

Datoteke, prijenos podataka

10/13

Rad s *Excel* datotekama

❑ MS *Excel* datoteke možemo čitati/pisati koristeći *OleDb Provider*

- U `DataSource` navodi se put do datoteke
- `Extended Properties` označava dodatna svojstva:
 - `Excel 8.0` označava priključak na *Excel* datoteku verzije 8.0 – nužno ga je navesti za rad s *Excel* datotekama
 - `HDR=Yes` označava da se u prvom retku nalaze nazivi stupaca tablice

```
"Provider=Microsoft.Jet.OLEDB.4.0;  
Data Source=C:\\Projects\\podaci.xls;  
Extended Properties='Excel 8.0; HDR=Yes'"
```

❑ Čitanje podataka iz lista (*sheet*)

- Naziv lista navodi se u uglatim zagrada

```
OleDbCommand datotekaComm = new OleDbCommand(  
    "SELECT * FROM [" + nazivLista + "]", datotekaConn);
```

Rad s datotekama s ograničivačem

❑ Datoteke s ograničivačem (delimiter, separator)

- Tekstovne datoteke u kojima su vrijednosti odvojene ograničivačima (zarez, razmak, tabulator, ...)
- CSV datoteke (*Comma Separated Values*) – ograničivač je zarez

❑ Možemo ih čitati koristeći *OleDb Provider*

- U `DataSource` navodi se direktorij u kojem je datoteka
- Svojstvo `Text` označava da se spajamo na tekstualnu datoteku
- `FMT=Delimited` označava da se radi o formatu datoteke s ograničivačima
 - Ograničivač je standardno zarez (,), a može se promijeniti u *Schema.ini* datoteci koja opisuje datoteku s ograničivačima.
 - *Schema.ini* se mora nalaziti u direktoriju datoteke s podacima.
 - Primjer – za ograničivač točka-zarez datoteka *Schema.ini* treba sadržavati redak: `FMT=Delimited(;)`

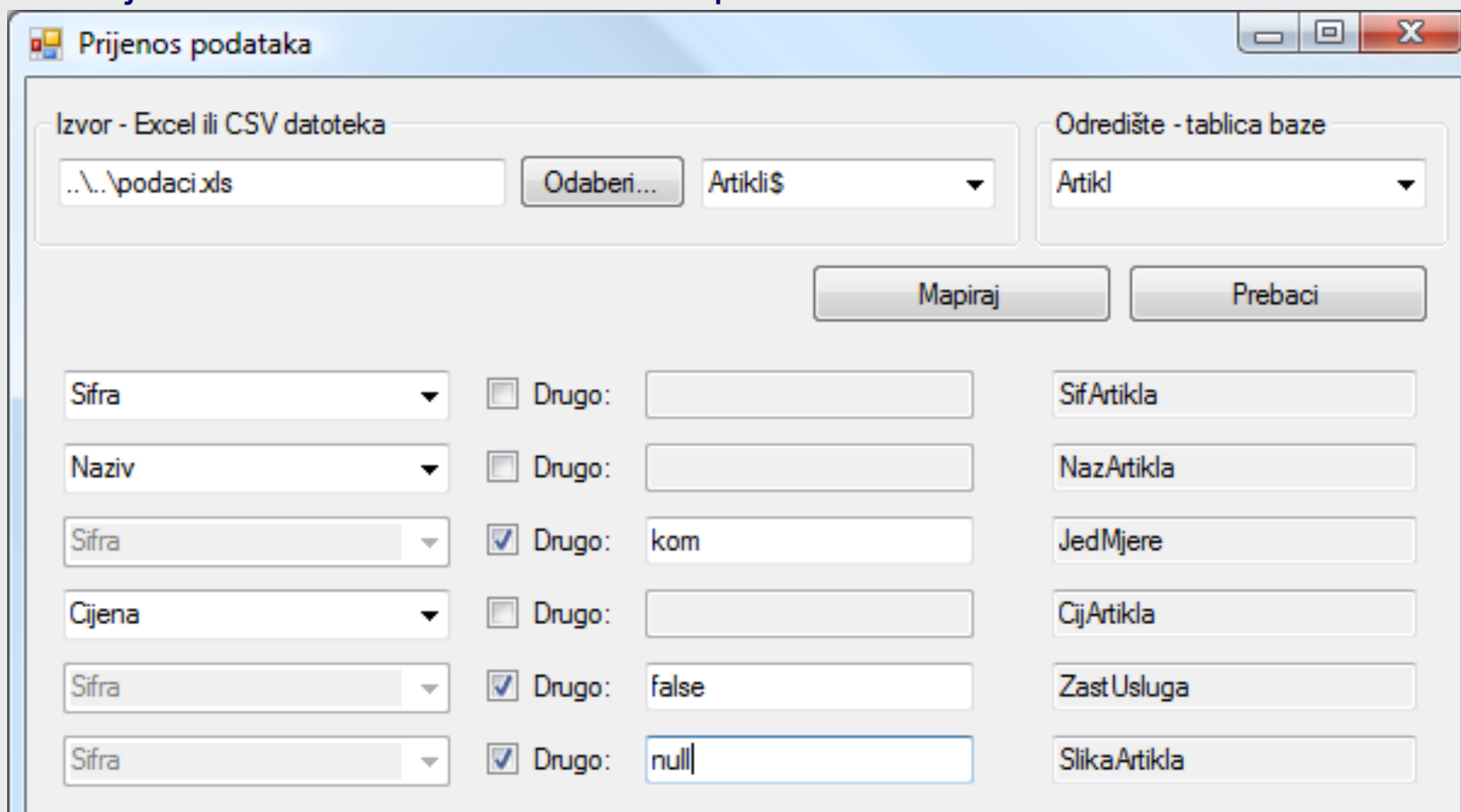
```
"Provider=Microsoft.Jet.OLEDB.4.0;  
Data Source=C:\\Projects;  
Extended Properties='Text; FMT=Delimited; HDR=Yes'"
```

❑ Dohvat podataka: `"SELECT * FROM nazivDatoteke"`

Primjer prijenosa podataka iz Excel/CSV datoteke

□ Primjer: Datoteke \ Import

- Čitanje Excel/CSV datoteke (u prvom retku su nazivi stupaca)
- Mapiranje stupaca Excel/CSV datoteke i stupaca tablice baze podataka
- Prijenos u odabranu tablicu baze podataka



Izvor - Excel ili CSV datoteka

Odredište - tablica baze

Mapiraj Prebaci

Izvor	Drugo:	Odredište
Sifra	<input type="checkbox"/>	SifArtikla
Naziv	<input type="checkbox"/>	NazArtikla
Sifra	<input checked="" type="checkbox"/> kom	JedMjere
Cijena	<input type="checkbox"/>	CijArtikla
Sifra	<input checked="" type="checkbox"/> false	ZastUsluga
Sifra	<input checked="" type="checkbox"/> null	SlikaArtikla

Primjer prijenosa podataka iz Excel/CSV datoteke

❑ Primjer čitanja i spremanja u *DataTable*

```
OleDbDataAdapter adapter = new OleDbDataAdapter(  
    "SELECT * FROM [" + nazivLista + "]", connString);  
DataTable dt = new DataTable();  
adapter.Fill(dt);
```

❑ Prijenos iz datoteke u bazu podataka (preko skupa podataka)

```
// vrijednosti jednog retka koje treba spremiti u bazu  
List<string> vrijednosti;  
DataRow dr = dt.NewRow();  
for (int i = 0; i < vrijednosti.Count; i++)  
{  
    if (vrijednosti[i] == "null") dr[i] = DBNull.Value;  
    else dr[i] = vrijednosti[i];  
}  
dt.Rows.Add(dr);  
...  
adapter.Update(dt);
```

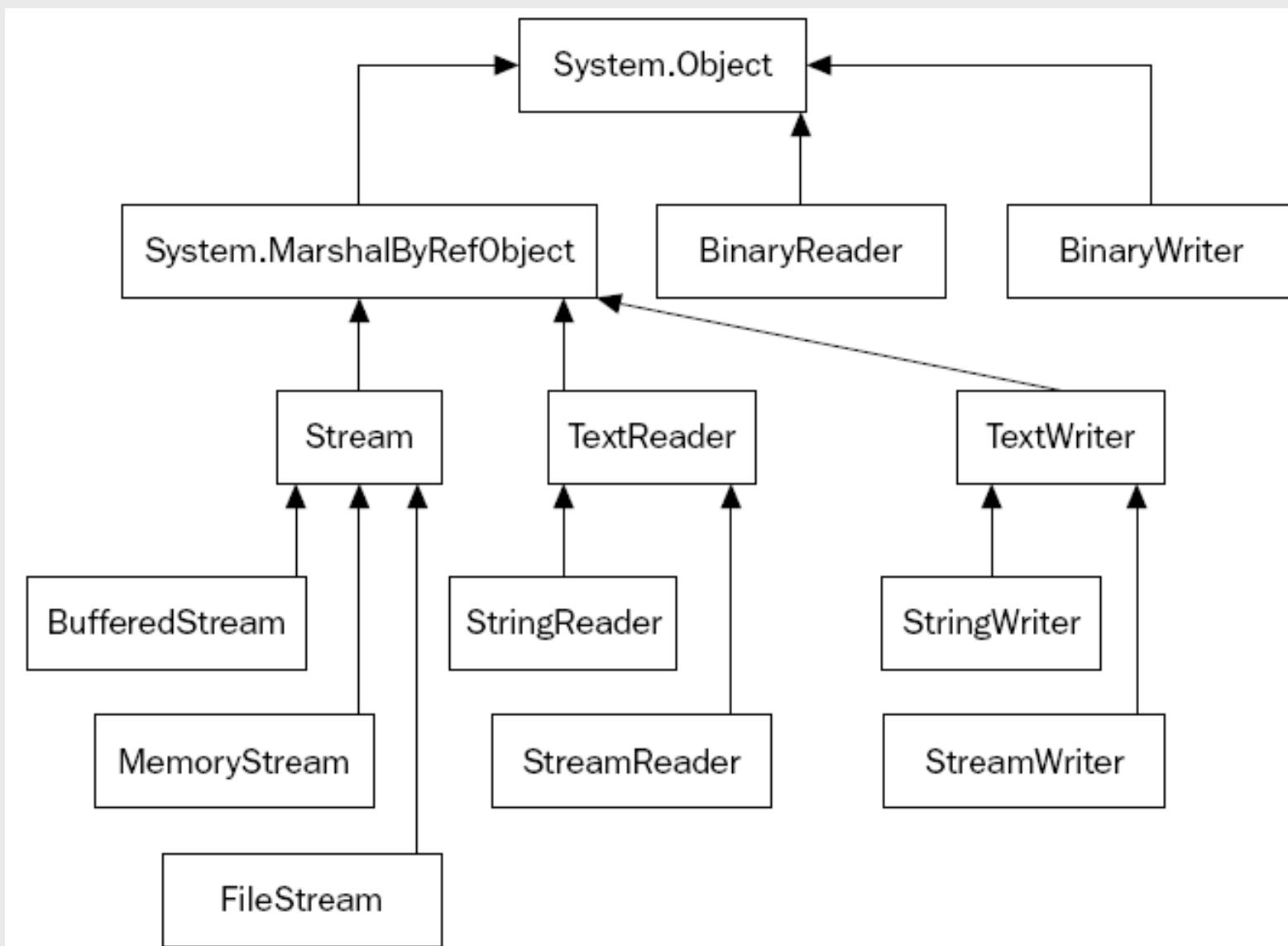
Prostor imena *System.IO*

- ❑ **Ulaz i izlaz (Input/Output tj. I/O) odnose se na postupke čitanja i pisanja podatka putem I/O tokova (*streams*)**
 - Članovi *System.IO* prostora imena koriste se za te postupke

