C#

ODABRANI PROBLEMI

STVARANJE DATUMA I VREMENA IZ STRINGOVA

- Problem:
- Kreirati System.DateTime instancu koja predstavlja vrijeme i datum određen stringom.

- Rješenje:
- Upotrebom Parse ili ParseExact metode klase DateTime.

OBJAŠNJENJE

- Statička metoda DateTime.Parse osigurava fleksibilan mehanizam za kreiranje DateTime instanci iz stringova koji sadrže parcijalne ili informacije koje uključuju greške. Nedostajuće vrijednosti zamjenjuju se sa defaultnim (tekući datum i 12:00:00 a.m.). Ako ne uspije dolazi do System.FormatException iznimke.
- Ukoliko je potrebno parsirati samo stringove koji odgovaraju određenom formatu, koristi se ParseExact. Najjednostavniji oblik te metode zahtjeva tri argumenta: string koji sadrži datum i vrijeme, format string koji određuje strukturu stringa i referencu IFormatProvider koja osigurava specifične informacije kulture (ako je null, koristi se trenutne informacije). U slučaju neodgovarajućeg oblika dolazi do System.FormatException iznimke. Za određivanje formata može se koristiti i standardne i korisničke specifikatore.
- Detaljnije: dokumentacija za System.Globalization.DateTimeFormatInfo klasu.

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

namespace ca2_7
{
    class Program
    {
        static void Main(string[] args)
        {
            string ds1 = "Sep 2005";
            string ds2 = "Monday 5 September 2005 14:15:33";
            string ds3 = "5,9,5";
            string ds4 = "5/9/2005 14:15:33";
            string ds5 = "2:15 PM";

            DateTime dt1 = DateTime.Parse(ds1);
            DateTime dt2 = DateTime.Parse(ds2);
            DateTime dt3 = DateTime.Parse(ds3);
            DateTime dt4 = DateTime.Parse(ds4);
            DateTime dt5 = DateTime.Parse(ds5);

            Console.WriteLine("String: {0} DateTime: {1}", ds1, dt1);
            Console.WriteLine("String: {0} DateTime: {1}", ds2, dt2);
            Console.WriteLine("String: {0} DateTime: {1}", ds3, dt3);
            Console.WriteLine("String: {0} DateTime: {1}", ds4, dt4);
            Console.WriteLine("String: {0} DateTime: {1}", ds5, dt5);

            try
            {
                DateTime dt6 = DateTime.ParseExact("2:13:30 PM", "h:mm:ss 'PM'", null);
                Console.WriteLine(dt6);
            }
            catch (FormatException)
            {
                Console.WriteLine("not in the correct format.");
            }

            try
            {
                DateTime dt7 = DateTime.ParseExact("Mon, 05 Sep 2005 14:13:30 GMT",
                "ddd',' dd MMM yyyy HH':'mm':'ss 'GMT'", null);
            Console.WriteLine(dt7);
            }
            catch (FormatException)
            {
                Console.WriteLine("not in the correct format.");
            }

            try
            {
            DateTime dt8 = DateTime.ParseExact("Sep 05", "MMM dd", null);
            Console.WriteLine(dt8);
            }
            catch (FormatException)
            {
                Console.WriteLine("not in the correct format.");
            }

            Console.WriteLine("\nGlavna metoda zavrsena. Enter za dalje...");
            Console.ReadLine();

