Osnove naprednog programiranja

Svojstva i metode klase System.String za rad s podacima tekstualnog tipa

Proširenje funkcionalnosti upotrebom gotovih klasa .NET biblioteke

Objekti tipa string mogu se stvoriti:

- string ime=„Neki niz znakova.”;
- string putanja=„C:\\MyDoc\\datoteka.txt”; ili
- string putanja =@„C:\MyDoc\datoteka.txt”; //znakovi se tretiraju doslovno

Podatak bilo kojeg tipa pozivom metode ToString daje svoju tekstualnu reprezentaciju, npr. ako je int a=5, nakon a.ToString() dobijemo „5”

Kod Console.WriteLine(„Rezultat: {0}”, rezultat); implicitno se poziva ToString metoda pomoću koje se spaja rezultat u izlazni tekst koji se ispisuje na ekran.

Bitnije metode i svojstva klase String

Metoda ili svojstvo	Svrha
Compare	Statička metoda koja uspoređuje dva niza znakova (poziva se preko imena same klase, npr. string.Compare(s, „...."))
Concat	Statička metoda koja spaja više nizova znakova u jedan
EndsWith	Označava završava li pozivajući niz znakova zadanom znakom
IndexOf	Vraća poziciju prvog pojavljivanja uzorka unutar niza znakovaa
Insert	Umeće navedeni teskt u niz znakova
Length	Daje duljinu stringa, broj znakova
Replace	Unutar pozivajućeg niza znakova zamjenjuje navedeni stari podniz novim
Remove	Briše zadani broj znakova
Split	Razdvaja niz znakova na podnizove prema navedenom znaku razdvajanja (string[] rijeci = recenica.Split(‘ ‘);
StartsWith	Označava počinje li pozivajući niz znakova zadanim znakom
Substring	Dohvaća podniz znakova
ToLower	Pretvara niz znakova u mala slova
ToUpper	Pretvara niz znakova u velika slova
Trim	Uklanja bjeline s početka i završetka niza znakova

Rad s podacima datumskog tipa

Datum i vrijeme

U .NET razvojnoj platformi implementirani kao sistemska struktura DateTime

Strukture posjeduju svojstva i metode no njihove se instance spremaju na stogu, a ne na hrpi.

Dohvaćanje trenutnog datuma i vremena: DateTime dativrij = DateTime.Now;

Sa DateTime.Today dobije se samo datum

Unos ispravno upisanog datuma s tipkovnice u obliku tekstualnog podatka:

- DateTime datum = DateTime.Parse(Console.ReadLine());
- - zbog mogućnosti unosa neispravnog datuma trebalo bi ovo osigurati try-catch blokom

Instanciranje datuma pomoću operatora new: DateTime datum = new DateTime(2011, 7, 4);

Uspoređivanje datuma pomoću relacijskih operatora uspoređivanja (dan prije je manji).

Bitnije metode i svojstva DateTime instance

Metoda ili svojstvo	Svrha
AddYears	Pozivajućoj instanci dodaje zadani broj godina
AddMonths	Pozivajućoj instanci dodaje zadani broj mjeseci
AddDays	Pozivajućoj instanci dodaje zadani broj dana
AddHours	Pozivajućoj instanci dodaje zadani broj sati
AddMinutes	Pozivajućoj instanci dodaje zadani broj minuta
Year	Iz pozivajuće instance vraća godinu
Month	Vraća mjesec
Day	Vraća dan
Hour	Vraća sat
Minute	Vraća minute
Second	Vraća sekunde
DayOfWeek	Vraća dan u tjednu
DayOfYear	Vraća dan u godini
ToString	Vraća datumski zapis prema zadanom formatu
ToLongDateString	Vraća duži oblik zapisa datuma (reg. Postavke)

Primjer

Program traži unos datuma rođenja, a zatim ispisuje starost korisnika.

TimeSpan struktura

Sistemska struktura za opis vremenskih intervala.

Oduzimanjem dva datumska podatka dobije se podatak tipa TimeSpan.

Instanca TimeSpan putem svojih svojstava omogućuje izdvajanje pojedinih vremenskih komponenti.

