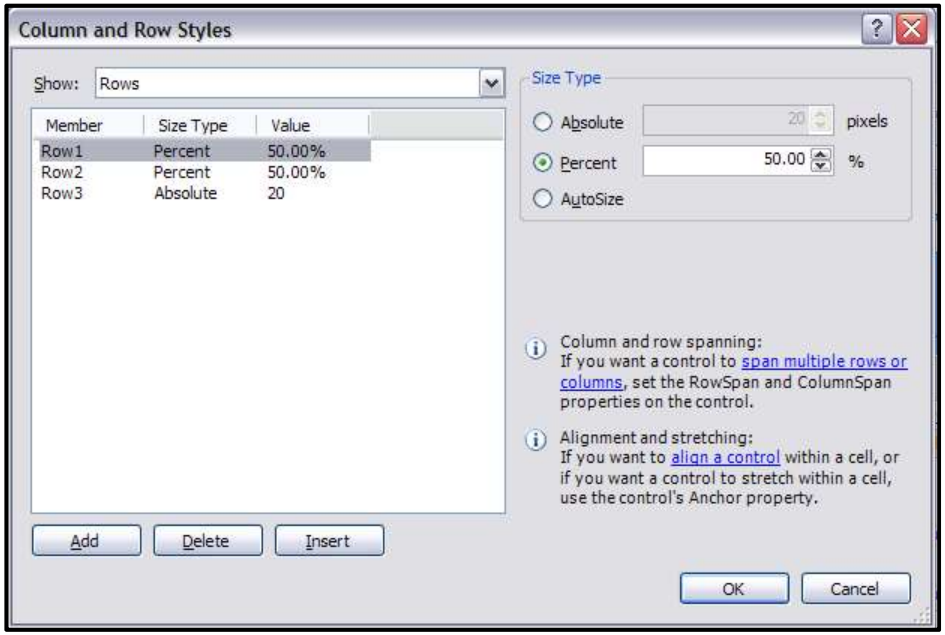
- Ova kontrola uređuje redosled kontrola na
  - Horizontalan ili
  - Vertikalan način.
- **Sadržaj se prelama sa jednog reda u drugi ili iz jedne kontrole u drugu.**
- Njihov sadržaj može se i presecati (ili prelamati) (WrapContents)
- FlowDirection definiše pravac pomeranja kontrola.
- Bilo koja kontrola ili sam **FlowLayoutPanel**, može biti deo sadržaja **FlowLayoutPanela**