        }
    }
}
```

ZBRAJANJE, ODUZIMANJE I USPOREĐIVANJE DATUMA I VREMENA

- Problem:
- Kako izvoditi osnovne aritmetičke operacija i usporedbe datuma i vremena.

- Rješenje:
- Upotrebom DataTime i TimeSpan struktura koje podržavaju standardne aritmetičke operatore i operatore usporedbe.

OBJAŠNJENJE

- Instanca DateTime predstavlja specifično vrijeme, TimeSpan instanca predstavlja vremenski period. Interno, obje instance za predstavljanje vremena koriste tickove (1 tick je 100 ns). TimeSpan interval vremena sprema kao broj tickova u tom intervalu, a DateTime kao broj tickova od 12:00:00 ponoći 1.1.0001 CE (Common Era ili AD u Gregorijanskom kalendaru).
- Podržani operatori su: =, +, -, ==, !=, >, >=, <, <=
- TimeSpan strukture podržavaju i unarni plus i negaciju
- DateTime strukture implementiraju i AddTicks, AddMilliseconds, AddSeconds, AddMinutes, AddHours, AddDays i AddYears metode. Upotrebom navedenih metoda stvara se nova instanca sa izmijenjenom vrijednošću.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ca2_8
{
    class Program
    {
        static void Main(string[] args)
        {
            TimeSpan timespan1 = new TimeSpan(2, 12, 0, 0);
            TimeSpan timespan2 = new TimeSpan(4, 12, 0, 0);
            TimeSpan timespan3 = TimeSpan.FromDays(3.2);
            TimeSpan oneWeek = timespan1 + timespan2;

            DateTime now = DateTime.Now;
            DateTime past = now - oneWeek;
            DateTime future = now + oneWeek;

            Console.WriteLine("now : {0}", now);
            Console.WriteLine("past : {0}", past);
            Console.WriteLine("future : {0}", future);
            Console.WriteLine("now is greater than past: {0}", now > past);
            Console.WriteLine("now is equal to future : {0}", now == future);

            Console.WriteLine("\nMain method complete. Press enter");
            Console.ReadLine();
        }
    }
}
```

SORTIRANJE POLJA ILI KOLEKCIJE

- Problem:
- Sortirati elemente polja ili kolekcije.

- Rješenje:
- Korištenjem statičke metode System.Linq.Enumerable.OrderBy koja sortira generičke kolekcije i polja. Za druge kolekcije može se koristiti Cast metoda za konverziju u generičku kolekciju i zatim koristiti Enemerable.OrderBy. Za ArrayList objekte može se koristiti ArrayList.Sort.

OBJAŠNJENJE

- Polja i klase koje su generičke kolekcije mogu biti sortirani. Za sortiranje može se koristiti neko svojstvo ili metoda.
- Ako se sortira kolekcija MyType instanci upotrebom myProperty svojstva piše se List<MyType> list = new List<MyType>();
- Enumerable.OrderBy(list, x=>x.myProperty);
- Vraćena instanca može se koristiti za enumeraciju sortiranih podataka (npr. u foreach petlji) ili za stvaranje nove sortirane kolekcije pozivom ToArray, ToDictionary ili ToList.
- Negeneričke kolekcije (kreirane bez <type> sintakse) trebaju biti konvertirane u generičke upotrebom Cast<> metode, pri čemu svi članovi kolekcije moraju biti specificiranog tipa ili se koristi Cast<object> za stvaranje kolekcije koja će raditi sa bilo kojim sadržanim tipom.
- ArrayList kolecija je iznimka po tome što se ne može koristiti sa generičkim sintaksom. Za instance ArrayList treba koristiti ArrayList.Sort() metodu.

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ca2_9
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] array = { 4, 2, 9, 3 };

            array = Enumerable.OrderBy(array, e => e).ToArray <int>() ;

            foreach (int i in array) {
                Console.WriteLine(i);
            }

            List<string> list = new List<string>();
            list.Add("Ana");
            list.Add("Kevin");
            list.Add("Alen");
            list.Add("Marin");

            Console.WriteLine("\nList sorted by content");
            foreach (string person in Enumerable.OrderBy(list, e=>e))
                {
                Console.WriteLine(person);
                }

            Console.WriteLine("\nList sorted by property");
            foreach (string person in Enumerable.OrderBy(list, e =>
e.Length))
            {
                Console.WriteLine(person);
            }

            ArrayList arraylist = new ArrayList(4);
            arraylist.Add("Ana");
            arraylist.Add("Kevin");
            arraylist.Add("Alen");
            arraylist.Add("Marin");

            arraylist.Sort();

            Console.WriteLine("\nList sorted by content");
            foreach (string s in list)
            {
                Console.WriteLine(s);
            }
            Console.WriteLine("Press Enter");
            Console.ReadLine();