Metoda ili svojstvo	Svrha
Days	Dohvaća komponentu dana (int).
Hours	Dohvaća komponentu sati (int).
Minutes	Dohvaća komponentu minuta (int).
Seconds	Dohvaća komponentu sekundi (int).
TotalDays	Dohvaća ukupan broj dana (double).
TotalHours	Dohvaća ukupan broj sati (double).
TotalMinutes	Dohvaća ukupan broj minuta (double).
TotalSeconds	Dohvaća ukupan broj sekundi (double).

Vježba:
Ispisati koliko je vremena ostalo do:
a. Nove godine.
b. Vašeg slijedećeg rođendana.

Kolekcije – kolekcija ArrayList

Klasa koja omogućuje rad s nekakvom kolekcijom više elemenata (npr. nizovi, ArrayList, List<T>..)

ArrayList (dio imenskog prostora System.Collections): ArrayList a1=new ArrayList();//count za broj elemenata.

Elementima kolekcije ArrayList možemo pristupiti preko indeksa, prevelik indeks uzrokuje gresku.

Metoda ili svojstvo	Svrha	Primjer
Add	Dodaje novi element u kolekciju.	a1.Add(nekiobjektbilokojegtipa);
Clear	Uklanja sve elemente iz kolekcije.	a1.Clear();
Contains	Utvrđuje pripada li element kolekciji.	a1.Contains(„Marko”);//vraća true ili false
IndexOf	Traži proslijeđeni element u kolekciji i vraća njegov indeks.	a1.IndexOf(„Marko”);//vraća prvu poziciju na kojoj se nalazi ili -1
Insert	Umeće novi element u kolekciju na zadanu poziciju.	a1.Insert(1, objekt);//ubacuje novi element na prvu poziciju
Remove	Uklanja objekt iz kolekcije.	a1.Remove(„Marko”);
RemoveAt	Uklanja element na zadanoj poziciji iz kolekcije.	a1.RemoveAt(1);
Reverse	Obrće redoslijed elemenata u kolekciji.	
Sort	Sortira kolekciju.	
ToArray	Kopira elemente kolekcije u novi niz zadanog tipa.	

Primjer

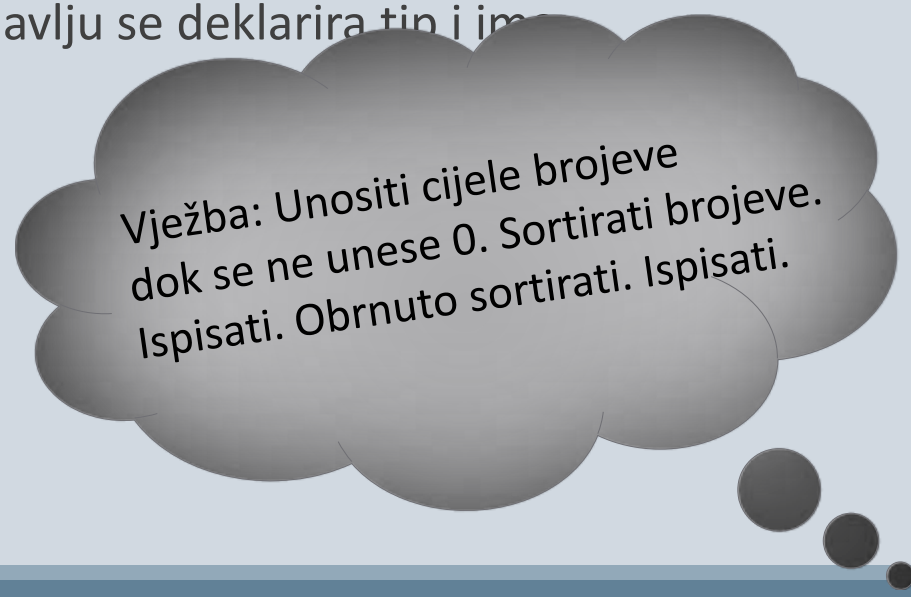
```
ArrayList a1 = new ArrayList();
        a1.Add(3);
        a1.Add("Nikola");
        a1.Insert(1, "RT");
        Console.WriteLine("Elementi u listi su ");
        for(int i=0;i<a1.Count;i++)
           Console.WriteLine(a1[i]);
        if (a1.Contains(3))
           a1.Remove(3);
        int pozicija = a1.IndexOf("RT");
           a1.RemoveAt(pozicija);
        Console.WriteLine("Elementi u listi su ");
        for (int i = 0; i < a1.Count; i++)
           Console.WriteLine(a1[i]);
        Console.ReadKey();
```