- ❑ **Razredi ovog prostora imena mogu se podijeliti u tri skupine**
 - Razredi za čitanje i pisanje okteta (*bytes*)
 - Razredi za čitanje i pisanje znakova (*character*)
 - Razredi za čitanje i pisanje binarnih podataka

- ❑ **Odluka koji od razreda koristiti ovisi o vrsti podataka**

Hijerarhija razreda



Razred *Stream*

- ❑ Razred *Stream* je apstraktni razred iz kojeg su izvedeni razredi *BufferedStream*, *FileStream* i *MemoryStream* te neki rjeđe korišteni
- ❑ Tok (*stream*) je apstrakcija niza bajtova (npr. datoteka, U/I jedinica, TCP/IP priključnica, međuproceni komunikacijski cjevovod, ...)
- ❑ Na tokove se odnose tri osnovne operacije:
 - Čitanje – prijenos podataka iz toka u neku strukturu podataka
 - Zapisivanje – prijenos podataka iz neke strukture podataka u tok
 - Pozicioniranje – tok može podržavati pozicioniranje po toku (*seeking*), što ovisi o svojstvima toka (npr. pri mrežnoj komunikaciji pozicioniranje uglavnom nema smisla)
- ❑ Iako se ne može izravno instancirati objekt razreda *Stream*, postoje postupci koji vraćaju referencu na *Stream* objekt
 - Primjer: `System.Windows.Forms.OpenFileDialog.OpenFile`

Neki od članova razreda `Stream`

□ Apstraktna svojstva

- `bool CanRead` - `true` ukoliko se iz toka može čitati
- `bool CanSeek` - `true` ukoliko tok podržava pomicanje (`seek`)
- `bool CanWrite` - `true` ukoliko se u tok može pisati
- `long Length` - duljina toka u bajtovima
- `long Position` - vraća ili postavlja trenutnu poziciju u toku

□ Apstraktni postupci

- `void Close()` – zatvara tok i oslobađa njime zauzete resurse
- `void Flush()` – zapisuje sadržaj međuspremnika na pripadnu jedinicu
- `int Read(byte[] buffer, int offset, int count)`
 - čita podatke iz toka i pomiče trenutnu poziciju unutar toka
- `void Write(byte[] buffer, int offset, int count)`
 - zapisuje podatke u tok i pomiče trenutnu poziciju unutar toka

□ Virtualni postupci

- `int ReadByte()`
- `void WriteByte(byte value)`

Razred *FileStream*

❑ **FileStream** razred služi za rad s datotekama

❑ **Najčešće korišteni konstruktori:**

```
FileStream(string filename, FileMode mode)
```

```
FileStream(string filename, FileMode mode, FileAccess how)
```

❑ **how** može imati sljedeće vrijednosti:

FileAccess.Read FileAccess.Write FileAccess.ReadWrite

❑ **mode** može imati sljedeće vrijednosti:

FileMode.Append

Podaci se nadodaju na kraj datoteke.

FileMode.Create

Kreira se nova izlazna datoteka. Postojeća s istim imenom bit će uništena.

FileMode.CreateNew

Kreira novu izlaznu datoteku. Datoteka ne smije već postojati.

FileMode.Open

Otvora postojeću datoteku.

FileMode.OpenOrCreate

Otvora datoteku ako postoji, inače je kreira.

FileMode.Truncate

Otvora postojeću datoteku, ali smanjuje njenu duljinu na 0.

Razred *FileStream* (nastavak)

❑ Iznimke koje se mogu pojaviti prilikom poziva konstruktora **FileStream**:

- | | |
|---|--------------------------------------|
| ■ <code>FileNotFoundException</code> | ako datoteka ne postoji |
| ■ <code>IOException</code> | ako se dogodi pogreška pri otvaranju |
| ■ <code>ArgumentNullException</code> | ako je ime datoteke null |
| ■ <code>ArgumentException</code> | ako je mode parametar nevaljao |
| ■ <code>SecurityException</code> | nedostatak prava željenog pristupa |
| ■ <code>DirectoryNotFoundException</code> | zadani direktorij nije valjan |

Primjer kopiranja datoteka

❑ Otvaranje datoteke:

```
FileStream fin = new FileStream(args[0], FileMode.Open);  
FileStream fout = new FileStream(args[1], FileMode.Create);
```

❑ Kopiranje datoteka:

```
while((i = fin.ReadByte()) != -1) fout.WriteByte((byte)i);
```

❑ Čitanje više bajtova odjednom:

```
int bufSize = 10;  
byte[] buf = new byte[bufSize];  
while((i = fin.Read(buf, 0, bufSize)) > 0)  
    fout.Write(buf, 0, i);
```

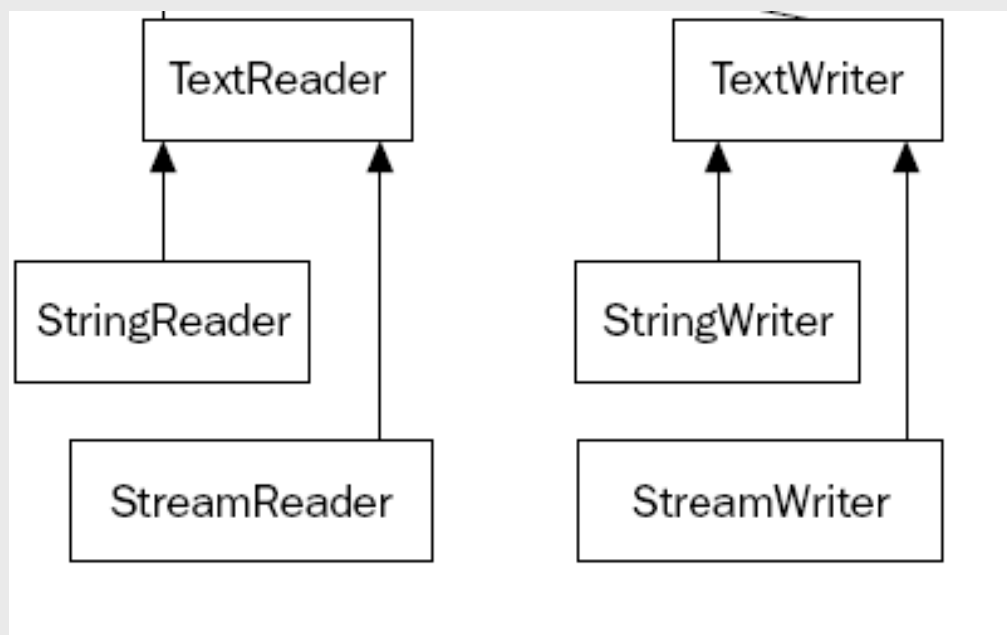
❑ Zatvaranje datoteke:

```
fin.Close();  
fout.Close();
```

Znakovni ulaz i izlaz

❑ Razredi za čitanje i zapisivanje slijednih znakova u datoteke, ali i nizove znakova izvedeni su iz razreda:

- `TextReader` – apstraktni razred koji čita slijedni niz znakova
- `TextWriter` – apstraktni razred koji piše slijedni niz znakova



Primjeri *StreamReader* i *StreamWriter*

❑ Prednost pred čitanjem bajtova: izravno radi s *Unicode* znakovima

■ Najčešće korišteni konstruktori:

- `StreamWriter(Stream stream), StreamWriter(String dat)`
- `StreamReader(Stream stream), StreamReader(String dat)`

■ Postupci za čitanje iz datoteke:

- `int Read()`
- `int Read(char[] buffer, int offset, int numChars)`
- `int ReadBlock(char[] buffer, int offset, int numChars)`
- `string ReadLine()` // vraća null ako je dosegnut kraj datoteke
- `string ReadToEnd()`

■ Postupci za pisanje u datoteku:

- `void Write(type value)`
- `void WriteLine(type value)`

Primjer datoteke s ograničivačem

❑ Primjer: Datoteke\Tokenizer

- Učitavanje vrijednosti iz datoteke s ograničivačem, korištenjem *StreamReader*
- Spremanje u novu datoteku s novim ograničivačem, korištenjem *StreamWriter*

```
Sifra#Naziv#Jamstvo#Cijena
3920036#Olympus Papir P-60LE, 60
5952#Olympus Papir P-A6WPAS, 120
3789#Olympus P-P100, 100 listova
4140141#HAMA ramica za slike CLI
5809#Olympus Daljinski upravljač
4140227#Pentax objektiv 18-250mm
4140226#Pentax objektiv DA 50-20
```

Tokenizer

Izvorišna datoteka: \RPPP_Primjeri\Datoteke\Tokenizer\dat.txt Delimiter: #

Odredišna datoteka: RPPP_Primjeri\Datoteke\Tokenizer\dat2.txt Delimiter: ?

Sifra	Naziv	Jamstvo	Cijena
3920036	Olympus Papir P-60LE, 60 listova Passport papira (laminirani) za P-330(N)E	---	445.00
5952	Olympus Papir P-A6WPAS, 120 listova + toner A6 Passport papir za P-400/P-440 /13582	---	821.00
3789	Olympus P-P100, 100 listova + toner A6 papir za P-10/P-11 /N1448600	---	293.00
4140141	HAMA ramica za slike CLIP-FIX 70x100 (63050)	---	121.00
5809	Olympus Daljinski upravljač RM-1 za C-50/40/30/2000/5000/5060/750UZ/mju 800/mju...	---	182.00
4140227	Pentax objektiv 18-250mm, F3,5-6,3EDAL	1 god.	4.199.00
4140226	Pentax objektiv DA 50-200mm, 4-5,6	1 god.	1.499.00

Primjer datoteke s ograničivačem

❑ Čitanje datoteke i odvajanje vrijednosti

```
StreamReader reader = new StreamReader(nazivDatoteke);  
while ((redak = reader.ReadLine()) != null)  
{  
    lista = redak.Split(new char[] { delimiter });  
    ...  
}  
reader.Close();
```

❑ Pisanje u novu datoteku s novim ograničivačem

```
StreamWriter writer = new StreamWriter(nazivDatoteke);  
for (int i = 0; i < listViewPodaci.Items.Count; i++)  
{  
    string redak = ...;  
    for (int j = 0; j < listViewPodaci.Items[i].SubItems.Count; j++)  
        redak += delimiter + listViewPodaci.Items[i].SubItems[j].Text;  
    writer.WriteLine(redak);  
}  
writer.Close();
```


Preddefinirani tokovi

❑ Postoje tri preddefinirana znakovna toka

- `Console.In` (vraća objekt tipa `TextReader`)
- `Console.Out` (vraća objekt tipa `TextWriter`)
- `Console.Error` (vraća objekt tipa `TextWriter`)

❑ Preddefinirani tokovi mogu se preusmjeriti na bilo koji kompatibilni U/I uređaj ili datoteku sljedećim postupcima

- posebnim znakovima `<` ili `>` u komandnoj liniji ukoliko to podržava operacijski sustav
- nekim od postupaka:

```
static void SetIn(TextReader input);  
static void SetOut(TextWriter output);  
static void SetError(TextWriter output);
```

Binarne datoteke

- ❑ Za čitanje i zapisivanje ugrađenih C# tipova podataka u binarnom formatu koristimo razrede `BinaryReader` i `BinaryWriter`

- ❑ Najčešće korišteni konstruktori su:

```
BinaryWriter(Stream outputStream)
```

```
BinaryReader(Stream inputStream)
```

- ❑ Postupak `void Write(tip val)` kao parametar može primiti jedan od sljedećih tipova podataka:

```
sbyte, byte, byte[], bool, short, ushort, int, uint, long, ulong,  
float, double, char, char[], string
```

- ❑ `BinaryReader` definira postupke čitanja za svaki ugrađeni tip
- ❑ `BinaryReader` također definira i sljedeće `Read` postupke:

```
int Read(byte[] buf, int offset, int num)
```

```
int Read(char[] buf, int offset, int num)
```

BinaryReader postupci za čitanje

❑ Postupci za čitanje pojedinog ugrađenog tipa podatka

```
bool ReadBoolean()  
byte ReadByte()  
sbyte ReadSByte()  
byte[] ReadBytes(int num)  
char ReadChar()  
char[] ReadChars(int num)  
double ReadDouble()  
float ReadSingle()  
short ReadInt16()  
int ReadInt32()  
long ReadInt64()  
ushort ReadUInt16()  
uint ReadUInt32()  
ulong ReadUInt64()  
string ReadString() // samo za stringove pisane s BinaryWriter
```

❑ Svi postupci bacaju iznimku `EndOfStreamException` ako je dosegnut kraj datoteke.