        }
    }
}
```

KOPIRANJE KOLEKCIJE U POLJE

- Problem:
- Kopiranje sadržaja kolekcije u polje.

- Rješenje:
- Upotrebom ICollection.CopyTo metode koju implementiraju sve klase kolekcija ili upotrebom ToArray metode koju implementiraju ArrayList.Stack i Queue kolekcije.

OBJAŠNJENJE - COPYTO

- Metode ICollection.CopyTo i ToArray izvode slične funkcije kopiranja elemenata kolekcije u polje. Ključna je razlika što CopyTo kopira elemente kolekcije u postojeće polje, dok ToArray kreira novo polje prije nego što u njega kopira elemente kolekcije.
- CopyTo metoda ima dva argumenta, polje i indeks. Polje je cilj operacije kopiranja i mora biti odgovarajućeg tipa da primi elemente kolekcije. Ako tipovi ne odgovaraju ili nije moguća implicitna konverzija, javlja se System.InvalidCastException iznimka. Indeks je početni element polja u koje će se kopirati. Ako je indeks jednak ili veći duljini polja ili broj elemenata prelazi kapacitet polja javlja se iznimka System.ArgumentException.

OBJAŠNJENJE – ARRAYLIST.TOARRAY

- ArrayList, Stack i Queue klase i njihove generičke verzije također implementiraju ToArray metodu koja automatski kreira polje odgovarajuće veličine za sve elemente kolekcije. Ako se ToArray poziva bez argumenata, vraća object[] polje, neovisno o tipu objekata sadržanih u kolekciji.
- ArrayList.ToArray metoda ima preopterećeni oblik kojem se može predati System.Type objekt koji specificira tip polja koji ToArray metoda treba kreirati.
- Arra

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;

namespace ca2_10
{
    class Program
    {
        static void Main(string[] args)
        {
            ArrayList list = new ArrayList(5);
            list.Add("Brenda");
            list.Add("George");
            list.Add("Justin");
            list.Add("Shaun");
            list.Add("Karlo");

            string[] array1 = new string[list.Count];
            list.CopyTo(array1, 0);

            object[] array2 = list.ToArray();

            string[] array3 =
(string[])list.ToArray(typeof(String));

            Console.WriteLine("Array 1:");
            foreach (string s in array1)
                Console.WriteLine("\t{0}", s);

            Console.WriteLine("Array 2:");
            foreach (string s in array2)
                Console.WriteLine("\t{0}", s);

            Console.WriteLine("Array 3:");
            foreach (string s in array3)
                Console.WriteLine("\t{0}", s);

            Console.WriteLine("\nMain method complete.
Press Enter");
            Console.ReadLine();

        }
    }
}
```

ČITANJE KORISNIČKOG ULAZA IZ KONZOLE

- Problem:
- Čitanje korisničkog ulaza s Windows konzole, liniju ili znak.

- Rješenje:
- Korištenjem Read ili ReadLine metode klase System.Console za čitanje ulaza nakon što korisnik pritisne Enter ili bez tog sa Console.ReadKey

