

Projekt sustava, Dizajn sustava (Systems Design)

8/13

Opći dizajn

- ❑ **Opći dizajn (konceptualni, visoke razine, arhitekturni) → funkcionalne specifikacije**
- ❑ **Odabir tehničke arhitekture sustava**
 - Centralizirana ili distribuirana obrada i pohrana? Kako? Tehnologije?
 - Softver: nabavljeni, napravljen po mjeri, mješavina? Razvojni alati?
- ❑ **Analiza i distribucija podataka**
 - pretvorba konceptualnog modela podataka u logički model (relacijski, postrelacijski, objektno-relacijski), ako nije učinjena ranije
- ❑ **Analiza i distribucija procesa**
 - pretvorba logičkog modela procesa u fizički model za odabranu arhitekturu
→ shema aplikacije
- ❑ **Opći dizajn sučelja**
 - izgled, ergonomija, tzv. 'look and feel'

Detaljni dizajn

❑ Detaljni dizajn, dizajn na nižoj razini → tehničke specifikacije

❑ Izrada fizičkog modela podataka

- pretvorba logičkog modela podataka u fizički model podataka za odabrani SUBP → shema baze podataka
 - prilagodba modela mogućnostima i ograničenjima SUBP
 - fizički parametri (volumetrija) i ugađanje baze podataka (indeksi)
- oblikovanje fizičkih datoteka

❑ Dizajn programa

- utvrđivanje strukture programa na temelju modela procesa
 - (logički) proces ili skup procesa ↔ jedan ili više modula/razreda
- preciziranje programske logike

❑ Dizajn sučelja

- dizajn sučelja sustava – protokoli pristupa i razmjene podataka
- oblikovanje zaslonskih maski i izvješća

Dokumentiranje dizajna

- ❑ **Dizajn programske podrške (software design)**
- ❑ **Dizajn programa (program design)**
 - proces pretvorbe zahtjeva na programsku podršku u oblik koji omogućuje programiranje
 - opis jezikom za projektiranje programa (PDL - Program Design Language) pri čemu program napisan pomoću PDL nema oblik izvedbenog programa
- ❑ **Primjer:  \Prilozi\SpecifikacijaDizajna.dot**
 - predložak specifikacije
- ❑ **Primjer:  \Prilozi\Firma-Dizajn.doc**
 - primjer specifikacije

Arhitektura sustava

Dizajn arhitekture sustava

□ Dizajn arhitekture

- sastoji se od planova koji definiraju pojedine komponente sustava
 - računalnu opremu
 - programsku podršku
 - komunikacije
 - sustav zaštite
 - globalnu podršku aplikacije
- Specifikacija hardvera i softvera → podloga za nabavu

□ Uobičajeni modeli arhitekture

- poslužiteljska (server-based) – obrada se obavlja na poslužitelju
- klijentska (client-based) – obrada se obavlja na osobnom računalu
- klijent-poslužitelj (client-server based) – kombinacija prethodne dvije

□ Model mreže

- prikaz glavnih komponenti sustava, njihove fizičke lokacije i način njihovog međusobnog povezivanja

□ Specifikacije računalne opreme i programske podrške

- elementi za nabavu ili izradu

Elementi arhitekture sustava

❑ Funkcije sustava = slojevi arhitekture

- Pohrana podataka (data storage) – baza podataka
- Pristup podacima (data access logic) – npr. ADO.NET, INFORMIX.NET, ...
- Elementi obrade (application logic) – aplikacijski program, pohranjene proc.
- Sučelje (presentation logic) – zaslonske maske

❑ Poslužitelji

- Velika računala (Mainframe)
- Mala računala (Minicomputer)
- Mikroračunala (Microcomputer) – PC

❑ Klijenti

- Klasični terminali - ansi, vt220, IBM 3270, ...
- Mikroračunala – PC
 - pristup emulatorima terminala ili udaljenim radnim ploham (RDC)
- Terminali posebne namjene
 - bankovni terminali (bankomati), Internet kiosk, ručna računala, ...