Primjer binarne datoteke

```
string artikl; // ime predmeta
int kolicina; // količina na skladištu
double cijena; // jedinična cijena

BinaryWriter bout = new BinaryWriter(new
FileStream("zaliha.dat", FileMode.Create));

// pisanje jednog zapisa
bout.Write("Cekic");
bout.Write(10);
bout.Write(3.95);
...
// čitanje jednog zapisa
artikl = bin.ReadString();
kolicina = bin.ReadInt32();
cijena = bin.ReadDouble();
```

Razredi *MemoryStream*, *StringReader*, *StringWriter*

- ❑ Ponekad je korisno čitati ili pisati u neko polje ili string, a ne direktno u neku datoteku na disku.
- ❑ *MemoryStream* je izveden iz razreda *Stream* i koristi polje bajtova za U/I - jedan od konstruktora je:
 - `MemoryStream(byte[] buf)`
- ❑ Ukoliko je umjesto polja bajtova potrebno koristiti string, mogu se upotrijebiti *StringReader* i *StringWriter* - konstruktori su:
 - `StringReader(string str)` – implementira *TextReader* koji čita string
 - `StringWriter()` – implementira *TextWriter*, a tekst sprema u automatski kreirani *StringBuilder*
- ❑ Sadržaj pohranjen u *StringWriter* može se dobiti s `ToString()`

Primjer *StringReader*, *StringWriter*

```
// instanciranje pisača
StringWriter strwtr = new StringWriter();
// pisanje "na string"
for(int i=0; i < 10; i++)
    strwtr.WriteLine("Redak: " + i);

// sadržaj pisača
Console.WriteLine (strwtr.ToString());

// instanciranje čitača
StringReader strrdr =
    new StringReader(strwtr.ToString());
//StringReader strrdr =
//  new StringReader("prvi\ndrugi\ntreci\n"); // opcija

// čitanje "iz stringa"
string str;
while((str = strrdr.ReadLine()) != null)
    Console.WriteLine(str);
```

Direktne datoteke

❑ Osim slijednih datoteka C# omogućava i direktne datoteke.

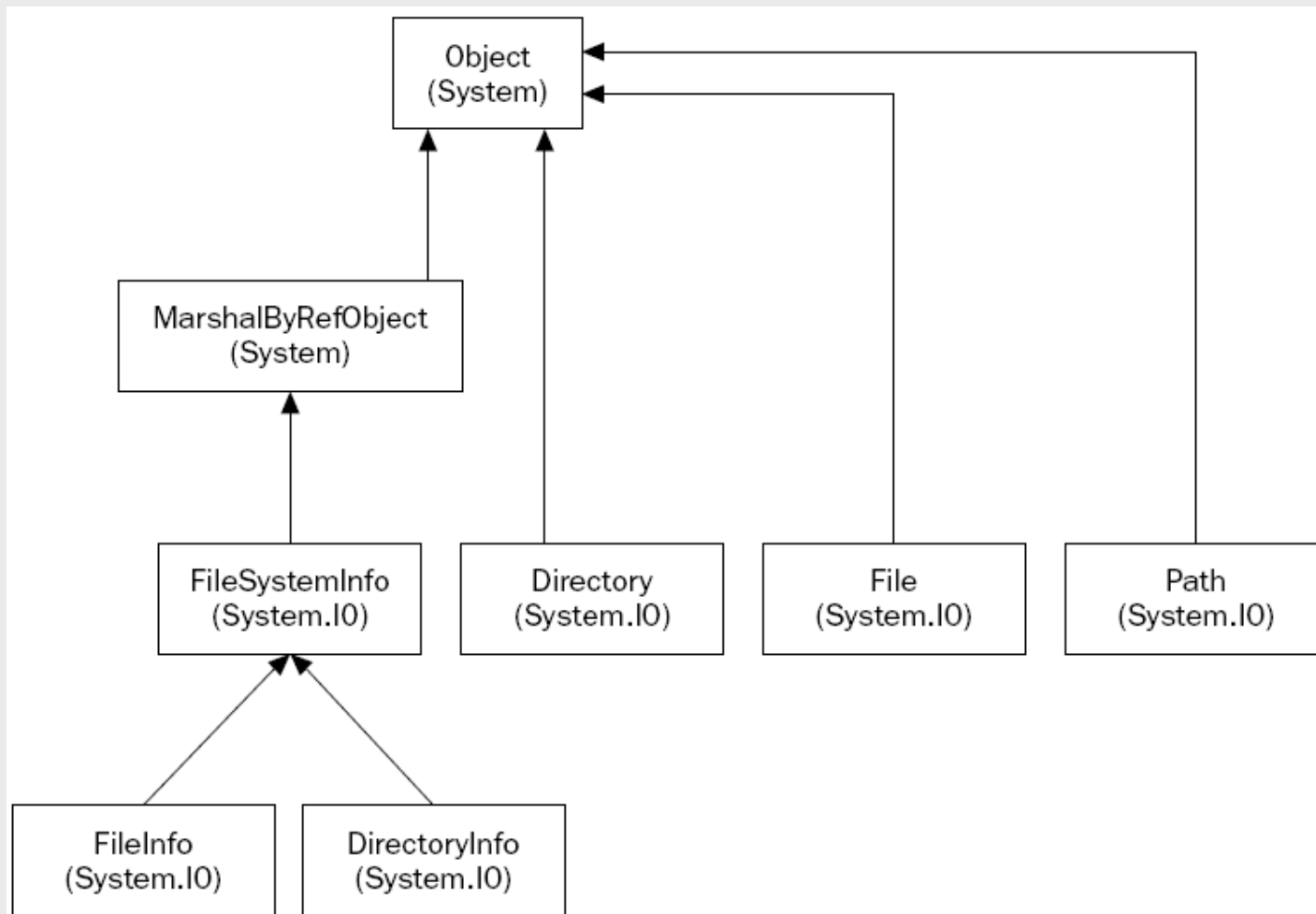
❑ Za pozicioniranje unutar datoteke koristi se postupak:

- `long Seek (long newPos, SeekOrigin origin)`
- *newPos* predstavlja novu poziciju (u bajtovima) pokazivača na datoteku na lokaciju zadanu s *origin*
- *origin* može imati sljedeće vrijednosti:
 - `SeekOrigin.Begin` pozicioniranje od početka datoteke
 - `SeekOrigin.Current` pozicioniranje od trenutne lokacije
 - `SeekOrigin.End` pozicioniranje od kraja datoteke

❑ Primjer

```
// čitanje s pozicije i iz FileStream f
fileStream.Seek(i, SeekOrigin.Begin);
znak = (char)fileStream.ReadByte();
Console.WriteLine("\n{0}. vrijednost: {1} "
    , fileStream.Position, znak);
```

Razredi *DirectoryInfo* i *FileInfo*



Razredi *DirectoryInfo* i *FileInfo*

❑ Apstraktni razred *System.IO.FileSystemInfo*

- Bazni razred za razrede *DirectoryInfo* i *FileInfo*
- *FileSystemInfo* sadrži i sljedeće članove:

<code>Attributes</code>	Vraća ili postavlja <i>FileAttributes</i> kolekciju (enumeraciju) s atributima datoteke (skrivena, normalna, jedinica, ...)
<code>CreationTime</code>	Vraća vrijeme stvaranja datoteke
<code>Exists</code>	Postoji li zadana datoteka
<code>Extension</code>	Vraća string koji predstavlja ekstenziju datoteke
<code>FullName</code>	Potpuni naziv (apsolutni) datoteke
<code>LastAccessTime</code>	Vraća ili postavlja vrijeme zadnjeg pristupa datoteci
<code>LastWriteTime</code>	Vraća ili postavlja vrijeme zadnjeg zapisivanja u datoteku
<code>Delete</code>	Briše datoteku
<code>Refresh</code>	Osvježava stanje datoteke

❑ Napomena: direktorij je također "datoteka"

❑ *FileAttributes* enumeracija ima (između ostalog) sljedeće članove:

- *Archive*, *Directory*, *Hidden*, *Normal*, *ReadOnly*, *System*

Razred *DirectoryInfo*

- ❑ Razred `DirectoryInfo` - stvaranje, premještanje (preimenovanje) i iteraciju (enumeraciju) kroz (pod)direktorije
- ❑ Neki članovi razreda `DirectoryInfo`

Parent	Vraća nad-direktorij direktorija
Root	Vraća osnovni (root) dio direktorija (u stvari naziv diska)
Create	Stvara direktorij
CreateSubDirectory	Stvara poddirektorij (ili više poddirektorija)
GetDirectories	Vraća polje objekata tipa <code>DirectoryInfo</code> koje odgovara poddirektorijima
GetFiles	Vraća polje objekata tipa <code>FileInfo</code> koje odgovara datotekama u tom direktoriju
MoveTo	Premješta direktorij predstavljen instancom objekta na drugo mjesto

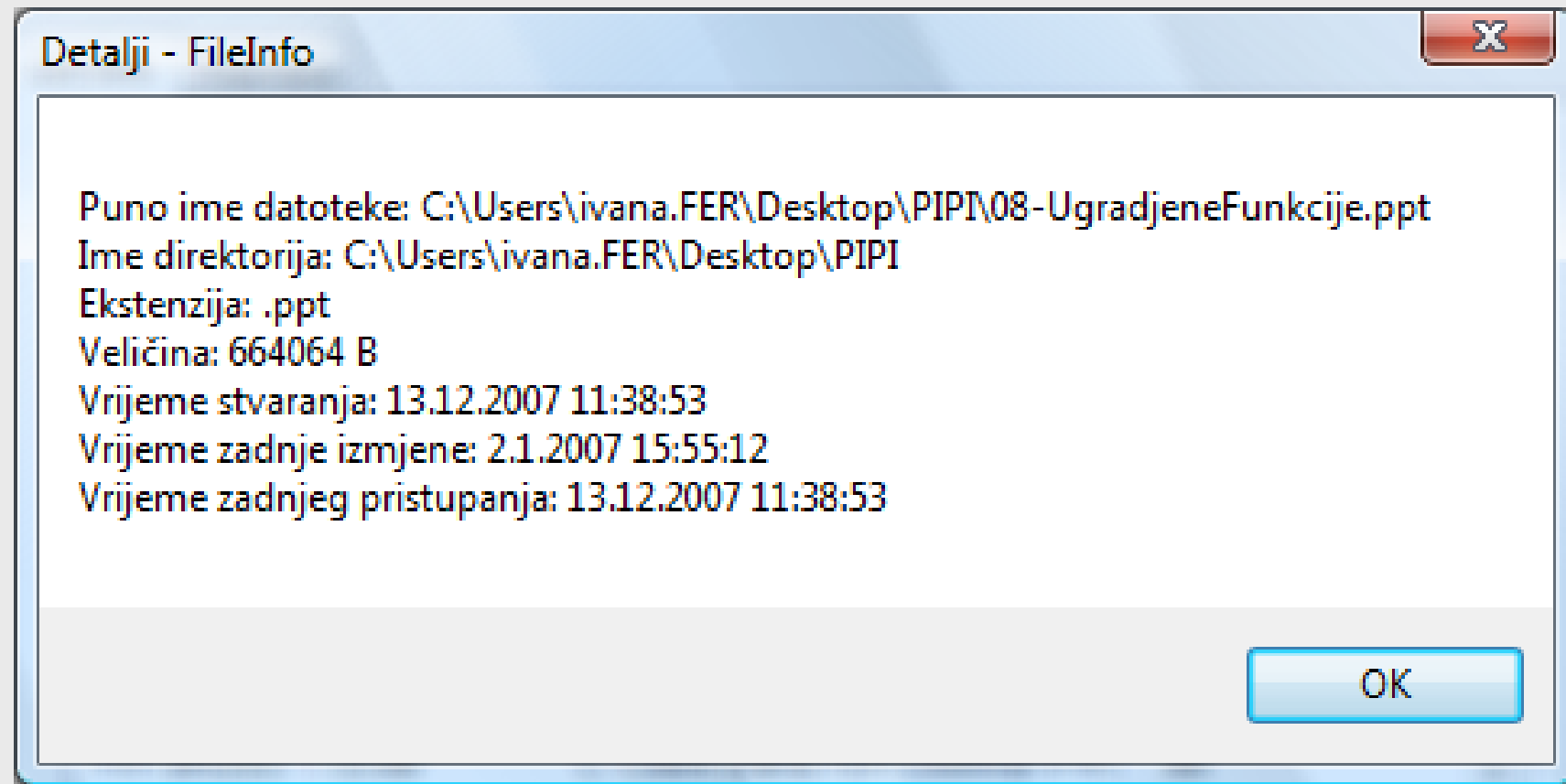
❑ Primjer uporabe

```
DirectoryInfo dir1 = new DirectoryInfo(@"C:\WINDOWS");  
Console.WriteLine("Full Name is : {0}", dir1.FullName);  
Console.WriteLine("Attributes are : {0}",  
    dir1.Attributes.ToString());
```