TableLayoutPanel

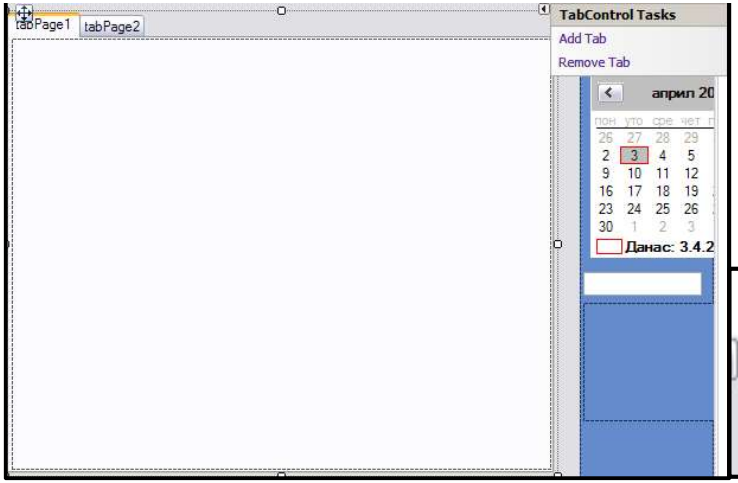


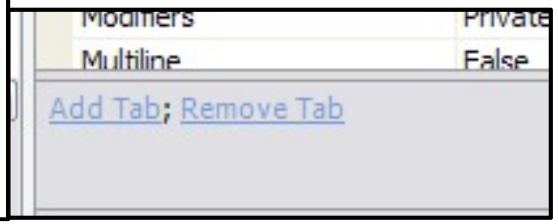
- TableLayoutPanel kontrola uređuje sadržaj na osnovu grida. Daje mogućnost proporcionalnog menjanja veličine ili dužine teksta na osnovu okruženja.
- Bilo koja Windows kontrola, uključujući i sam TableLayoutPanel može biti deo ovo kontrole.
- TableLayoutPanel može da menja svoje dimenzije pomoću svojstava
  - RowCount,
  - ColumnCount,
  - GrowStyle definiše način promene u kontroli kada se dodaje nova kontrola tj. nova ćelija *this.TableLayoutPanel1.Controls.Add(new Button());*