OBJAŠNJENJE

- Najjednostavniji način čitanja s konzole je upotrebom statičkih Read ili ReadLine metoda klase Console. Obje metode blokiraju aplikaciju dok se ne obavi ulaz. Obje instance prikazuju ulazne znakove. Read metoda vraća int vrijednost koja predstavlja slijedeći znak ulaznih podataka ili -1 ako više nema podataka. ReadLine metoda vraća string koji sadrži sve unijete podatke ili prazan string.
- .NET uključuje Console.ReadKey metodu koja osigurava čitanja s ulaza bez čekanja na Enter. Čeka korisnika da pritisne tipku i vraća System.ConsoleKeyInfo objekt pozivatelju. Ako se kao argument stavi true u preopterećenom obliku metode, može se spriječiti prikaz ulaznog znaka na ekranu konzole.
- Vraćeni ConsoleKeyInfo objekt u svojstvima klase sadrži detalje o pritisnutoj tipki: Key vraća vrijednost ConsoleKey enumeracije koja predstavlja pritisnutu tipku tipkovnice (znakovi, funkcijske tipke, navigacijske i tipke za editiranje, specijalizirane tipke...), KeyChar vraća vrijednost Unicode znaka, Modifiers vraća bitovnu kombinaciju vrijednosti System.ConsoleModifiers enumeracije koja identificira jedan ili više modifikatora pritisnutih istovremeno (Alt, Control i Shift)
- KeyAvaliable metoda klase Console vraća bool vrijednost koja prikazuje da li je ulaz dostupan u ulaznom bufferu bez da blokira kod.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace primjer_2_15
{
    class Program
    {
        static void Main(string[] args)
        {
            ConsoleKeyInfo key;
            bool secret = false;
            List<char> input = new List<char>();
            string msg = "Unesi znakove i pritisni ESC da vidis ulaz" +
                "\nStisni F1 za ulaz/izlaz tajanstveni mode Alt-x da izdjes";
            Console.WriteLine(msg);
            do
            {
                key = Console.ReadKey(true);
                if (key.Key == ConsoleKey.F1)
                {
                    if (secret)
                    {
                        secret = false;
                    }
                    else
                    {
                        secret = true;
                    }
                }
                if (key.Key == ConsoleKey.Backspace)
                {
                    if (input.Count > 0)
                    {
                        input.RemoveAt(input.Count - 1);
                        Console.Write(key.KeyChar);
                        Console.Write(" ");
                        Console.Write(key.KeyChar);
                    }
                }
                else if (key.Key == ConsoleKey.Escape)
                {
                    Console.Clear();
                    Console.WriteLine("Ulaz: {0}\n\n",
                        new String(input.ToArray()));
                    Console.WriteLine(msg);
                    input.Clear();
                }
                else if (key.Key >= ConsoleKey.A && key.Key <= ConsoleKey.Z)
                {
                    input.Add(key.KeyChar);
                    if (secret)
                    {
                        Console.Write("*");
                    }
                    else
                    {
                        Console.Write(key.KeyChar);
                    }
                }
            } while (key.Key != ConsoleKey.X
            || key.Modifiers != ConsoleModifiers.Alt);
            Console.WriteLine("\n\nGlavna metoda je izvrsena. Stisni Enter");
            Console.ReadLine();
        }
    }
}
```

UPOTREBA VELIKIH CJELOBROJNIH VRIJEDNOSTI

- Problem:
- Rad sa cjelobrojnim vrijednostima koje su veće od veličine podrazumijevanih numeričkih tipova.

- Rješenje:
- Upotrebom System.Numerics.BigInteger klase.

OBJAŠNJENJE

- Numeričke vrijednosti .NET imaju maksimalne i minimalne vrijednosti u ovisnosti o tome koliko memorije je alocirano tim podatkovnim tipom. Klasa System.Numerics.BigInteger nema takva ograničenja pa se može koristiti za izvođenje operacija za vrlo velikim cjelobrojnim vrijednostima.
- Operacije se izvode upotrebom statičkih metoda klase BigInteger koje vraćaju nove instance.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Numerics;

namespace Apress.VisualCSharpRecipes.Chapter02
{
    class Recipe2_16
    {
        static void Main(string[] args)
        {
            // Create a new big integer.
            BigInteger myBigInt = BigInteger.Multiply(Int64.MaxValue, 2);
            // Add another value.
            myBigInt = BigInteger.Add(myBigInt, Int64.MaxValue);
            // Print out the value.
            Console.WriteLine("Big Integer Value: {0}", myBigInt);

            // Wait to continue.
            Console.WriteLine("\n\nMain method complete. Press Enter");
            Console.ReadLine();
        }
    }
}
```

DOHVAĆANJE INFORMACIJA O DATOTECI, MAPI I POGONU

- Problem:
- Dohvatiti informacije o datoteci, mapi i pogonu

- Rješenje:
- Kreiranjem novog System.IO.FileInfo, System.IO.DirectoryInfo ili System.IO.DriveInfo objekta. Potrebno je navesti stazu resursa u konstruktor čime se omogućava pristup informacijama preko svojstava klase.