Primjer izlista datoteka

❑ Primjer iz poglavlja GUI (📁 GUI\ListaDatoteka)

- Prikaz datoteka/direktorija u *ListView* i *TreeView* kontroli
- Desni klik na element u *ListView* – detalji o odabranom elementu



Svojstva datoteka

❑ Prikaz informacija o odabranoj datoteci

```
FileInfo datoteka;  
...  
string detalji =  
"Puno ime datoteke: " + datoteka.FullName + "\n"  
+ "Ime direktorija: " + datoteka.DirectoryName + "\n"  
+ "Ekstenzija: " + datoteka.Extension + "\n"  
+ "Veličina: " + datoteka.Length + " B\n"  
+ "Vrijeme stvaranja: " + datoteka.CreationTime + "\n"  
+ "Vrijeme zadnje izmjene: " + datoteka.LastWriteTime + "\n"  
+ "Vrijeme zadnjeg pristupanja: " + datoteka.LastAccessTime  
+ "\n";
```

❑ Implementacija desnog klika na element liste

- ContextMenu, dohvat koordinata

Zadaci za vježbu

- ❑ **Napisati aplikaciju koja binarno uspoređuje dvije datoteke**

- ❑ **Napisati aplikaciju za grupni (*batch*) uvoz slike**
 - Id artikla i naziv slike (put do slike) zadani su u opisnoj datoteci.
 - Treba pročitati datoteku te ažurirati artikle.

Zadaci za vježbu

- ❑ Napisati aplikaciju za prijenos podataka iz *Excel* datoteke u bazu podataka Firma. Izvorni podaci su:

A	B	C	D	E	F
ImeOsobe	PrezimeOsobe	JMBG	AdrPartnera	NazMjestaPartnera	PostBrMjestaPartnera
Biljana	Ljubešić	1001971330031	Borovik 10	Zagreb	10000
Vladimir	Linke	0603941330092	Tržnica Utrine	Zagreb	10000

G	H	I
AdrlIsporuke	NazMjestalsporuke	PostBrMjestalsporuke
Borovik 10	Zagreb	10000
Dalmmatinska 2	Zagreb	10000

- ❑ *AdrPartnera*, *AdrlIsporuke*, te pripadni *IdMjestaPartnera* i *IdMjestaIsporuke* (pročitani iz tablice *Mjesto* na temelju *NazMjesta* i *PostBrMjesta*) treba kopirati u tablicu *Partner*.
- ❑ *IdPartnera* je *identity*, a *TipPartnera* 'O'.
- ❑ *ImeOsobe*, *PrezimeOsobe* i *JMBG* trebaju biti prebačeni u tablicu *Osoba* (*IdOsobe* treba biti jednak *IdPartner*)

Reference

❑ **C# School Book – Free ebook**

- <http://www.programmersheaven.com/2/CSharpBook>

Windows servisi

(Windows) servisi

□ Windows servisi (engl. *Windows Services*)

- dugotrajni (*long-running*) izvršni programi
- nemaju korisničkog sučelja, ne zahtijevaju interakciju s korisnikom
- izvršavaju se neovisno o prijavi korisnika na računalo
- mogu biti automatski pokrenuti pri pokretanju operacijskog sustava
- moraju biti instalirani na računalo da bi se mogli pokrenuti
- OS Unix imaju sličan tip programa pod nazivom demon (*daemon*)

□ Životni vijek windows servisa – nekoliko internih stanja

- instalacija servisa na računalo - servis se učitava u *Services Control Manager* (centralno mjesto za administraciju servisa)
- pokretanje – nakon što je servis instaliran može se pokrenuti.
 - Pokretanje se obavlja iz *Management Console*, pozivom *Start* metode iz programskog koda, automatski prilikom pokretanja računala ...
- nakon pokretanja servis se nalazi u *Running* stanju sve dok ne bude zaustavljen (stanje *Stopped*), pauziran (*Paused*) ili dok računalo ne bude isključeno.

Upravljanje instaliranim servisima

☐ Upravljanje servisom vrši se kroz *Services Management Console*

- Control Panel \ Administrative Tools \ Services
- ili My Computer – Manage \ Services and Applications \ Services

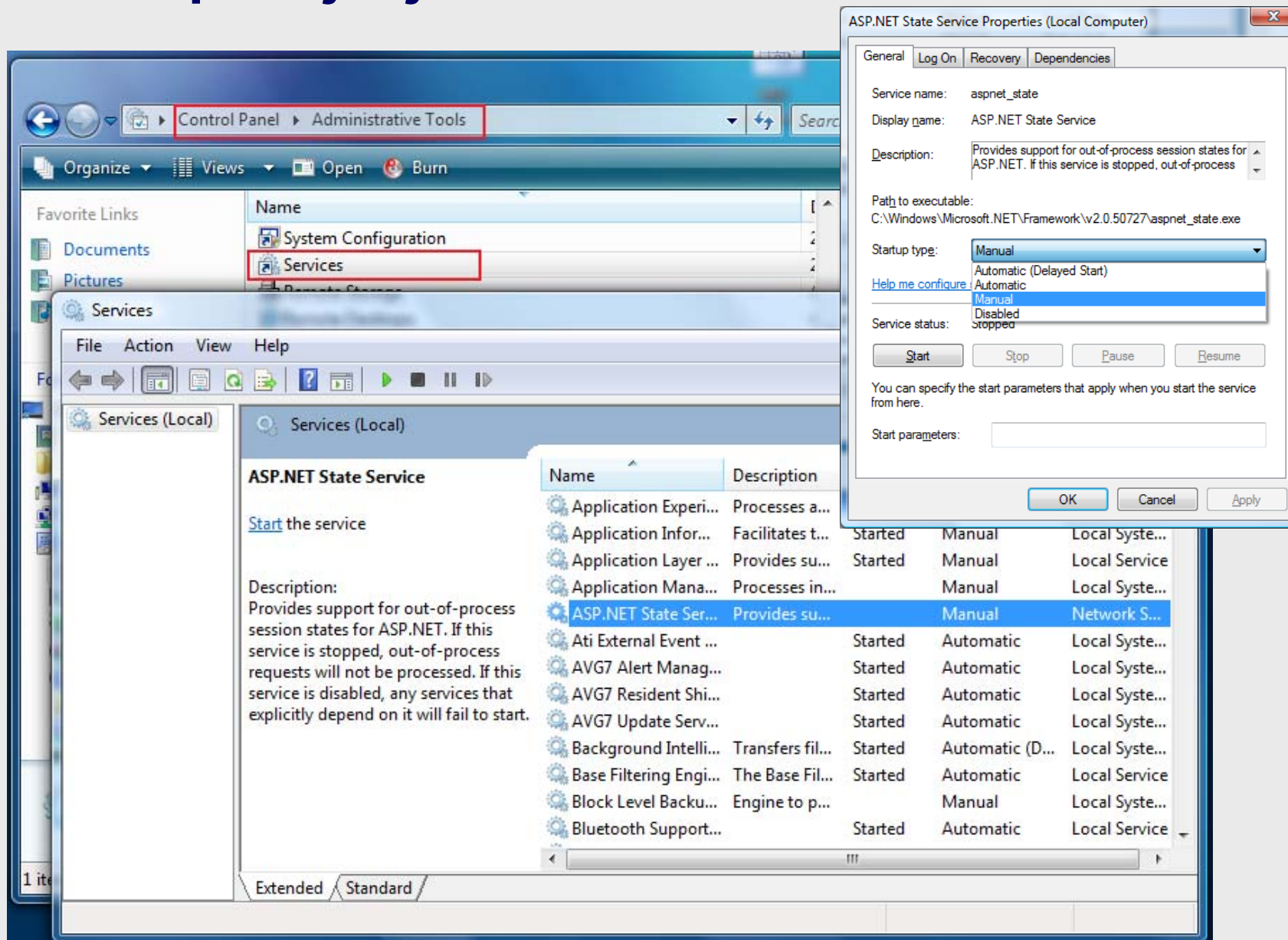
☐ Pregled postojećih servisa na računalu i njihovo stanje

- pokretanje, zaustavljanje, pauziranje servisa
- administriranje načina pokretanja servisa
 - automatski – servis se pokreće pri pokretanju OS
 - automatski odgođeno – nakon pokretanja OS kada završe početne akcije koje zauzimaju procesne resurse
 - ručno – korisnik pokreće iz konzole ili drugi program pokreće Start metodom
 - onemogućeno

☐ Iako servisi obično nemaju korisničko sučelje, programer može dodati formu ili neku drugu UI komponentu.

- U tom slučaju je potrebno označiti “*Allow service to interact with desktop*” u *Logon* dijelu panela s postavkama.