Kolekcije – klasa List<T> i petlja foreach

Klasa kolekcija koja posjeduje iste funkcionalnosti kao i klasa ArrayList, uz zahtjev da njezini elementi moraju biti točno određenog tipa. Spada u skupinu generičkih kolekcija jer pripada imenskom prostoru System.Collections.Generic.

List <T> klasa pri instanciranju novog objekta traži da se odabere tip elemenata nove kolekcije:
List<string> imeListe = new List <string>();

foreach petlja ciklički prolazi kroz sve elemente kolekcije. U zaglavlju se deklarira tip i im
foreach(string s in imListe) Console.WriteLine(s);



Upotreba klasa prostora System.IO za rad sa datotekama i datotečnim sustavom

Aplikacije pišu i čitaju podatke iz tekstualnih datoteka.

Za rad sa mapama postoje klase: **Directory** i **DirectoryInfo**

Za operacije manipuliranja datotekama postoje klase: **File** i **FileInfo**

Za manipulaciju putanjama koristi se klasa: **Path**

Pri čitanju i pisanju tekstualnih podataka u kombinaciji s objektom klase **FileStream** koriste se objekti klase **StreamReader** i **StreamWriter**

Klasa Directory

Pruža brojne statičke metode za rad sa mapama

Zadatak:

Napraviti program koji će za zadanu putanju ispisati sve datoteke u toj mapi:

if (Directory.Exists(putanja))

- foreach(string datoteka in Directory.GetFiles(putanja))
- ....

Provjeriti da li neka mapa u drugoj postoji, pa ju kreirati ako ne postoji.

Ispisati sve mape sadržane u nekoj mapi.

Obrisati novostvorenu mapu, pa ponovo ispisati sadržaj roditeljske mape.

Metoda ili svojstvo	Svrha
CreateDirectory	Kreira mapu na zadanoj putanji
Delete	Briše zadanu mapu
Exists	Ispituje postoji li zadana mapa
GetCurrentDirectory	Dohvaća trenutnu radnu mapu aplikacije
GetDirectories	Dohvaća sve podmape u zadanoj mapi (punu putanju)
GetFiles	Dohvaća sve datoteke u zadanoj mapi (punu putanju)
GetParent	Dohvaća mapu roditelj zadane mape
Move	Premješta mapu i njezin sadržaj

Klasa File

Pruža statičke metode za rad sa datotekama.

Zadatak:

Unijeti ime datoteke. Provjeriti da li datoteka s tim imenom i ekstenzijom .txt postoji. Ako postoji kopirati ju u mapu \backup (stvoriti ju ako ne postoji). Dodati imenu podatak o datumu i vremenu kreiranja. Obrisati prvu datoteku.

Metoda ili svojstvo	Svrha
Copy	Kopira zadanu datoteku na novu lokaciju
Delete	Briše zadanu datoteku
Exists	Ispituje postoji li zadana datoteka
Move	Premješta zadanu datoteku na novu lokaciju
WriteAllText	Zapisuje proslijeđeni tekst u zadanu datoteku

The following code example demonstrates the use of the WriteAllText method to write text to a file.

```
using System;/* www .j  av a2s .  c o  m*/
using System.IO;
using System.Text;

class Test
{
    public static void Main()
    {
        string path = @"c:\temp\MyTest.txt";

        string createText = "Hello and java2s.com" + Environment.NewLine;
        File.WriteAllText(path, createText);

        string readText = File.ReadAllText(path);
        Console.WriteLine(readText);
    }
}
```

Klasa Path

Manipulacije putanjama možemo obavljati pomoću statičkih metoda klase Path

Zadatak:

Program traži unos putanja izvorne i ciljne mape i nakon toga kopira sve datoteke. Koristiti try-catch blok.