OBJAŠNJENJE

- Da bi se kreiralo FileInfo, DirectoryInfo ili DriveInfo objekt, potrebno je preko konstruktora predati stazu i kroz ključna svojstva prihvatiti informacije: Exists (File, Dir), Attributes (File, Dir, vrijednosti iz System.IO.FileAttributes enumeracija), CreationTime, LastAccessTime (File, Dir, vraća System.DateTime i LastWriteTime instance), FullName, Name, Extension (File, Dir), IsReadOnly (File), Length (File size u broju bytova), DirectoryName, Directory (File), Parent, Root (Dir, vraća DirectoryInfo objekt), CreateSubdirectory (Dir), GetDirectories (Dir, vraća polje DirectoryInfo objekata), GetFiles (Dir, vraća polje FileInfo objekata), EnumerateFiles (Dir, vraća IEnumerable FileInfo objekata), EnumerateDirectories (Dir, vraća IEnumerable DirectoryInfo objekata), DriveType (Drive, vraća DriveType enumeraciju koja određuje tip), AvaliableFreeSpace (Drive, vraća long), GetDrives (Drive, vraća polje DriveInfo objekata)
- Metoda Refresh radi update svojstava
- Ako se specificira staza koja ne postoji kod većine svojstava dolazi do FileNotFoundException ili DirectoryNotFoundException iznimke

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace Primjer5_1
{
    class Program
    {
        static void Main(string[] args)
        {
            if (args.Length == 0)
            {
                Console.WriteLine("Molimo vas unesite ime datoteke.");
                return;
            }

            FileInfo file = new FileInfo(args[0]);

            Console.WriteLine(„Provjera datoteke: " + file.Name);
            Console.WriteLine(„Datoteka postoji: " + file.Exists.ToString());

            if (file.Exists)
            {
                Console.Write(„Datoteka stvorena:  ");
                Console.WriteLine(file.CreationTime.ToString());
                Console.Write(„Posljednje azuriranje:  ");
                Console.WriteLine(file.LastWriteTime.ToString());
                Console.Write(„Posljednje pristupanje: ");
                Console.WriteLine(file.LastAccessTime.ToString());
                Console.Write(„Velicina datoteke (bajtovi):  ");
                Console.WriteLine(file.Length.ToString());
                Console.Write(„Lista atributa datoteke:  ");
                Console.WriteLine(file.Attributes.ToString());
            }
            Console.WriteLine();

            DirectoryInfo dir = file.Directory;
            Console.WriteLine(„Provjera mape "+dir.Name);
            Console.WriteLine(„U mapi:  " + dir.Parent.Name);
            Console.Write(„Mapa postoji: ");
            Console.WriteLine(dir.Exists.ToString());

            if(dir.Exists)
            {
                Console.Write(„Mapa stvorena: ");
                Console.WriteLine(dir.CreationTime.ToString());
                Console.Write(„Mapa zadnji put azurirana:  ");
                Console.WriteLine(dir.LastWriteTime.ToString());
                 Console.Write(„Posljednji pristup mapi:  ");
                Console.WriteLine(dir.LastAccessTime.ToString());
                Console.Write(„Lista atributa mape:  ");
                Console.WriteLine(dir.Attributes.ToString());
                Console.WriteLine(„Mapa sadrzi: "+ dir.GetFiles().Length.ToString()+„ datoteka");
            }
            Console.WriteLine();

            DriveInfo drv = new DriveInfo(file.FullName);
            Console.Write(„Drive: „);
            Console.WriteLine(drv.Name);

            if (drv.IsReady)
            {
                    Console.Write(„Tip pogona: „);
                    Console.WriteLine(drv.DriveType.ToString());
                    Console.Write(„Format zapisa pogona: „);
                    Console.WriteLine(drv.DriveFormat.ToString());
                    Console.Write(„Slobodan prostor na pogonu: „);
                    Console.WriteLine(drv.AvaliableFreeSpace.ToString());
             }
             Console.WriteLine(Environment.NewLine);
             Console.WriteLine(„Glavna metoda izvrsena. Pritisnite Enter.”);
             Console.ReadLine();
        }
    }
}
```

DOHVAĆANJE INFORMACIJA O INAČICI DATOTEKE

- Problem:
- Dohvaćanje informacija o inačici datoteke (publisher, broj revizija, komentari itd.)

- Rješenje:
- Upotrebom statičke GetVersionInfo metode klase System.Diagnostics.FileVersionInfo

```
using System;
using System.Diagnostics;

namespace Apress.VisualCSharpRecipes.Chapter05
{
    static class Recipe05_05
    {
        static void Main(string[] args)
        {
            if (args.Length == 0)
            {
                Console.WriteLine("Please supply a filename.");
                return;
            }

            FileVersionInfo info = FileVersionInfo.GetVersionInfo(args[0]);