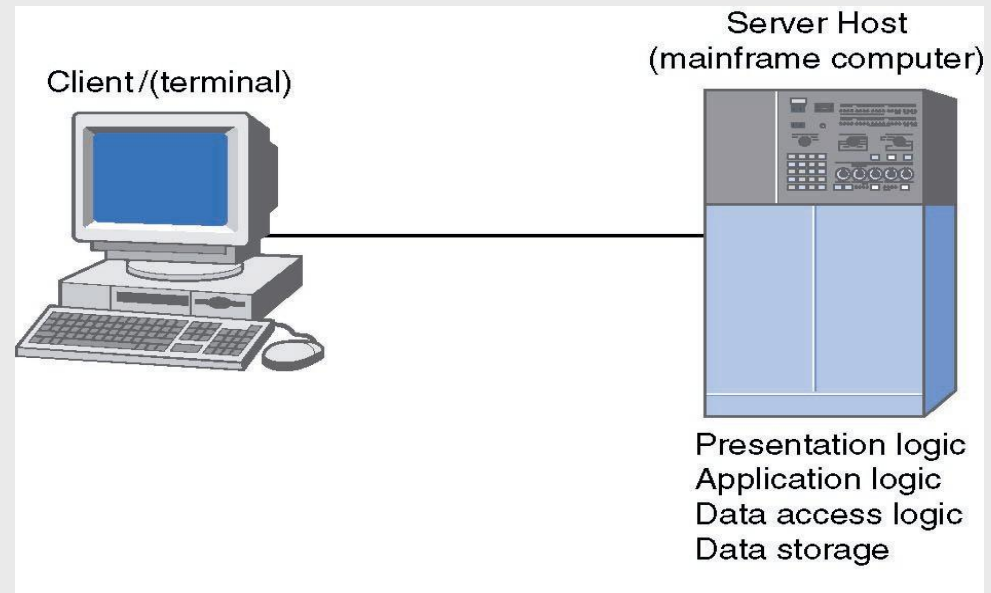
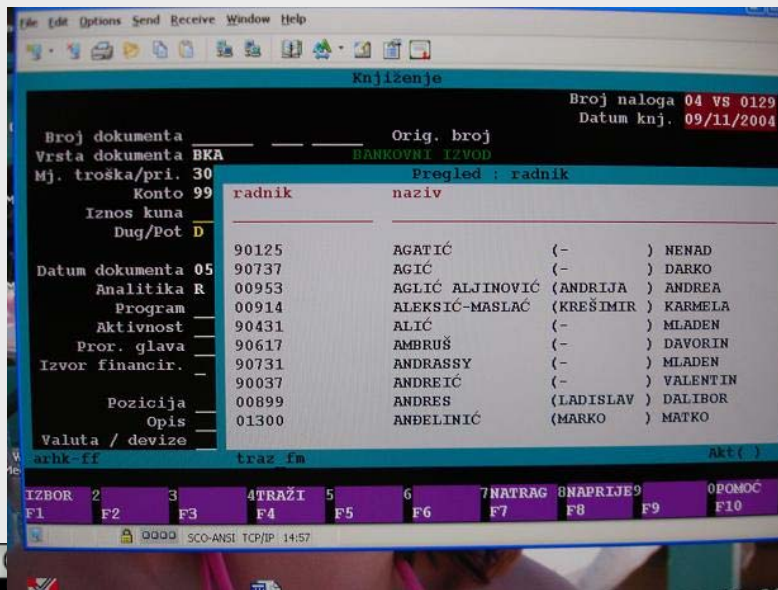
Centralizirana obrada

❑ Višekorisnički poslužitelj + terminali

- pohrana podataka - datoteke i baze podataka
- poslovna logika - programska podrška
- korisničko sučelje - uobičajeno znakovno sučelje
- sučelje sustava - mrežne i druge komponente

❑ Distribuirana prezentacija

- nadgradnja zamjenom znakovnog sučelja grafičkim, koje se izvodi na PC
- produljuje vijek aplikacija, ali se funkcionalnost ne može značajno poboljšati



Dvoslojna arhitektura klijent-poslužitelj (client-server)

❑ Klijent - jednokorisničko računalo

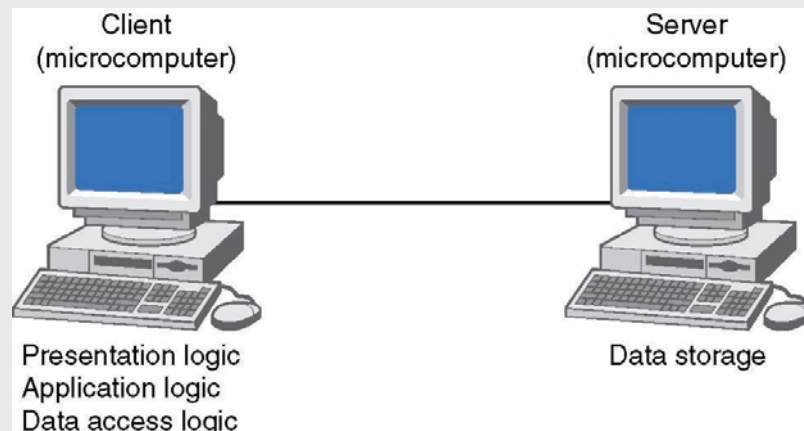
- sučelje, obrada i pohrana
- povezljivost na poslužitelje i druge klijente