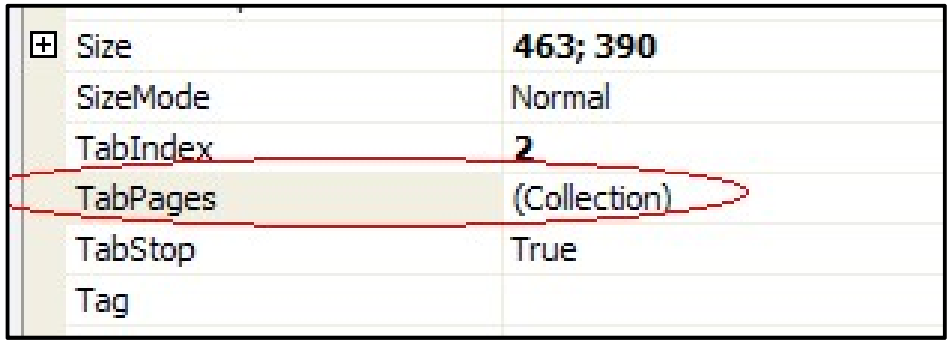
Tab

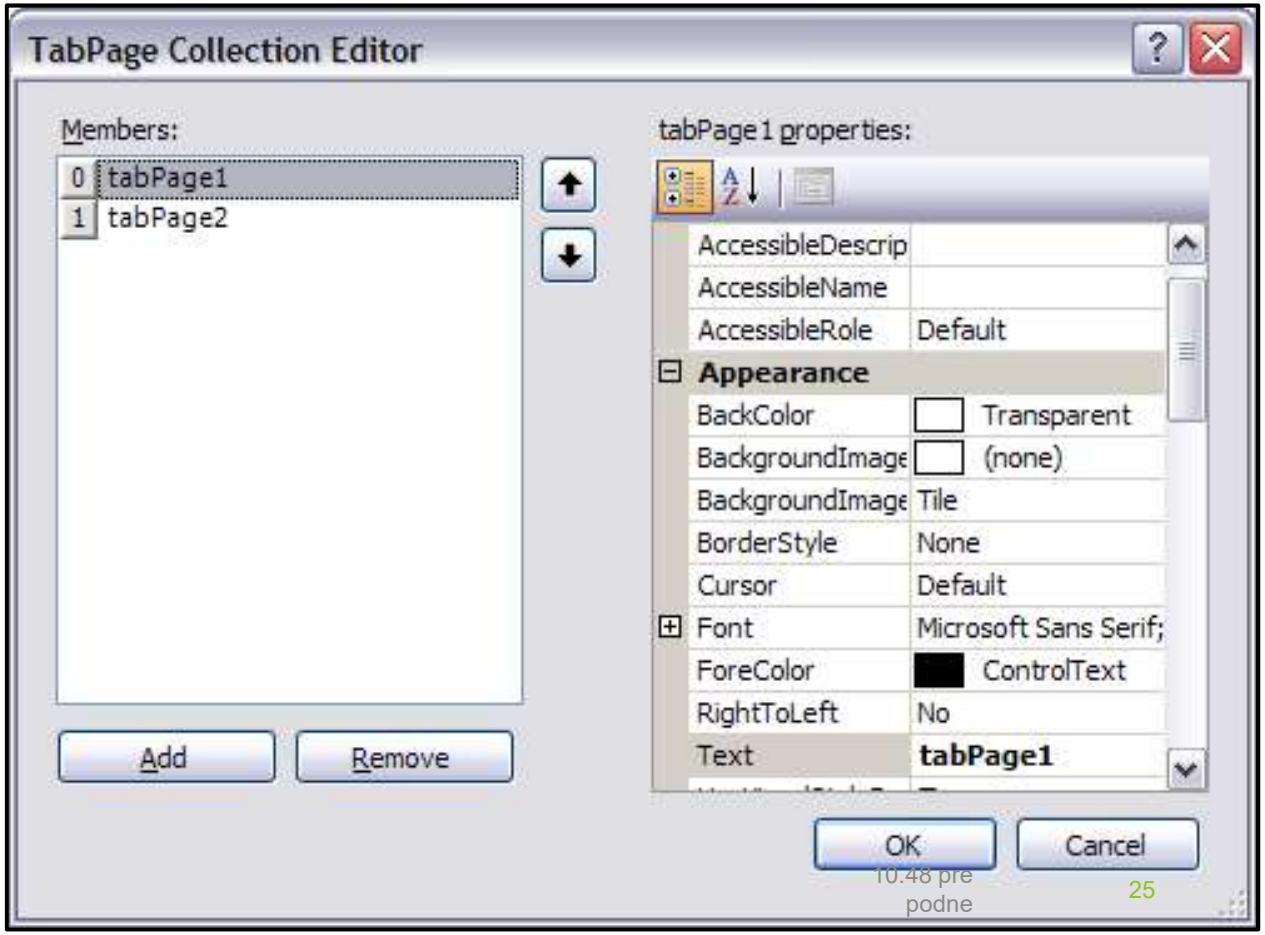




- **`TabControl`** sadrži *tab pages(stranice)*, tipa `TabPage`.
- Dodavanje nove stranice se obavlja pomoću svojstva `TabPages`.
  - Redosled u kolekciji se reflektuje na redosled u kontroli.

- Svojstva, metode i dogadjaji
  - `SelectedIndex`
  - `SelectedTab`
  - `SelectTab`
  - `DeselectTab`
- Događaji
  - `Deselecting`
  - `Deselected`
  - `Selecting`
  - `Selected`


Size
463; 390
SizeMode
Normal
TabIndex
2
TabPages
(Collection)
TabStop
True
Tag


TabPage Collection Editor
Members:
0 tabPage1
1 tabPage2
Add
Remove
tabPage1 properties:
AccessibleDescrip
AccessibleName
AccessibleRole
Default
Appearance
BackColor
Transparent
BackgroundImage
(none)
BackgroundImage
Tile
BorderStyle
None
Cursor
Default
Font
Microsoft Sans Serif;
ForeColor
ControlText
RightToLeft
No
Text
tabPage1
OK
Cancel

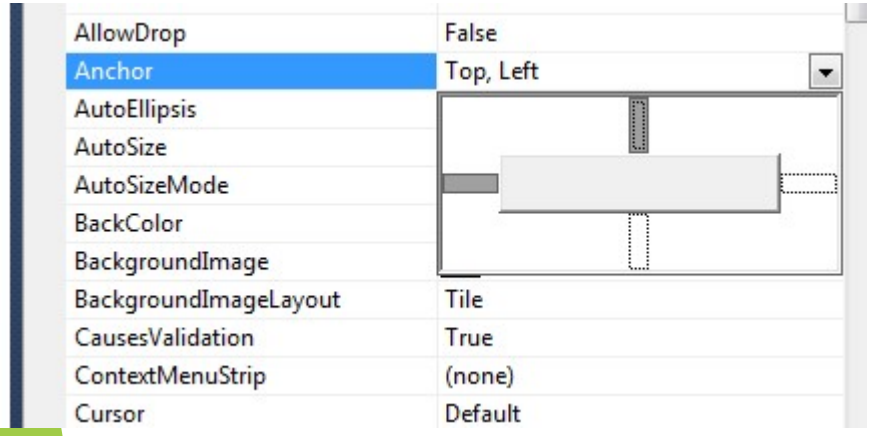
Primeri

- 1. Pokazati kako
  - Dodati kontrolu jednom Panel-u
  - Izbaciti kontrolu sa panela
- 2. Svim labelama na formi (a labele su i na panelima, groupbox-ovima....) promeniti boju slova u plavua