            // Display version information.
            Console.WriteLine("Checking File: " + info.FileName);
            Console.WriteLine("Product Name: " + info.ProductName);
            Console.WriteLine("Product Version: " + info.ProductVersion);
            Console.WriteLine("Company Name: " + info.CompanyName);
            Console.WriteLine("File Version: " + info.FileVersion);
            Console.WriteLine("File Description: " + info.FileDescription);
            Console.WriteLine("Original Filename: " + info.OriginalFilename);
            Console.WriteLine("Legal Copyright: " + info.LegalCopyright);
            Console.WriteLine("InternalName: " + info.InternalName);
            Console.WriteLine("IsDebug: " + info.IsDebug);
            Console.WriteLine("IsPatched: " + info.IsPatched);
            Console.WriteLine("IsPreRelease: " + info.IsPreRelease);
            Console.WriteLine("IsPrivateBuild: " + info.IsPrivateBuild);
            Console.WriteLine("IsSpecialBuild: " + info.IsSpecialBuild);

            // Wait to continue.
            Console.WriteLine(Environment.NewLine);
            Console.WriteLine("Main method complete. Press Enter.");
            Console.ReadLine();
        }
    }
}
```

ČITANJE I PISANJE TEKSTUALNE DATOTEKE

- Problem:
- Pisanje podataka u slijednu tekstualnu datoteku upotrebom ASCII, Unicode ili UTF-8 dekodiranja.

- Rješenje:
- Kreiranjem novog System.IO.FileStream objekta kao reference na datoteku. Za pisanje koristi se FileStream u System.IO.StreamWriter i preopterećena Write metoda. Za čitanje koristi se File Stream u System.IO.StreamReader i Read ili ReadLine metoda. Read čita jedan znak ili specificirani niz znakova i vraća znak ili znakovno polje. ReadLine metoda vraća string u kojem je čitava linija. ReadToEnd metoda vraća string sa sadržajem od početne pozicije do kraja streama.

OBJAŠNJENJE

- U .NET pisanje i čitanje teksta izvodi se upotrebom StreamWriter i StreamReader klasa. Za pisanje se koristi StreamWriter.Write metoda koja je preopterećena za sve uobičajene tipove podataka (stringove, znakove, cijele brojeve, realne brojeve itd.) ali ih konvertira u tekst. Za ponovno konvertiranje u prvobitni tip, treba koristiti WriteLine metodu kako bi svaka vrijednost bila u posebnoj liniji.
- Način na koji je string predstavljen ovisi o korištenom kodu: ASCII (7 bitova za znak, ne može sadržavati extended Unicode znakove); Full Unicode (UTF-16, svaki znak je niz od 16 bitova), UTF-7 Unicode (7 bitova za ASCII znakove i višestruke sedmorke za extended, primarno se koristi za 7-bitne protokole, npr. mail, nije uobičajeno), UTF-8 (8 bitova za obične ASCII znakove i višestruke parove za extended)
- .NET ima klasu za svaki tip u System.Text

```
using System;
using System.IO;
using System.Text;

namespace Apress.VisualCSharpRecipes.Chapter05
{
    static class Recipe05_07
    {
        static void Main()
        {
            // Create a new file.
            using (FileStream fs = new FileStream("test.txt", FileMode.Create))
            {
                using (StreamWriter w = new StreamWriter(fs, Encoding.UTF8))
                {
                    // Write a decimal, string, and char.
                    w.WriteLine(124.23M);
                    w.WriteLine("Test string");
                    w.WriteLine('!');
                }
            }
            Console.WriteLine("Press Enter to read the information.");
            Console.ReadLine();

            // Open the file in read-only mode.
            using (FileStream fs = new FileStream("test.txt", FileMode.Open))
            {
                using (StreamReader r = new StreamReader(fs, Encoding.UTF8))
                {
                    // Read the data and convert it to the appropriate data type.
                    Console.WriteLine(Decimal.Parse(r.ReadLine()));
                    Console.WriteLine(r.ReadLine());
                    Console.WriteLine(Char.Parse(r.ReadLine()));
                }
            }