❑ Poslužitelj - višekorisničko računalo

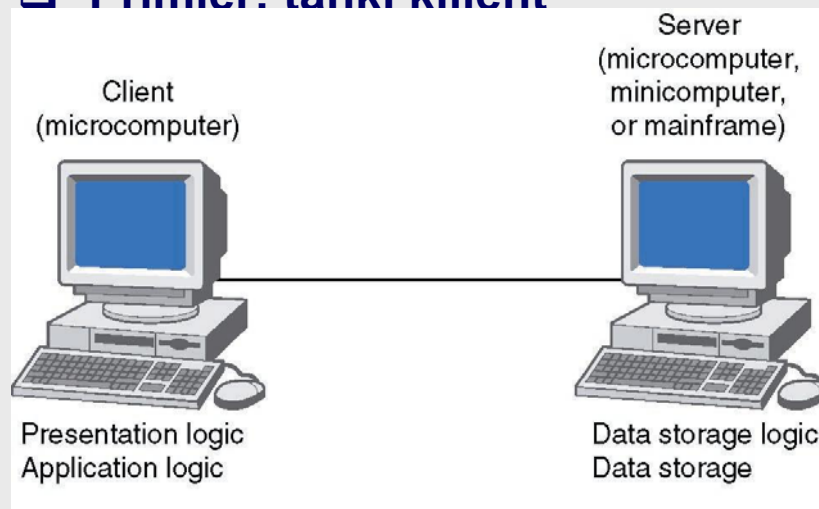
- baza podataka, obrada i servisi sučelja
- povezljivost s klijentima i poslužiteljima

❑ Korisnicima izgleda kao da njihovo računalo (PC) obavlja cijeli posao

❑ Primjer: debeli klijent



❑ Primjer: tanki klijent



Debeli klijent

☐ **Debeli klijent, u novije vrijeme “bogat” klijent (rich client)**

- Podatkovna logika integrirana u klijenta
- Nema obrade podataka na poslužitelju ili je obrada minimalna
- Klijent može imati lokalno spremište (bazu) podataka
- Minimalna ili nikakva elastičnost na promjene poslovne politike

☐ **Prednosti**

- brzi početni razvoj aplikacije
- veća samostalnost klijenta
- rasterećenje glavnog računala (poslužitelja)

☐ **Nedostaci**

- poslovna logika integrirana u klijenta
- promjena logike zahtijeva instaliranje nove verzije na svim klijentima
- razvoj velike aplikacije s vremenom postaje vrlo složen
- potreban veći broj klijentskih računala dovoljne procesne moći

Tanki klijent

❑ Tanki klijent (thin client)

- Podatkovna logika se nalazi na poslužitelju
- Osnovna namjena klijenta je prikaz podataka
- Tipičan primjer tankog klijenta je web preglednik

❑ Prednosti

- manja složenost razvoja velikih aplikacija (serverski dio i klijentski dio)
- olakšana distribucija, kao tanki klijent može se koristiti npr. općenito dostupan web preglednik
- lakše održavanje – centralizirana promjena poslovne logike
- klijentska računala ne moraju imati veliku moć obrade

❑ Nedostaci

- veliko opterećenje glavnog računala, a to znači skupo glavno računalo
- veće mrežno opterećenje (gotovo za svaku promjenu ide se na server)
- lošija funkcionalnost kada se kao klijent koristi web preglednik
- naglo povećanje složenosti zahtjevnog grafičkog sučelja (skriptiranjem)

Troslojna ili višeslojna arhitektura klijent-poslužitelj

❑ Distribucija baza podataka i poslovne logike na zasebne poslužitelje

- poslužitelj aplikacija + poslužitelj baza podataka + klijent
- poslužitelj baza podataka
 - upravljanje podacima
- poslužitelj aplikacija
 - upravljanje transakcijama, "preuzeto" s podatkovnog poslužitelja
 - dio ili čitava poslovna logika, "preuzeta" s klijenta
- klijent
 - korisničko sučelje
 - dio poslovne logike - onaj koji se ne mijenja ili je osobnog karaktera