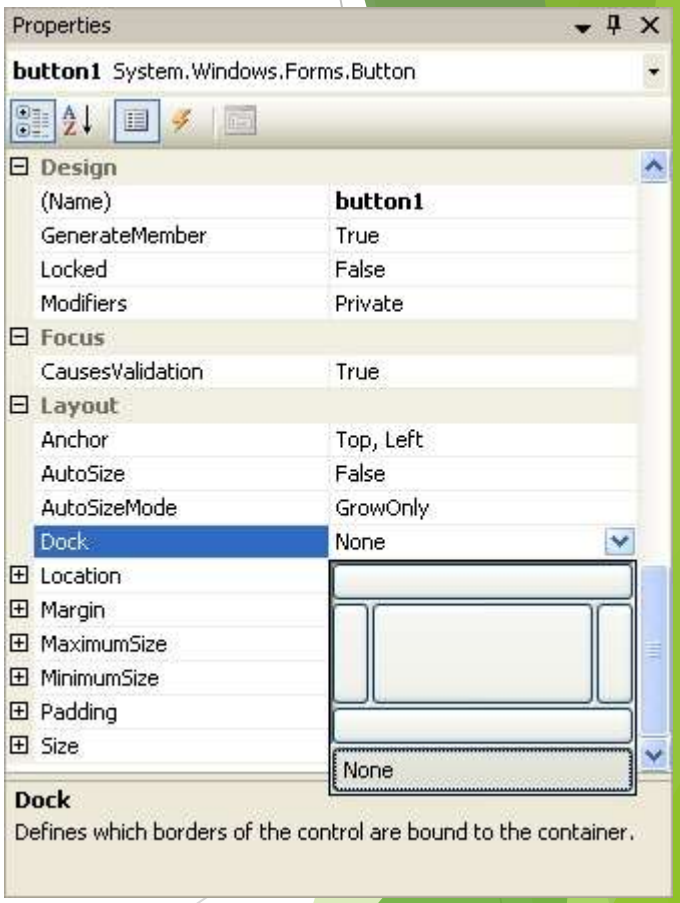
Podešavanje pozicije u kontejnerima

Anchor
"Vezivanje" kontrole se vrši za roditeljsku kontrolu.

- Moguće je vezati kontrolu na više načina:
  - *Bottom* - The control is anchored to the bottom edge of its container.
  - *Left* - The control is anchored to the left edge of its container.
  - *None* - The control is not anchored to any edges of its container.
  - *Right* - The control is anchored to the right edge of its container.
  - *Top* - The control is anchored to the top edge of its container.