            // Wait to continue.
            Console.WriteLine(Environment.NewLine);
            Console.WriteLine("Main method complete. Press Enter.");
            Console.ReadLine();
        }
    }
}
```

ČITANJE I PISANJE U BINARNU DATOTEKU

- Problem:
- Pisanje točno određenih tipova podataka u binarnu datoteku.

- Rješenje:
- Kreiranjem novog System.IO.FileStream objekta kao reference na datoteku. Za pisanje koristi se FileStream u System.IO.BinaryWriter i preopterećena Write metoda. Za čitanje datoteke koristi se FileStream u System.IO.BinaryReader i metoda Read odgovarajuća očekivanom podatkovnom tipu.

OBJAŠNJENJE

- Čitanje i pisanje binarnih podataka upotrebom BinaryWriter i BinaryReader klasa. Za pisanje se koristi preopterećena BinaryWriter.Write metoda koja podržava sve uobičajene podatkovne tipove (stringove, znakove, cijele brojeve, realne brojeve i td.). Informacija se pretvara u niz bajtova i piše u datoteku. Za stringove se može koristiti preopterećeni konstruktor koji prihvaća System.Text.Encoding objekt.
- Kod binarnih datoteka treba dobro paziti na tipove podataka i koristiti odgovarajuću Read metodu. Za decimalne podatke koristi se ReadDecimal, za string ReadString (uvijek se sprema i dužina stringa).

```
using System;
using System.IO;

namespace Apress.VisualCSharpRecipes.Chapter05
{
    static class Recipe05_08
    {
        static void Main()
        {
            // Create a new file and writer.
            using (FileStream fs = new FileStream("test.bin", FileMode.Create))
            {
                using (BinaryWriter w = new BinaryWriter(fs))
                {
                    // Write a decimal, two strings, and a char.
                    w.Write(124.23M);
                    w.Write("Test string");
                    w.Write("Test string 2");
                    w.Write('!');
                }
            }
            Console.WriteLine("Press Enter to read the information.");
            Console.ReadLine();

            // Open the file in read-only mode.
            using (FileStream fs = new FileStream("test.bin", FileMode.Open))
            {
                // Display the raw information in the file.
                using (StreamReader sr = new StreamReader(fs))
                {
                    Console.WriteLine(sr.ReadToEnd());
                    Console.WriteLine();

                    // Read the data and convert it to the appropriate data type.
                    fs.Position = 0;
                    using (BinaryReader br = new BinaryReader(fs))
                    {
                        Console.WriteLine(br.ReadDecimal());
                        Console.WriteLine(br.ReadString());
                        Console.WriteLine(br.ReadString());
                        Console.WriteLine(br.ReadChar());
                    }
                }
            }

            // Wait to continue.
            Console.WriteLine(Environment.NewLine);
            Console.WriteLine("Main method complete. Press Enter.");
            Console.ReadLine();
        }
    }
}
```

PRONALAŽENJE DATOTEKA KOJE ODGOVARAJU UZORKU

- Problem:
- Procesirati višestruke datoteke ovisno zadanom filteru (*.dll)

- Rješenje:
- Korištenjem preopterećene inačice System.IO.DirectoryInfo.GetFiles ili System.IO.DirectoryInfo.EnumerateFiles metoda koje prihvaćaju filter i vraćaju polje FileInfo objekata. Za rekurzivno pretraživanje svih podmapa, koristi se preopterećena inačica koja prihvaća SearchOption enumeraciju.

OBJAŠNJENJE

- DirectoryInfo i Directory objekti omogućavaju pretraživanje mapa za datotekama koje odgovaraju određenom filteru. Mogu se pretraživati i mape preko preopterećenih DirectoryInfo.GetDirectories ili DirectoryInfo.EnumerateDirectories metoda. Također se može koristiti preopterećeni GetFiles za rekurzivno pretraživanje upotrebom SearchOption.AllDirectories enumeracije.