Upravljanje instaliranim servisima



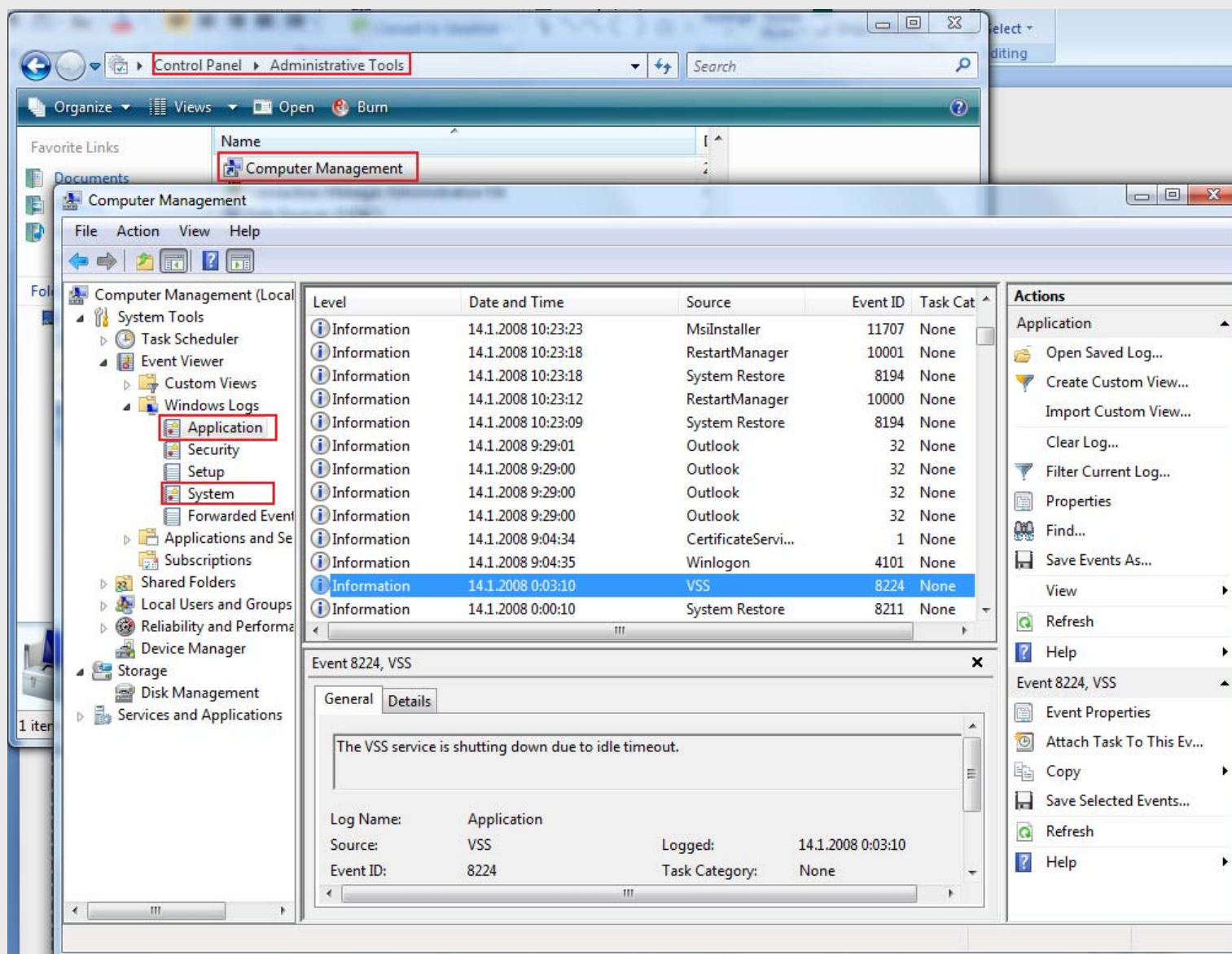
Informacije o radu servisa

- ❑ Windows servisi nemaju korisničko sučelje pa rezultate izvođenja evidentiraju na drugačiji način u odnosu na obične aplikacije.
- ❑ Osnovne informacije o promjeni stanja servisa i njihovim akcijama zapisuju se u dnevnik događaja (*event log*)
 - pregled dnevnika obavlja se u dijelu konzole *Event Viewer*
 - Pokretanje: Control Panel \ Administrative Tools \ Event Viewer
 - preglednik evidentira i druge događaje te podatke koje su zapisale (logirale) i druge aplikacije te sam operacijski sustav.

```
Event Type:      Information
Event Source:    Service Control Manager
Event Category:  None
Event ID:        7036
```

Date:	<u>Log Name:</u>	Application	
Time:	<u>Source:</u>	MSSQL\$IVANA	<u>Logged:</u> 15.5.2008 17:13:34
User:	<u>Event ID:</u>	17895	<u>Task Category:</u> (2)
Computer	<u>Level:</u>	Information	<u>Keywords:</u> Classic
Descript	<u>User:</u>	N/A	<u>Computer:</u> ivana-zpr.fer.hr
The Worl	<u>OpCode:</u>		
For more			
http://g			

Informacije o radu servisa



Izrada Windows servisa

❑ Izrada windows servisa u okolini *Visual Studio*

- prilikom kreiranja novog projekta odabire se predložak *Windows Service* koji automatski u projekt dodaje odgovarajući razred
- dodani razred nasljeđuje `System.ServiceProcess.ServiceBase` razred te je moguće implementirati ponašanje servisa pisanjem preopterećenih postupaka (`OnStart`, `OnStop`, `OnPause`, `OnContinue`,...).
- `ServiceBase` razred nalazi se u *System.ServiceProcess* knjižnici pa je referenca na knjižnicu također automatski dodana

```
namespace ZPR.ServiceDemo
{
    public partial class NadzornikServis: ServiceBase
    {
        ....
        public NadzornikServis()
```

❑ Primjer: **WinServis\NadzornikServis – NadzornikServis.cs**

- Servis za praćenje promjena (brisanja, dodavanja, izmjene datoteka i direktorija) u zadanom direktoriju

Izrada Windows servisa

❑ **Primjer:** **WinServis\NadzornikServis – Program.cs**

- `Main` postupak mora izdati `Run` naredbu svim servisima koje projekt sadrži.
- Pozivom `Run` metode servis se učitava u *Services Management Console*

```
static void Main()  
{  
    ServiceBase[] ServicesToRun;  
    ServicesToRun = new ServiceBase[] { new NadzornikServis() };  
    ServiceBase.Run(ServicesToRun);  
}
```

- ## ❑ **Ukoliko je windows servis projekt nastao korištenjem predloška odgovarajući kod u `Main` postupak je automatski dodan.**

Programska arhitektura Windows servisa

❑ Ponašanje servisa definira se preko događaja osnovnog razreda `System.ServiceProcess.ServiceBase`

- `OnStart` – izvršava se prilikom pokretanja servisa
- `OnPause` – prilikom pauziranja servisa
- `OnStop` – prilikom zaustavljanja servisa
- `OnContinue` – pri povratku iz pauziranog stanja u stanje izvršavanja servisa
- `OnShutdown` – prilikom isključivanja sustava
- `OnCustomCommand` – prilikom zadavanja korisničke naredbe (prima *int* parametar)
- `OnPowerEvent` – prilikom power management događaja (npr. baterija slaba, suspend)

❑ Da bi servis imao funkcionalnost događaj `OnStart` obavezno mora sadržavati kod. (Npr. kod za pokretanje nadgledanja direktorija.)

Programska arhitektura windows servisa

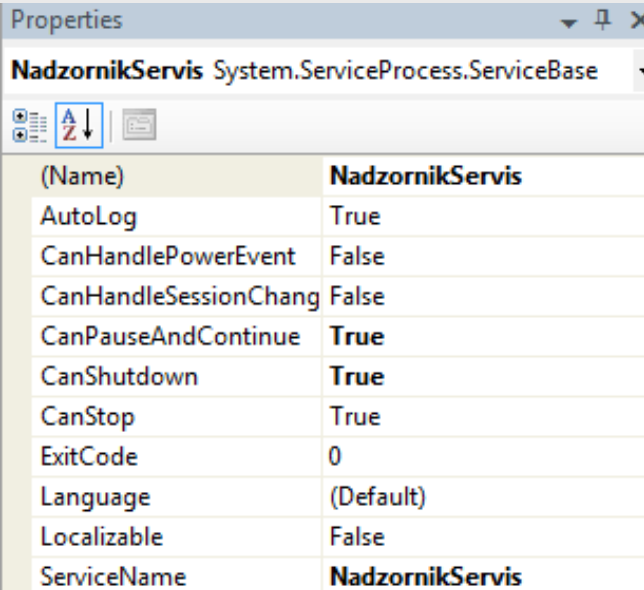
❑ Svojstva `System.ServiceProcess.ServiceBase` koja definiraju ponašanje servisa:

- `AutoLog` – automatsko zapisivanje u event log uobičajenih događaja (npr. *Install* ili *Start*)
- `CanHandlePowerEvent` – da li servis obrađuje *Power management* događaje (npr. baterija slaba, *suspend*)
- `CanStop` – mogućnost poziva zaustavljanja servisa (metoda `OnStop`)
- `CanPauseAndContinue` – mogućnost poziva `OnPause` i `OnContinue` metoda (pauziranje i nastavak izvršavanja servisa)
- `CanShutdown` – mogućnost izvršavanja `OnShutdown` metode na događaj isključivanja računala

❑ Primjer: **WinServis\NadzornikServis** – **NadzornikServis.cs**

- U pogledu dizajna `NadzornikServis.cs` datoteke u prozoru *Properties* postaviti svojstva

`CanPauseAndContinue`, `CanShutdown`, `CanStop` na *true*



The screenshot shows the 'Properties' window in Visual Studio for the project 'NadzornikServis'. The 'System.ServiceProcess.ServiceBase' properties are displayed. The table below represents the data shown in the screenshot.

(Name)	NadzornikServis
AutoLog	True
CanHandlePowerEvent	False
CanHandleSessionChange	False
CanPauseAndContinue	True
CanShutdown	True
CanStop	True
ExitCode	0
Language	(Default)
Localizable	False
ServiceName	NadzornikServis

Razred FileSystemWatcher

❑ Razred FileSystemWatcher

- nadzire sustav datoteka (*file system*) i podiže događaje kada se dogodi promjena u nadziranom kazalu ili datoteci.

❑ Svojstva

- `Path` – putanja do nadziranog direktorija ili datoteke
- `Filter` – određuje što će se nadzirati u direktoriju npr. `*.*` ili `*.exe`
- `EnableRaisingEvents` - određuje da li je komponenta aktivna
- `NotifyFilter` – enumerator tipa `NotifyFilters` koji određuje koje promjene se prate npr. promjene u nazivu datoteke, vremenu zadnjeg pisanja u datoteku (`FileName`, `DirectoryName`, `LastWrite`,...)

❑ Događaji

- (`FileSystemEventHandler d`, `FileSystemEventArgs e`)
- `Changed` – podignut u slučaju promjene nad nadziranim kazalom ili datotekom
- `Created` – u slučaju kreiranja novog direktorija ili datoteke
- `Deleted` - u slučaju brisanja

Razred FileSystemWatcher

❑ Događaji

- (RenamedEventHandler d, RenamedEventArgs e)
- Renamed – u slučaju preimenovanja direktorija ili datoteke

❑ Razredi FileSystemEventArgs i RenamedEventArgs nasljeđuju razred EventArgs koji je bazni razred za razrede koji sadrže podatke o događajima.