FileCopy ima treći parametar koji određuje da li će postojeća datoteka u ciljnoj mapi (s istim imenom) biti prepisana.

Metoda ili svojstvo	Svrha
Combine	Iz niza stringova iskombinira putanju
GetDirectoryName	Iz putanje dohvaća ime mape
GetExtension	Iz putanje dohvaća ekstenziju
GetFileName	Dohvaća ime datoteke
GetFileNameWithoutExtension	Dohvaća ime datoteke bez ekstenzije

Klase DirectoryInfo i FileInfo

Sve funkcionalnosti koje u radu s datotekama i mapama pružaju statičke metode implementirane u klasama Directory, File i Path, omogućuju i klase DirectoryInfo i FileInfo kroz svoje instance.

DirectoryInfo di=new DirectoryInfo();

FileInfo fi=new FileInfo();

Zadatak:

Unijeti putanju mape. Instancirati DirectoryInfo objekt. Ispisati sve podmape te mape sa ispisom imena, vremena stvaranja i vremena posljednjeg pristupa. Na isti način ispisati i sve datoteke u toj mapi.

Spremanje teksta u datoteku

Može se spremiti na više načina (npr. preko klasa File i FileInfo).

Najjednostavnije je upotrebljavati objekt klase StreamWriter; instanciranjem StreamWriter objekta otvaramo specificiranu datoteku i ona je spremna za pisanje:

StreamWriter sw = new StreamWriter(„datoteka.txt”);

Metode Write i WriteLine podatke u datoteku upisuju na isti način kao što klasa Console ispisuje podatke na ekran.

sw.WriteLine(„Ovo je redak teksta u datoteci.”);

Otpuštanje resursa

Datotečni sustav je dijeljeni resurs kojim se koriste i drugi programi na računalu, tako da je upotrebljene resurse (datoteke) potrebno na ispravan način što prije otpuštati: datoteku ćemo otvoriti u trenutku kad nam je zaista nužna, a čim nam više nije potrebna, zatvorit ćemo ju kako bi bila dostupna i drugim korisnicima.

StreamWriter objekt pruža dvije metode koje zatvaraju datoteku i otpuštaju sve upotrijebljene resurse: Close i Dispose. Nakon upisa sadržaja u datoteku, bilo koja od njih će ispravno zatvoriti datoteku.

```
sw.Close();
```

Čitanje sadržaja tekstualne datoteke

Postoji više načina, ali najjednostavnije je pomoću objekta StreamReader. Instanciranjem StreamReader objekta datoteka se otvara za čitanje:

StreamReader sr = new StreamReader(„datoteka.txt”);

Za čitanje teksta iz tekstualne datoteke pomoću StreamReader objekta najkorisnije su metode ReadToEnd (čita sav tekst odjednom) i ReadLine (čita liniju po liniju, a pomoću svojstva EndOfStream provjeravamo da li smo došli do kraja datoteke).

string savTekstDatoteke = sr.ReadToEnd();

while(!sr.EndOfStream) Console.WriteLine(sr.ReadLine());

Otpuštanje resursa izvodi se čim je čitanje datoteke gotovo, a datoteku se zatvara pozivom Close ili Dispose metode StreamReader objekta.

sr.Close();

Otpuštanje objekata pomoću naredbe using

Kako bi se programerima olakšalo otpuštanje objekata koji troše računalne resurse, implementirana je naredba using.

Svaki objekt koji implementira metodu Dispose moguće je zatvoriti u tzv. using blok koji će tu metodu na kraju bloka implicitno pozvati.

Općenita upotreba using bloka je:

```
using (Klasa objekt = new Klasa());

{//neke naredbe...

}
```

Zadatak: Unijeti ime i prezime pa ih upisati u datoteku. Pomoću using bloka automatski otpustiti korištenu datoteku. Nakon toga podatke učitati i ispisati na ekran (također sa using blokom).