```
using System;
using System.IO;

namespace Apress.VisualCSharpRecipes.Chapter05
{
    static class Recipe05_10
    {
        static void Main(string[] args)
        {
            if (args.Length != 2)
            {
                Console.WriteLine(
                  "USAGE:  Recipe05_10 [directory] [filterExpression]");
                return;
            }

            DirectoryInfo dir = new DirectoryInfo(args[0]);
            FileInfo[] files = dir.GetFiles(args[1]);

            // Display the name of all the files.
            foreach (FileInfo file in files)
            {
                Console.Write("Name: " + file.Name + "  ");
                Console.WriteLine("Size: " + file.Length.ToString());
            }

            // Wait to continue.
            Console.WriteLine(Environment.NewLine);
            Console.WriteLine("Main method complete. Press Enter.");
            Console.ReadLine();
        }
    }
}
```

RAD SA RELATIVNIM STAZAMA

- Problem:
- Postavljanje tekuće radne mape kako bi se u kodu koristile relativne staze.

- Rješenje:
- Upotrebom statičkih GetCurrentDirectory i SetCurrentDirectory metoda System.IO.Directory klasa.

OBJAŠNJENJE

- Relativne staze se automatski interpretiraju ovisno o trenutnoj radnoj mapi o čemu se informacija može dobiti pozivom Directory.GetCurrentDirectory ili promjenom preko Directory.SetCurrentDirectory. Dodatno, može se koristiti statička GetFullPath metoda System.IO.Path klase kako bi se napravila konverzija relativne staze u apsolutnu uotrebom tekuće radne mape.

```
using System;
using System.IO;

namespace Apress.VisualCSharpRecipes.Chapter05
{
    static class Recipe05_14
    {
        static void Main()
        {

            Console.WriteLine("Using: " + Directory.GetCurrentDirectory());
            Console.WriteLine("The relative path 'file.txt' " +
              "will automatically become: '" +
              Path.GetFullPath("file.txt") + "'");

            Console.WriteLine();

            Console.WriteLine("Changing current directory to c:\\");
            Directory.SetCurrentDirectory(@"c:\");

            Console.WriteLine("Now the relative path 'file.txt' " +
              "will automatically become '" +
              Path.GetFullPath("file.txt") + "'");

            // Wait to continue.
            Console.WriteLine(Environment.NewLine);
            Console.WriteLine("Main method complete. Press Enter.");
            Console.ReadLine();
        }
    }
}
```

DOHVAĆANJE UKUPNOG SLOBODNOG PROSTORA NA DISKU

- Problem:
- Ispitati drive i odrediti koliko je bajtova slobodnog prostora raspoloživo.

- Rješenje:
- Upotrebom DriveInfo.AvaliableFreeSpace svojstva.

OBJAŠNJENJE

- DriveInfo klasa osigurava članove za saznavanje tipa pogona, slobodnog prostora i dr. Da bi se kreiralo novi DriveInfo objekt, treba u konstruktoru navesti slovo pogona ili string korijena.
- Također se može dobiti lista raspoloživih logičkih pogona upotrebom statičke Directory.GetLogicalDrives metode koja vraća polje stringova.
- Za više detalja o svakom pogonu kreira se DriveInfo instanca preko korijena ili slova koje odgovara logičkom pogonu. Za detaljan opis svakog logičkog pogona poziva se DriveInfo.GetDrives metoda koja vraća polje DriveInfo objekata.

```
using System;
using System.IO;

namespace Apress.VisualCSharpRecipes.Chapter05
{
    static class Recipe05_16
    {
        static void Main(string[] args)
        {
            if (args.Length == 1)
            {
                DriveInfo drive = new DriveInfo(args[0]);

                Console.Write("Free space in {0}-drive (in kilobytes): ", args[0]);
                Console.WriteLine(drive.AvailableFreeSpace / 1024);
                Console.ReadLine();
                return;
            }

            foreach (DriveInfo drive in DriveInfo.GetDrives())
            {
                try
                {
                    Console.WriteLine(
                        "{0} - {1} KB",
                        drive.RootDirectory,
                        drive.AvailableFreeSpace / 1024
                        );
                }
                catch (IOException) // network drives may not be available
                {
                    Console.WriteLine(drive);
                }
            }

            // Wait to continue.
            Console.WriteLine(Environment.NewLine);
            Console.WriteLine("Main method complete. Press Enter.");
            Console.ReadLine();
        }
    }
}
```

PROGRAMSKO DODAVANJE KONTROLE

- Problem:
- Programski dodati kontrolu na formu (runtime).

- Rješenje:
- Kreirati instancu odgovarajuće klase kontrole. Zatim dodati objekt te kontrole na formu ili na kontejnersku kontrolu pozivom Controls.Add za taj kontejner (Svojstvo Controls kontejnera vraća instancu ControlCollection.)

OBJAŠNJENJE