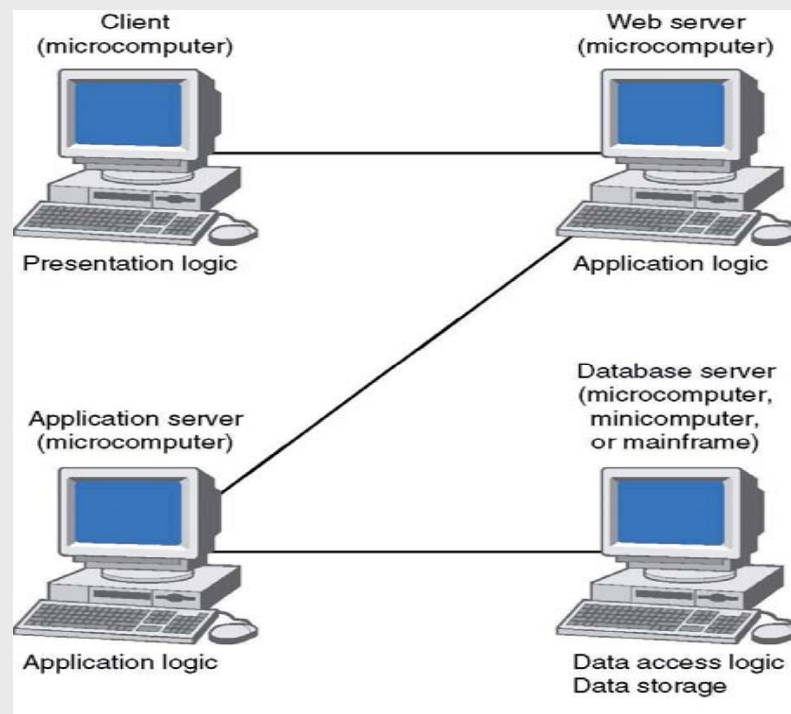
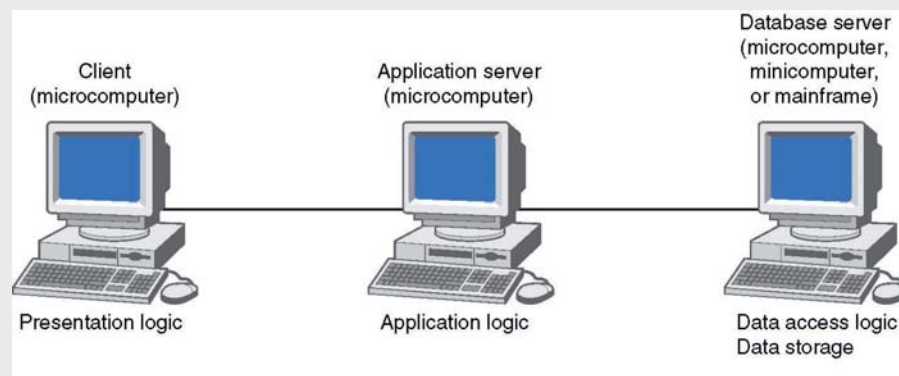
❑ Prednosti

- bolja raspodjela opterećenja
- veća skalabilnost - mogućnost ekspanzije, npr. povećanja broja korisnika, bez preopterećenja ili promjene procedura

❑ Nedostaci

- vrlo složen dizajn i razvoj
- problem raspodjele podataka, procesa, sučelja
- veće opterećenje mreže

❑ Primjeri



Primjer arhitektura klijent-poslužitelj

❑ Napraviti aplikaciju koja će

- prikazivati listu najboljih 10 partnera poredanih po iznosu prometa
- za odabranog partnera prikazati njegove dokumente
- problem oblikovanja dinamičkog skupa podataka i pisanja upita
 - potpuna informacija o partneru zahtijeva uniju zbog specijalizacije
 - kako dinamički povezati uniju i dokumente ?

❑ Primjer: Arhitecture\DebeliKlijent

- skup podataka najboljih partnera, adapter s odgovarajućim SQL upitom
- dinamička selekcija dokumenata za odabranog partnera, SQL upitom ugrađenim u programski kod programa

❑ Primjer: Arhitecture\TankiKlijent

- poziv pohranjene procedure koja vraća skup najboljih partnera
- poziv pohranjene procedure za dohvat dokumenata određenog partnera