Izrada korisničkog sučelja

Pomoću Windows Forms kontrola (skupa klasa imenskog prostora System.Windows.Foms u .NET okolini koje omogućuju brzi razvoj programa s grafičkim korisničkim sučeljem GUI)

Proces programiranja dodatno je olakšan dizajnerskom podrškom.

Dvije faze u razvoju programa: vrijeme dizajniranja (vizualno oblikovanje korisničkog sučelja pomoću Windows Forms Designer, razmještanje kontrola na ekranu) i vrijeme izvršavanja (pisanje programskog koda koje se izvršava nakon pokretanja aplikacije).

Kontrola (za prikaz teksta, gumb...) je klasa koja ima svoj izgled i učahuruje određenu jedinicu funkcionalnosti.

Nakon dizajniranja piše se programski kod koji se povezuje sa pojedinim kontrolama i izvršava kad korisnik dođe u interakciju s njima.

Izrada forme i implementacija događaja

Kreira se Windows Forms Application

Okvir Properties (za konfiguriranje svojstava kontrola), daje informacije o tipu označenog objekta i objašnjava čemu pojedino svojstvo služi.

Okvir Toolbox daje na raspolaganje standardne kontrole svrstane po grupama. Kontrole se biraju i iznose dvostrukim klikom ili sa drag&drop.

Primjer: staviti gumb i promijeniti mu svojstvo Text.

Pozadinsko kodiranje: ispis teksta kod pritiska na gumb, automatski se kreira metoda button1_Click.. MessageBox.Show(„Hello”);

Brisanje metode za obradu događaja – uklanja se u okviru Properties pod Events, zatim u kodnom editoru možemo obrisati tijelo metode.

Parcijalna klasa

Forma je sastavljena od tri dijela (datoteke).

Form1.cs sadrži programski kod forme koji piše programer.

Form1.Designer.cs sadrži kod kojeg automatski generira dizajner (ne mijenjati ručno).

Form1.resx čuva sve dodatne resurse forme koje sami dodajemo, a koji nisu programski kod (pozadinske slike, ikone...)

Pokretanjem programa ova se tri dijela spajaju u funkcionalnu cjelinu.

Svaka od ovih datoteka deklarirana je kao parcijalna klasa.

Nasljeđivanje iz bazne klase Form

Sve forme svoja osnovna svojstva i mogućnosti nasljeđuju iz bazne klase Form, članice imenskog prostora System.Windows.Forms u kojoj su deklarirani izgled forme, rubovi, mogućnost promjene veličine i dr.

: određuje nasljeđivanje iz klase koja ju slijedi

Svojstva forme

Za prilagođavanje izgleda i ponašanja: Name (ime forme, odnosno klase koja nasljeđuje klasu Form), BackColor, ControlBox (da li ima kontrolni set gumba), FormBorderStyle (izgled i ponašanje okvira i naslovne trake), Location, MaximizeBox, ShowInTaskbar, Size, StartPosition, Text, Visible, WindowState...

Izmjena svojstava tijekom izvršavanja aplikacije upotrebom ključne riječi this. Ključna riječ this upotrebljena u bilo kojoj metodi definiranoj unutar forme predstavlja cijelu formu:
this.Text=„Novi naslov”;

Vrijednosti svojstava koja nisu jednostavnog tipa su instance klase ili strukture. Vrijednosti takvim svojstvima postavljamo tako da im dodijelim o postojeću instancu ili kreirano novu.

this.BackColor=Color.Yellow; // iz imenskog prostora System.Drawing

this. Location=new Point(100,100);//nova instanca strukture Point

this.WindowState=FormWindowState.Normal;

Veličina, stil okvira i lokacija forme

Veličina se određuje svojstvom Size (ako forma nije maksimizirana ili minimizirana) koje je instanca istoimene strukture Size (Width i Height), a može se postaviti u okviru Properties, podesiti ručno ili promijeniti kroz kod this.Width=400 ili samo Width=400, ili this.Size= new Size(400, 500).

Svojstvo FormBorderStyle određuje kako će izgledati i kako se ponašati okvir forme za vrijeme izvršavanja aplikacije. Predodređena vrijednost je Sizable (moguće je ručno mijenjati veličinu okvira forme). FixedSingle i FixedDialog ne dozvoljavaju podešavanje veličine forme. None uklanja okvire forme i njezinu naslovnu traku.