❑ Svojstva FileSystemEventArgs razreda:

- ChangeType – tip događaja koji se dogodio
- FullPath – puna putanja do promijenjenog direktorija ili datoteke
- Name – ime promijenjenog direktorija ili datoteke

❑ Svojstva RenamedEventArgs razreda:

- ChangeType – tip događaja koji se dogodio
- FullPath – nova puna putanja do preimenovanog direktorija ili datoteke
- Name – novo ime preimenovanog direktorija ili datoteke
- OldFullPath – stara puna putanja do preimenovanog direktorija ili datoteke
- OldName – staro ime preimenovanog direktorija

Realizacija događaja servisa nadzornik

❑ Primjer: WinServis\NadzornikServis – NadzornikServis.cs

- Događaji prilikom kojih se informacije o promjeni zapisuju u dnevnik OS

```
private void fileSystemWatcher1_Changed(  
    object s, System.IO.FileSystemEventArgs e)  
{  
    EventLog.WriteEntry("NadzornikServis: "  
        + e.FullPath + " " + e.ChangeType.ToString());  
}
```

```
private void fileSystemWatcher1_Renamed(object s,  
    System.IO.RenamedEventArgs e)  
{  
    EventLog.WriteEntry("NadzornikServis: " +  
        e.OldFullPath + " RENAMED TO " + e.FullPath);  
}
```

Konfiguracijska datoteka

- ❑ U datoteku `App.config` zapisana je putanja do direktorija koji će windows servis pratiti (*Path*):

```
<appSettings>
  <add key="Path" value="c:\" />
</appSettings>
```

- ❑ **Primjer:**  **WinServis\NazdzornikServis – NazdzornikServis.cs**

- Čitanje konfiguracijske datoteke, postavljanje direktorija kojeg `FileSystemWatcher` objekt prati i zapis u log:

```
private void SetPath()
{
    ConfigurationManager.RefreshSection("appSettings");//osvježavanje
    fileSystemWatcher1.Path =
        ConfigurationManager.AppSettings["Path"].ToString();
    EventLog.WriteEntry("NadzornikServis: WATCHING-
        "+fileSystemWatcher1.Path);
}
```

Događaji servisa - OnStart, OnStop

Primjer: WinServis\NadzornikServis – NadzornikServis.cs

- **OnStart** postupak servisa, dakle, mora postaviti praćeni direktorij, pokrenuti nadziranje te zapisati informacije u log

```
protected override void OnStart(string[] args) {  
    isPaused = false;  
    SetPath();  
    fileSystemWatcher1.EnableRaisingEvents = true;  
    EventLog.WriteEntry("NadzornikServis : STARTED at" +  
        DateTime.Now.ToShortTimeString());  
}
```

- **OnStop** postupak zaustavlja praćenje direktorija i zapisuje u log

```
protected override void OnStop()  
{  
    fileSystemWatcher1.EnableRaisingEvents = false;  
    EventLog.WriteEntry("NadzornikServis": STOPPED at "+  
        DateTime.Now.ToShortTimeString());  
}
```

Događaji servisa – OnCustomCommand

❑ Omogućuje ugradnju korisnički definiranih postupaka

- Prima int parametar koji određuje koju radnju servis treba obaviti.
- Vrijednosti parametra od 0 do 127 su sistemski zauzete pa je za korisnički definirane komande dozvoljeno koristiti vrijednosti 128 do 256.

❑ Primjer: WinServis\NazdzornikServis – NazdzornikServis.cs

- Komanda 128 osvježava podatke o nadgledanom direktoriju iz konfiguracyjske datoteke.

```
protected override void OnCustomCommand(int command)
{
    base.OnCustomCommand(command);
    if (command == 128)
    {
        SetPath();
    }
}
```

Ugradnja i izvođenje servisa

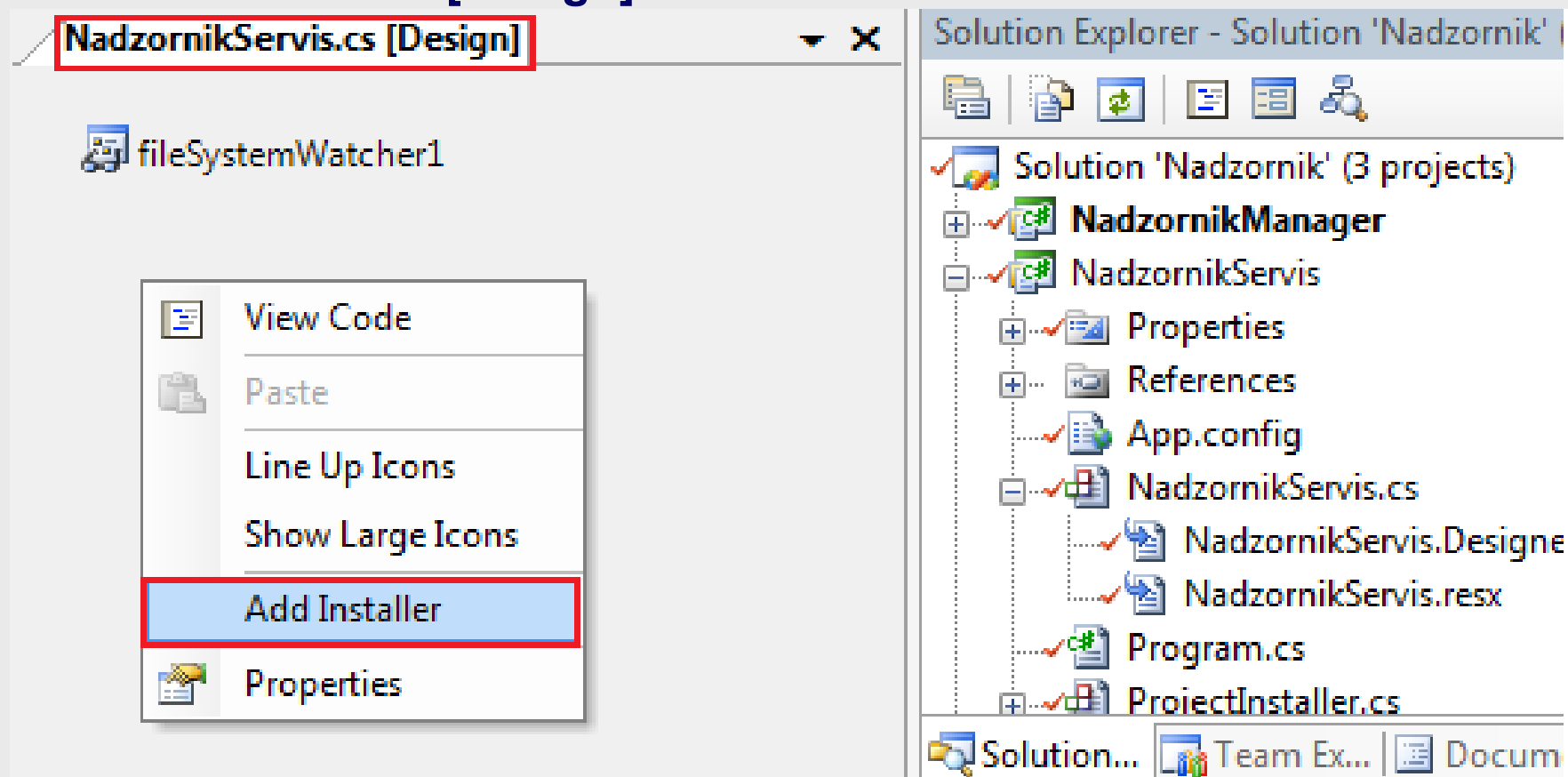
- ❑ Prevedena izvršna datoteka servisa mora biti instalirana da bi servis mogao obavljati željenu funkciju
- ❑ Nužno je izraditi instalacijsku komponentu.
 - Instalacijska komponenta instalira i registrira servis na računalu te stvara ulaznu točku servisa za *Windows Service Control Manager*
- ❑ Servis se u razvojnoj okolini ne može pokretati ili *debugirati* (engl. *debug*) na način uobičajen za interaktivne aplikacije, pritiskom gumba F5 ili F11 (*Visual Studio*)
- ❑ Tek nakon što se servis instalira i pokrene, može se koristiti *VS debbuger* pri čemu se potrebno spojiti na proces servisa:
 - *Tools / Attach to Process*

Izrada instalacijske komponente

❑ Izrada instalacijske komponente

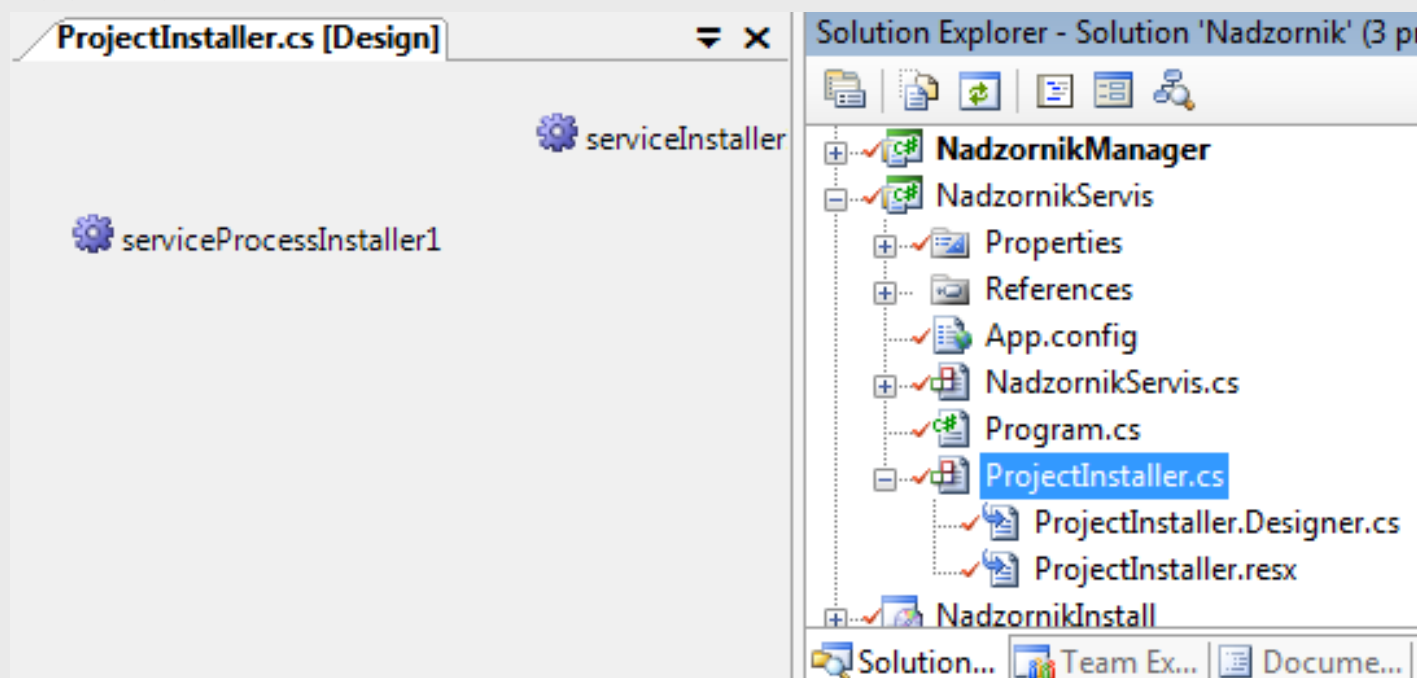
- *Add Installer* funkcija iz skočnog izbornika datoteke NadzornikServis.cs

❑ Primjer: WinServis\NadzornikServis – NadzornikServis[Design].cs



Instalacija servisa – ProjectInstaller.cs

- ❑ **Add Installer** funkcija dodaje u projekt datoteku **ProjectInstaller.cs** koja sadrži dvije komponente:
 - `serviceInstaller`
 - `serviceProcesInstaller`
- ❑ **Automatski je dodana i referenca projekta na `System.Configuration.Install` knjižnicu s navedenim razredima**