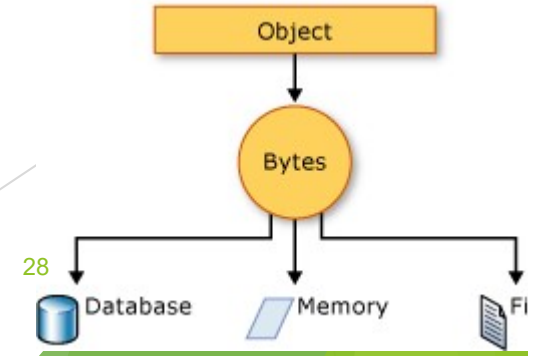


- **Dock**



Serializacija podataka

- Kada kreirate objekat u aplikaciji .Net framevork, se brinuti o tome kako su podaci sačuvani u memoriji. Međutim, ako želite da sačuvate konkretni objekt u fajlu, da ga pošaljete drugom procesu ili da ga prenesete preko mreže, morate razmisliti o tome kako je objekat predstavljen jer ćete morati da konvertujete u drugi format. Ova konverzija se zove SERIJALIZACIJA. Deserializacija je suprotan potupak tj. konvertovanje niza bajtova u objekat.
- Kroz serijalizaciju, programer može izvesti radnje kao što je slanje objekta na udaljenu aplikaciju putem veb-servisa, prosleđivanje objekta iz jednog domena u drugi, prosleđivanje objekta kroz firewall kao XML niz...
- Atribut `Serializable` klase se koristi pri serijalizaciji. Primenjuje se samostalno kada se sva polja klase čuvaju, ali se primenjuje i zajedno sa implementacijom interfejsa `ISerializable` za kontrolu procesa serijalizacije. Dakle, sva javna i privatna polja se serijalizju ako je klasa označena sa `Serializable` atributom. Implementacijom interfejsa `ISerializable` ovaj proces se preklapa implementacijom metoda interfejsa.



- [NonSerialized - Podrazumevani proces serijalizacije isključuje polja koja su označena sa atributom [NonSerialized].
- Ako je polje tipa koji se ne može ili koji ne želite da čuvate onda je smisleno koristiti atribut [NonSerialized] na to polje.
- [OnSerializing] -> Koristi se kada želimo da izvršimo neku akciju dok serijalizujemo objekat
- [OnSerialized] -> Koristi se kada želimo da izvršimo neku akciju nakon serijalizovanja objekta u strea

- Kada je reč o čuvanju podataka koristimo ili binarni zapis, SOAP ili XML zapis podataka i pri tome vršimo serializaciju objekata koristeći klase:
  - **`BinaryFormatter`** (`System.Runtime.Serialization`)
  - **`SoapFormatter`** (`System.Runtime.Serialization`)
  - **`XmlSerializer`** (`System.Xml.Serialization`)
  - `JsonSerializer ..`
- Binarni formater je najefikasniji (ali podaci nisu čitljivi).

```
[Serializable]
class Osoba //: Iserializable
{
    public string ime;
    public string prezime;
    [NonSerialized]
    public string punoIme;
    public Osoba(string i, string p){
       this.ime = i;
       prezime = p;
       punoIme = p.ToUpper() +
                    "," + i.ToUpper();
       }
    //public Osoba(SerializationInfo info,
StreamingContext context){
    // ime = info.GetString("ime");
    // prezime = info.GetString("prezime");
    //}
    //public void
        GetObjectData(SerializationInfo info,
treamingContext context)
    //{
    //    info.AddValue("ime", ime);
    //    info.AddValue("prezime", prezime);
    //}
    public override string ToString(){
            return ime + " " + prezime;
        }
    }

    private void upisiObjekat(object o){
    try{
        fb = new FileStream("studenti.bin",
                            FileMode.Create);

       fs = new FileStream("studenti.xml",
                           FileMode.Create);
        BinaryFormatter binform = new
                           BinaryFormatter();
        SoapFormatter soapform = new
                                 SoapFormatter();

        binform.Serialize(fb, o);
        soapform.Serialize(fs, o);

        fb.Flush(); fs.Flush();
        fb.Close(); fs.Close();
        MessageBox.Show("ok");
    }
    catch (Exception ex){
       MessageBox.Show(ex.Message);

    }
}

    private object procitajObjekat(){
       try{
          fb = new FileStream("studenti.bin",
                              FileMode.Open);
          fs = new FileStream("studenti.xml",
                              FileMode.Open);

          BinaryFormatter binform =
                        new BinaryFormatter();

          SoapFormatter soapform =
                          new SoapFormatter();

            //object o = binform.Deserialize(fb);
            object o = soapform.Deserialize(fs);
            fb.Close(); fs.Close();
            return o;
        }
        catch (Exception ex){
            MessageBox.Show(ex.Message);return null;
        }
    }
```

Zadatak

- Generisati na slučajan način po 5 različitih geometrjskh figura: trougao, kvadrat, krug i pravougaonik
- Krug je definisan centrom i poluprečnikom, a ostali svojim temenima.
- Opseg korišćenih koordinata kao i prečnika je (-20,+20)
- Sačuvati generisanu listu figura u fajlu.
- Pročitati iz fajla pdatke i formirati novu listu.
- Izračunati sumu površina svih figura.