Početna lokacija forme određena je kombinacijom StartPosition i Location. Svojstvo Location određuje poziciju gornjeg lijevog kuta forme.

Primjer upotrebe svojstava forme

1. stvoriti novu WFA

2. kroz dizajner postaviti inicijalna svojstva forme: žutozelena, moguće mijenjati veličinu, 500x250, na centru ekrana, naslov

3. dodati šest button kontrola: button1-promijeni boju pozadine, button2-postavi na 0;0, button3-povećaj širinu, button4-povećaj visinu, button5-smanji širinu, button6-smanji visinu

4. promjena boje: Random rnd=new Random(); BackColor=Color.FromArgb(rnd.Next(0,255), rnd.Next(0,255), rnd.Next(0,255));//objekt rnd klase Random omogućuje generiranje slučajno odabranog cijelog broja iz zadanog intervala.

5. promjena lokacije: Location=new Point(0,0);//pozicioniranje forme u lijevi gornji kut ekrana

6. promjena veličine: +-10 pixela

Svojstvo Controls

Forma svoje osnovne značajke nasljeđuje iz bazne klase Form.

Svaka Windows Forms kontrola osnovu svoje funkcionalnosti izvodi iz bazne klase Control (imenski prostor System.Windows.Forms). Zato se svaku kontrolu dodanu na formu tretira kao klasu Control.

Na tome se temelji svojstvo Controls tipa ControlCollection (kolekcija koja sadrži elemente tipa Control). Sve kontrole dodane na formu sadržane su u kolekciji Controls.

Kolekcija Controls omogućuje dodavanje elemenata (Add) i uklanjanje postojećih (Remove, RemoveAt), dohvaćanje preko indeksa, iteriranje u foreach petlji itd te time i rad s kontrolama kroz kod.

Primjer upotrebe svojstava Controls

1. novi WFA projekt sa jednom button kontrolom (Text: ispiši sve kontrole na formi)

2. dvoklik na površinu forme i zatim u FormLoad: Button btn1=new Button(); btn1.Text=„Gumb 1”; btn1.Location=new Point(20,50); this.Controls.Add(btn1); Button btn2=new Button(); btn2.Text=„Gumb 2”; btn2.Location=new Point(120,50); this.Controls.Add(btn2);

3. u dizajneru na click metodu gumba na formi dodati: string ispis=„”;foreach(Control ctl in this.Controls) ispis+=ctl.GetType().ToString()+”: „+ctl.Text+”\n”; MessageBox.Show(ispis);

Upotreba jednostavnih kontrola

Nakon konfiguracije osnovnih svojstava, za oblikovanje sučelja na formu se dodaju različite kontrole.

Kako su sve kontrole izvedene iz bazne klase Control, nasljeđuju mnoga zajednička svojstva.

Zajednička svojstva kontrola

Svojstvo	Opis
Anchor	Određuje kako je kontrola pričvršćena za rubove forme.
BackColor	Pozadinska boja kontrole.
Dock	Određuje kako je kontrola usidrena na formi.
Enabled	Dostupnost kontrole.
Font	Font kojim se prikazuje tekst u kontroli.
ForeColor	Boja prikaza teksta u kontroli.
Height	Visina kontrole.
Location	Označava poziciju gornjeg lijevog kuta kontrole u odnosu na gornji lijevi kut forme.
Name	Ime kontrole preko kojeg ju dohvaćamo u kodu.
Parent	Forma u čijoj se Controls kolekciji kontrola nalazi.
Size	Veličina kontrole.
TabOrder	Označava redni broj kontrole vezano za selektiranje pomoću tipke TAB.
Tag	Omogućuje spremanje vrijednosti povezane s kontrolom.
Text	Tekst kontrole.
Visible	Određuje da li je kontrola vidljiva ili ne.
Width	Širina kontrole

Svojstva Anchor i Dock, Font, odabir boje

Nalažu kontroli kako će se ponašati unutar forme.