Instalacija servisa – ProjectInstaller.cs

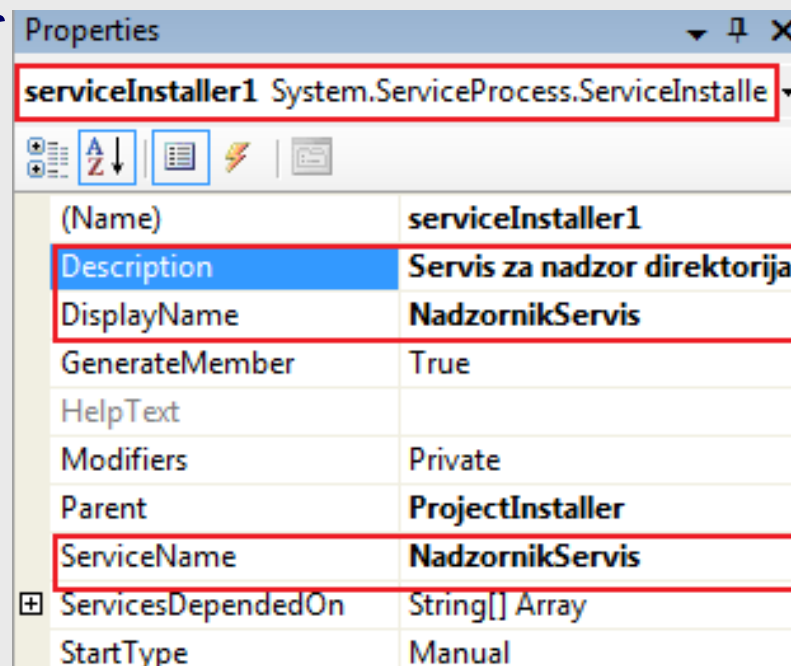
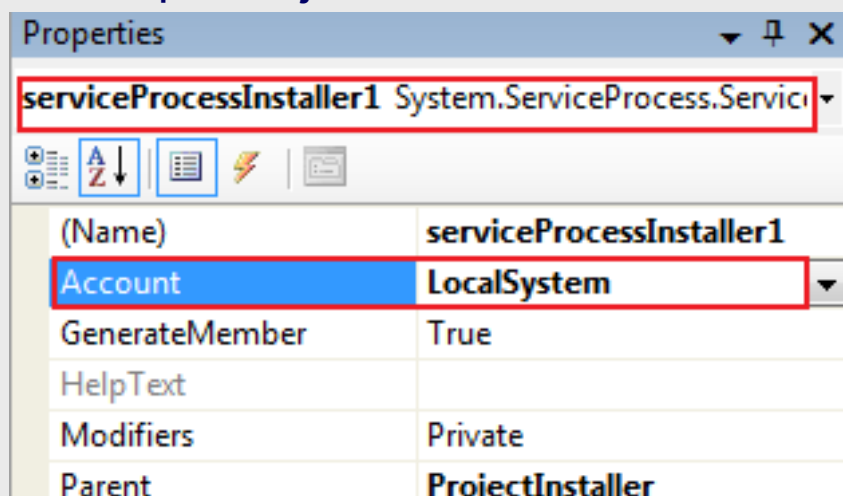
❑ Objekti tipa `serviceInstaller` i `serviceProcessInstaller` pozvani su tijekom instalacije servisa.

❑ Svojstva razreda `serviceInstaller`

- `Description` – opis servisa
- `DisplayName` – ime servisa koje će se prikazati korisniku (*friendly name*)
- `ServiceName` – ime servisa po kojem će OS identificirati servis.
Obavezno mora biti jednako imenu razreda koji je naslijedio `ServiceBase`

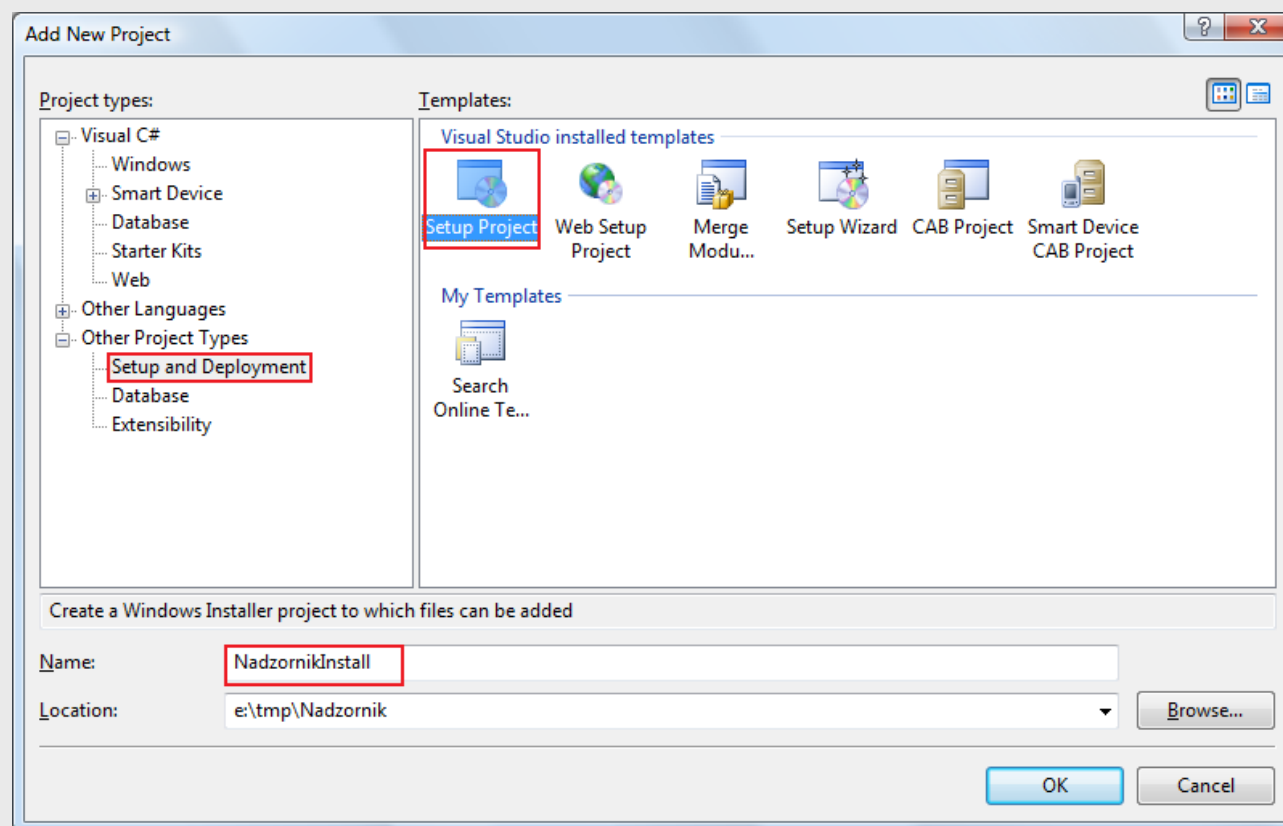
❑ Svojstva razreda `serviceInstaller`

- `Account` – tip korisničkog računa pod kojim se servis izvršava



Instalacija servisa – instalacijski projekt

- ❑ Osim dodavanja instaler datoteke za servis, potrebno je dodati i instalacijski projekt koji kopira izvršne datoteke u instalacijski direktorij te instalira servis na računalo
- ❑ Dodavanje instalacijskog projekta (*Add -> New Project* u kontekstnom izborniku *Solution*):



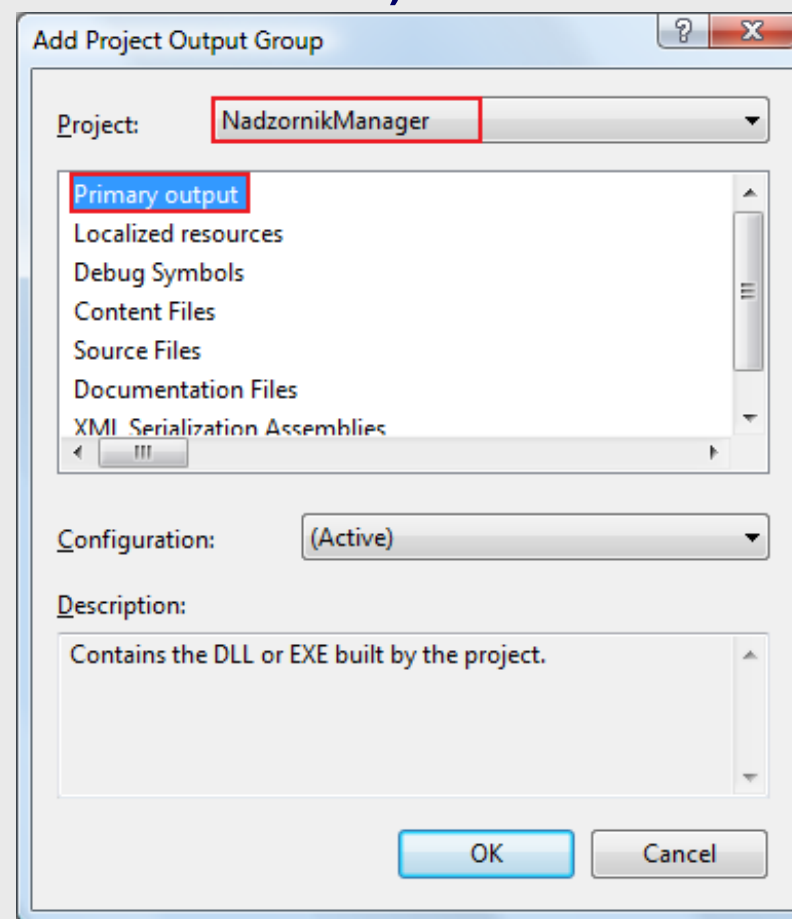
Instalacija servisa – instalacijski projekt

- ❑ U Instalacijskom projektu definiraju se radnje koje se izvršavaju tijekom instalacije kao što su kopiranje datoteka na konačnu lokaciju, postavljanje ikona i prečaca, provjera i instalacija potrebnih aplikacija i knjižnica (npr. .NET Framework).

- ❑ Za instaliranje servisa potrebno je dodati kompiliranu inačicu servisa u instalacijski projekt (**Primary output**):

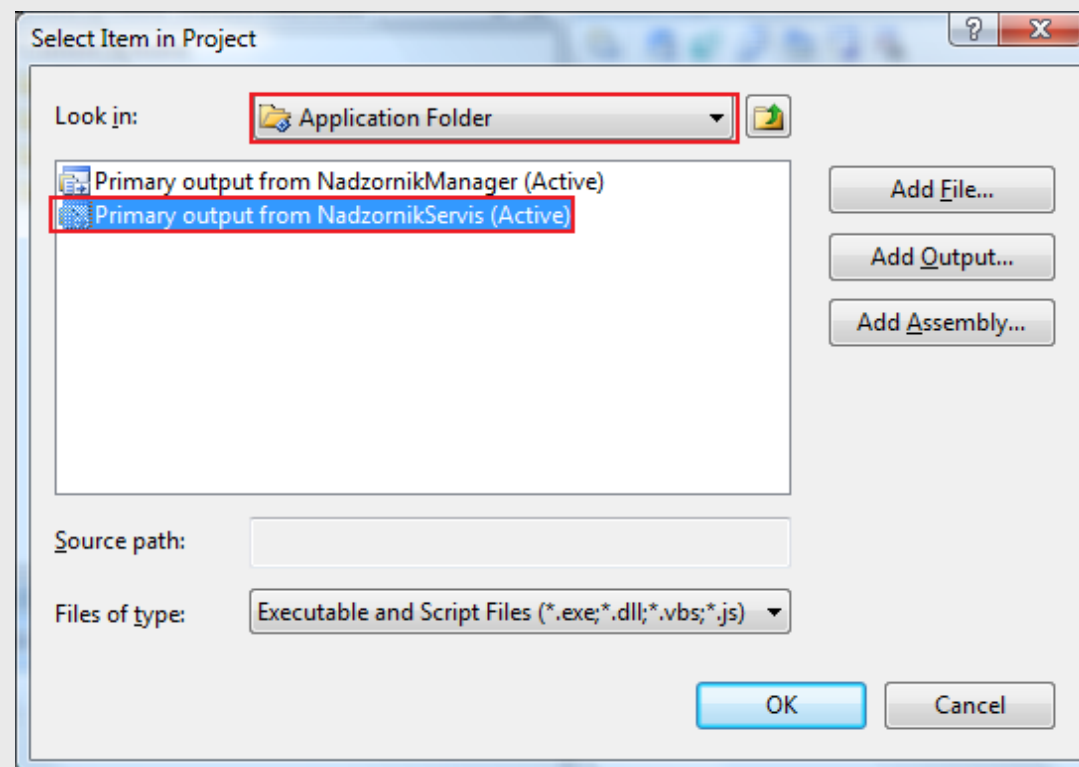
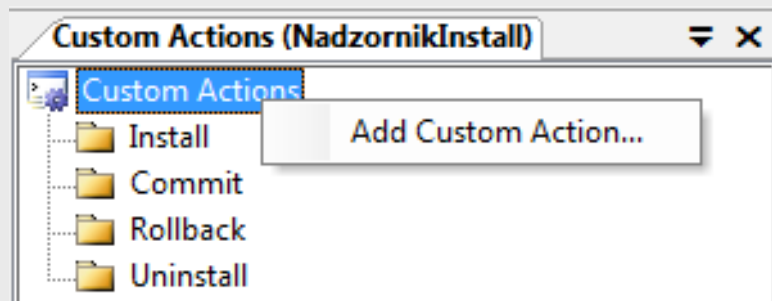
skočni izbornik *NadzornikInstall* projekta (desni klik): *Add -> Project Output*

u dobivenom dijalogu odabere se *NadzornikServis* projekt, odnosno njegov izlaz:



Instalacija servisa – instalacijski projekt

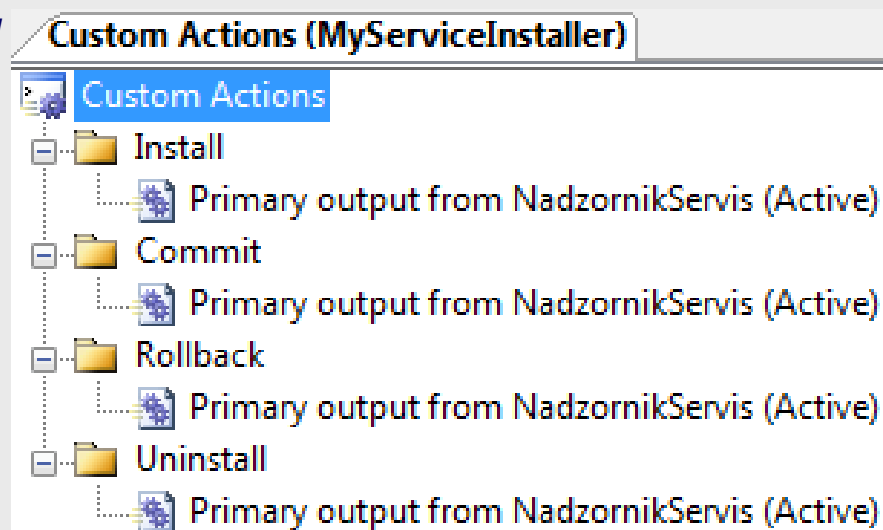
- ❑ Osim dodavanja kompilirane inačice (*Primary output*) potrebno je definirati i korisnički definirane akcije (*Custom actions*) za projekt:
 - kontekstni izbornik *NadzornikInstall* projekta (desni klik): *View -> Custom Actions*
 - u otvorenom dijalogu iz kontekstnog izbornika nad *Custom Actions* odabrati *Add Custom Action*
 - odabrati *Application Folder* te *Primary output* od servisa



Instalacija servisa – instalacijski projekt

❑ **Primary output** dodan je na sve točke korisničkih akcija:

- *Install, Commit, Rollback, Uninstall*



❑ Instalacijski projekt se sada može sagraditi (*build*) te instalirati.

- *Build* i *Install* komande nalaze se u kontekstnom izborniku projekta.

❑ **Napomena:** instalaciju servisa može obaviti samo korisnik s administratorskim dozvolama.

- Ako se instalacija radi pod OS Vista iz razvojne okoline, potrebno je pokrenuti kao administrator (*Run as administrator*).
- Isto vrijedi i za instalacijsku datoteku koja je izlaz instalacijskog projekta.

Izrada aplikacije za upravljanje windows servisom

❑ Primjer:

- Za NadzornikServis servis (iz prošlog primjera) izraditi upravljačku aplikaciju koja upravlja radom servisa (*Start*, *Stop*, *Pause*, *Continue*) te omogućava promjenu nadgledanog direktorija.
- Aplikacija ne treba imati formu već samo ikonu u Windows status traci za upravljanje servisom putem kontekstnog izbornika.

❑ U *Solution* (postojeći, koji već sadrži servis i instalacijski projekt) dodati novi projekt tipa *Windows Application*.

❑ Na formu aplikacije (*Form1.cs*) iz alatne trake (*toolbox*) dovući sljedeće kontrole: `NotifyIcon`, `ContextMenuStrip` i `FolderBrowserDialog`

Razred NotifyIcon

❑ Komponenta koja kreira ikonu programa u Windows status traci

❑ Svojstva

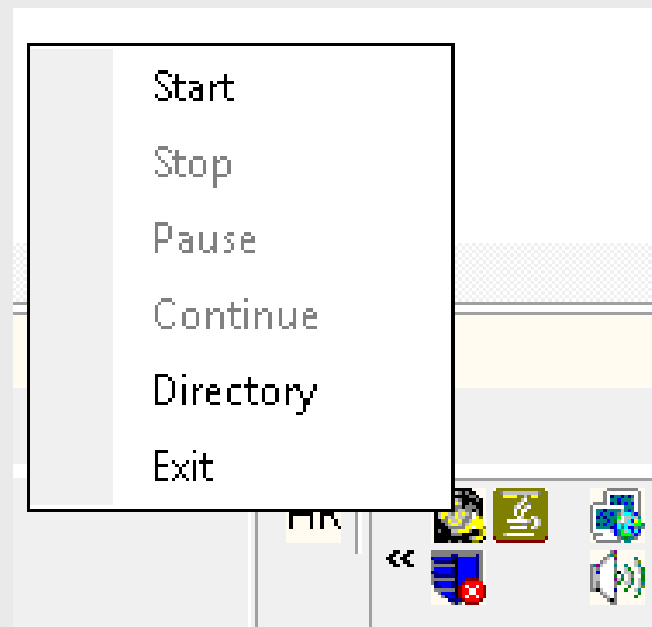
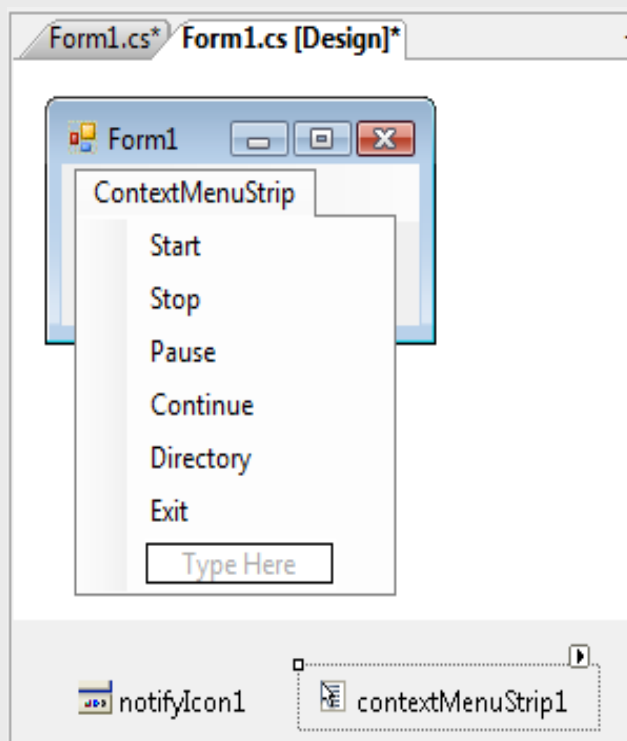
- `ContextMenu` – izbornik koji se prikazuje na desni klik ikone
- `Icon` – ikona komponente
- `Text` – tekst objašnjenja nad ikonom
- `Visible` – oznaka da će ikona biti prikazana
- `BalloonTipIcon` – ikona balončića vezanog uz ikonu
- `BalloonTipText` – tekst unutar balončića
- `BalloonTipTitle` – naslov tekst unutar balončića

❑ Događaji:

- `Click`, `DoubleClick`, `MouseMove`

Povezivanje NotifyIcon i ContextMenuStrip

❑ Primjer: WinServis\NadzornikManager – Form1.cs



❑ Primjer: NadzornikManager \ Form1.cs[Design]

- Povezivanje NotifyIcon kontrole i ContextMenuStrip-a – Svojstvo ContextMenuStrip u NotifyIcon objektu treba postaviti na ContextMenuStrip objekt koji smo dodali na formu

Razred `ServiceController`

❑ `ServiceController` razred omogućava:

- interakciju sa Windows servisima na lokalnom računalu i računalima na koje postoji pristup
 - dohvat dostupnih servisa
 - zadavanje naredbi za pokretanje, zaustavljanje, pauziranje servisa
 - zadavanje korisničkih (*custom*) naredbi

❑ Metode

- `GetServices` – dohvaća servise na računalu (`static` postupak)
- `Refresh` – osvježava vrijednosti svih svojstava
- `Start` – pokreće servis
- `Stop` – zaustavlja
- `Pause` – pauzira
- `Continue` – nastavlja

Primjer ServiceController

- ❑ **Primjer:**  **WinServis\NadzornikManager – Form1.cs**
 - Upotreba ServiceController objekta

```
private ServiceController nadzornikServis = null;

private void CheckServiceInstallation()
{
    ServiceController[] installedServices;
    installedServices = ServiceController.GetServices();
    foreach (ServiceController tmpService in installedServices)
    {
        if (tmpService.DisplayName == "NadzornikServis")
        {
            nadzornikServis = tmpService;
            return;
        }
    }
}
```

Primjer ServiceController

❑ Primjer: WinServis\NadzornikManager – Form1.cs

- Pokretanje procesa (prilikom klika na “Start” u kontekstnom izborniku):

```
private void startToolStripMenuItem_Click(  
    object s, EventArgs e){  
    try  
    {  
        nadzornikServis.Start();  
    }  
    catch (Exception ex) {}  
    finally  
    {  
        UpdateServiceStatus();  
    }  
}
```

Zapisivanje konfiguracije servisa

❏ Primjer: WinServis\NadzornikManager – Form1.cs

```
private void directoryToolStripMenuItem_Click(object sender, EventArgs e)
{
    //mapiranje na konfiguracijsku datoteku drugog programa
    ExeConfigurationFileMap fileMap = new ExeConfigurationFileMap();
    fileMap.ExeConfigFilename = @"NadzornikServis.exe.config";
    System.Configuration.Configuration config =
        ConfigurationManager.OpenMappedExeConfiguration(fileMap,
            ConfigurationUserLevel.None);
    folderBrowserDialog1.SelectedPath =
        config.AppSettings.Settings["Path"].Value.ToString();
    folderBrowserDialog1.ShowDialog();
    config.AppSettings.Settings["Path"].Value=
        folderBrowserDialog1.SelectedPath.ToString();
    config.Save();
    if(nadzornikServis.Status==ServiceControllerStatus.Running)
        nadzornikServis.ExecuteCommand(128);
}
```

Zadaci za vježbu

- ❑ **Implementirati windows servis koji u zadanim vremenskim intervalima provjerava dnevnik OS-a:**
 - ukoliko su se u sistemskom logu, od zadnje provjere, pojavili zapisi o pogreškama servis treba poslati e-mail sa izvještajem o broju grešaka na zadane adrese.
 - e-mail adrese i vremenski interval zadaju se preko konfiguracijske datoteke servisa.

Reference

❑ MSDN dokumentacija o windows servisima

- [http://msdn2.microsoft.com/en-us/library/y817hyb6\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/y817hyb6(VS.80).aspx)

❑ Wikipedija

- http://en.wikipedia.org/wiki/Windows_service