Anchor omogućuje pričvršćivanje kontrole za jedan ili više rubova forme u kojoj je kontrola sadržana – kontrola je uvijek jednako udaljena od rubova forme za koje je pručvršćena.

Svojstvo Dock omogućuje priljubljivanje kontrole uz rub forme.

Primjer upotrebe zajedničkih svojstava kontrola (118).

Button, Label i TextBox kontrole

Kontrole za interakciju između korisnika i aplikacije s grafičkim korisničkim sučeljem.

Label kontrola koristi se za prikaz tekstualnih informacija na formi (npr. natpis uz kontrolu Textbox)

TextBox kontrola služi za prihvaćanje tekstualnog unosa korisnika. Omogućuje prikaz i izmjenu, odnosno unos teksta.

Svojstvo	Opis
Lines	Vraća niz stringova koji predstavljaju linije u tekstu kontrole.
PasswordChar	Maskirni znak koji se prikazuje umjesto stvarnih znakova.
ReadOnly	Određuje može li korisnik mijenjati tekst u kontroli.
ScrollBars	Određuje kako će se prikazivati trake za pomicanje.
Text	Dohvaća ili postavlja tekst.
WordWrap	Određuje hoće li se riječi automatski prelamati iz jedne linije teksta u drugu.

Button kontrola

Omogućuje zapovjednu interakciju između korisnika i grafičkog sučelja aplikacije.

Primjer upotrebe Label, TextBox i Button kontrola (120).

RadioButton, CheckBox, GroupBox i Panel kontrole

RadioButton i CheckBox spadaju u skupinu kontrola koje omogućuju odabir vrijednosti iz ponuđene grupe opcija. Iz grupe RadioButton kontrola na formi može se odabrati samo jedna od ponuđenih opcija, a svaku opciju označava kružić. Iz grupe CheckBox kontrola moguć je odabir više opcija.

Najbitnija svojstva su Text i Checked.

RadioButton kontrole obično se grupiraju na posebnu GroupBox kontrolu ili Panel kontrolu.

GroupBox kontrola ima svoj okvir i natpis i služi kao kontejnerska kontrola za grupiranje drugih kontrola, tipično RadioButton kontrola. Ako ne želimo posebno isticati izdvojenost grupe, tada možemo upotrebljavati Panel kontrolu koja ne mora imati vidljivi natpis i okvir, a može sadržavati i grupu RadioButton kontrola.

Primjer upotrebe RadioButton i CheckBox kontrola (122).

List kontrola ListBox

Za prikaz popisa (liste) podataka iz koje korisnik odabire stavke. Stavke se čuvaju u kolekciji Items i moguć je odabir jedne ili više stavaka.

Svojstvo	Opis
FormatString	Određuje niz znakova za oblikovanje stavki.
FormattingEnabled	Omogućuje korisničko oblikovanje stavki.
Items	Dohvaća kolekciju stavki.
SelectedIndex	Indeks prve označene stavke, inače -1.
SelectedIndices	Vraća kolekciju svih označenih indeksa.
SelectedItem	Prva označena stavka.
SelectedItems	Vraća kolekciju svih označenih stavki.
SelectionMode	Određuje koliko se stavki može označiti (None, Single, MultiExtended....)
Sorted	Sortira stavke sadržane u kontroli.

List kontrola ComboBox

Za prikaz popisa (liste) opcija iz koje korisnik odabire stavke. Stavke se čuvaju u kolekciji Items i moguć je odabir isključivo jedne stavke. Osim odabira korisniku dopušta i unos nove stavke.

Svojstvo	Opis
DropDownHeight	Maksimalna visina za padajući dio kontrole
DropDownStyle	Određuje stil kontrole (Simple, DropDown, DropDownList (ne dopušta unos)...)
DropDownWidth	Širina padajućeg dijela kontrole
FormatString	Određuje niz znakova za oblikovanje stavki
FormattingEnabled	Omogućuje korisničko oblikovanje stavki
Items	Dohvaća kolekciju stavki
SelectedIndex	Indeks označene stavke
SelectedItem	Označena stavka
Sorted	Sortira stavke sadržane u kontroli
Text	Sadržaj upisan u zaglavlju kontrole

Dodavanje i uklanjanje stavki

Za vrijeme dizajniranja (Items, String Collection Editor) ili kroz kod za vrijeme izvršavanja aplikacije (listKontrola.Items.Add(„Stavka A”), listKontrola.Items.Insert(5, ”Stavka B”)).

Uklanjanje stavki može se izvesti pozivom listKontrola.Items.Remove(„Stavka A”) ili listKontrola.Items.RemoveAt(2). Uklanjanje svih stavki odjednom moguće je pozivom metode listKontrola.Items.Clear();

Formatiranje stavki (prikaz valute, prikaz brojeva sa decimalnim zarezom, prikaz datuma) može se definirati za vrijeme dizajniranja preko Format String Dialog.

Primjer upotrebe list kontrola (127).

NumericUpDown kontrola

Može se koristiti za unos umjesto TextBox kontrole kako bi spriječili pogrešan unos numeričkog podatka.

Omogućuje definiranje intervala brojeva iz kojeg korisnik može odabrati vrijednost.

Svojstvo	Opis
Increment	Iznos za koji će se povećavati ili smanjivati vrijednost
Maximum	Maksimalna vrijednost
Minimum	Minimalna vrijednost
Value	Trenutačna vrijednost kontrole

Kontrole za odabir datuma i vremena: DateTimePicker

Za izbjegavanje parsiranja datuma unesenih u TextBox kontrolu.

DateTimePicker kontrola korisniku omogućuje označiti datumski interval.

Svojstvo	Opis
CustomFormat	Prilagođeni format datuma i vremena
Format	Postavlja format za prikaz datuma i vremena
MaxDate	Maksimalna vrijednost koju kontrola može prihvatiti
MinDate	Minimalna vrijednost koju kontrola može prihvatiti
Value	Vrijednost na koju je kontrola trenutačno postavljena

Kontrole za odabir datuma i vremena: MonthCalendar

Za izbjegavanje parsiranja datuma unesenih u TextBox kontrolu.

MonthCalendar kontrola korisniku omogućuje odabir datuma (dana) iz kalendara. Vrijeme se upisuje u tekstualno polje DateTimePicker kontrole. Odabrani datum i vrijeme vraća svojstvo Value. Može se označiti jedan datum ili datumski interval.

Primjer upotrebe kontrola za odabir datuma (131).

Svojstvo	Opis
MaxDate	Maksimalni datum koji se može odabrati u kontroli
MinDate	Minimalni datum koji se može odabrati
MaxSelectionCount	Maksimalni broj dana koji se mogu označiti
SelectionEnd	Završni datum označenog datumskog intervala
SelectionRange	Datumski interval koji je korisnik selektirao
SelectionStart	Početni datum označenog datumskog intervala

PictureBox kontrola

Osnovna kontrola za prikaz slika različitih formata i iz različitih izvora. Slika koju želimo prikazati postavlja se preko svojstva Image i prozora Select Resource (odabire se slika koja je već prije dodana kao resurs dostupan u projektnoj skladišnoj datoteci Resources.resx ili importira nova slika u skladišnu datoteku projekta). Sliku možemo importirati i kao lokalni resurs tako da je dostupna samo toj PictureBox kontroli. Slika dodana kao projektni resurs može se u kodu dohvatiti i postaviti kao vrijednost Image svojstva PictureBox kontrole: pictureBox1.Image=Properties.Resources.Slika2453; Umjesto učitavanja slike iz skladišne datoteke možemo specificirati stazu do datoteke u kojoj je ta slika postavljanjem svojstva pictureBox1.ImageLocation=@"D:\My Documents\NOOP\slika.bmp";

Svojstvo	Opis
ErrorImage	Slika koja će se prikazati ako se ne učita specificirana slika
Image	Slika koja se učitava u kontrolu
ImageLocation	Internetska adresa ili adresa diska odakle će se slika učitati
InitialImage	Slika koja će biti prikazana u kontroli dok se slika učitava
SizeMode	Određuje kako kontrola upravlja smještanjem slike i promjenom veličine.