Primjer debelog klijenta

❑ Primjer: Arhitekture\DebeliKlijent

- skup podataka partnera definiran u dizajnu DataAdapter komponente
- povezivanje kontrole (dataGridViewPromet) u dizajnu

```
SELECT TOP (10) Dokument.IdPartnera, PoslovniPartner.Naziv,  
    PoslovniPartner.Broj, SUM(Dokument.IznosDokumenta) AS Promet  
FROM Dokument INNER JOIN  
    (SELECT IdOsobe AS IdPartnera,  
        PrezimeOsobe + ' ' + ImeOsobe AS Naziv, JMBG AS Broj  
    FROM Osoba  
    UNION  
    SELECT IdTvrtke AS IdPartnera, NazivTvrtke AS Naziv,  
        MatBrTvrtke AS Broj  
    FROM Tvrtka) AS PoslovniPartner  
ON Dokument.IdPartnera = PoslovniPartner.IdPartnera  
GROUP BY Dokument.IdPartnera, PoslovniPartner.Naziv,  
    PoslovniPartner.Broj  
ORDER BY Promet DESC
```

Primjer debelog klijenta – selekcija detalja

❑ Primjer: Arhitekture\DebeliKlijent

- kreiranje konekcije, adaptera i skupa podataka te povezivanje kontrole za prikaz – dinamički, u programskom kodu

```
private void dataGridViewPromet_CurrentCellChanged(...) {  
    ...  
    int idPartnera =  
        (int)dataGridViewPromet.CurrentRow.Cells[0].Value;  
  
    string selectDokument = @"  
        SELECT IdDokumenta, VrDokumenta, BrDokumenta, ...  
        FROM Dokument  
        WHERE Dokument.IdPartnera = " + idPartnera.ToString();  
  
    SqlConnection conDokument = new SqlConnection(...);  
    conDokument.Open();  
    SqlDataAdapter sqlDataAdapterDokument = new SqlDataAdapter();  
    DataSet dataSetDokument = new DataSet("DokumentPartnera");  
    sqlDataAdapterDokument.Fill(dataSetDokument);  
    dataGridViewDokument.DataSource = dataSetDokument.Tables[0];  
}
```

Primjer tankog klijenta

❑ Primjer: Arhitekture\TankiKlijent

- poziv pohranjenih procedura za skup partnera i dokumente partnera
- dinamičko kreiranje skupova i povezivanje na podatke

```
FormTanki_Load () {  
    SqlDataAdapter sqlDataAdapterPromet =  
        new SqlDataAdapter("ap_PrometPartnera", sqlConnection1);  
    DataSet dataSetPromet = new DataSet("PrometPartnera");  
    sqlDataAdapterPromet.Fill(dataSetPromet);  
    dataGridViewPromet.DataSource = dataSetPromet.Tables[0];  
}
```

```
private void dataGridViewPromet_CurrentCellChanged(...) {  
    ...  
    int idPartnera =  
        (int)dataGridViewPromet.CurrentRow.Cells[0].Value;  
  
    SqlDataAdapter sqlDataAdapterDokument = new SqlDataAdapter(  
        "ap_DokumentPartnera " + idPartnera.ToString(), conDokument);  
    ...  
}
```


Tanki klijent – pogledi i pohranjene procedure

❏ Primjer: Arhitekture\TankiKlijent – TankiKlijent.sql

```
CREATE PROCEDURE [dbo].[ap_PrometPartnera] AS
    SELECT * FROM vw_PrometPartnera ORDER BY Promet DESC
...
CREATE VIEW [dbo].[vw_PrometPartnera] AS
    SELECT ... FROM dbo.vw_Promet INNER JOIN dbo.vw_Partner
...
CREATE VIEW [dbo].[vw_Promet] AS
    SELECT TOP (10) IdPartnera, SUM(IznosDokumenta) AS Promet
    FROM    dbo.Dokument
    GROUP BY IdPartnera
    ORDER BY Promet DESC
...
CREATE VIEW [dbo].[vw_Partner] AS
    SELECT ... FROM Osoba
    UNION
    SELECT ... FROM Partner
...
```

Primjer promjene zahtjeva

❑ Promjena zahtjeva

- Korisnik se, naravno, predomislio i želi pregled 20 najboljih partnera
- Također, zahtijeva prikaz postotka poreza u prikazu dokumenata.

❑ Na debelom klijentu treba

- promijeniti SQL upite u izvornom kodu
 - SELECT naredbu adaptera,
 - postupak za obradu događaja pri selekciji partnera
- prevesti ga u novu izvršnu inačicu te
- instalirati aplikaciju pojedinom korisniku
- zahtjevno, sporo i skupo

❑ Na tankom klijentu treba

- na poslužitelju baze podataka promijeniti
 - pogled za selekciju partnera *vw_Promet*
 - pohranjenu proceduru *ap_DokumentPartnera*
- centralizirano, jednostavno, brzo i jeftinije

Višeslojna aplikacija

8/13

Višeslojna aplikacija

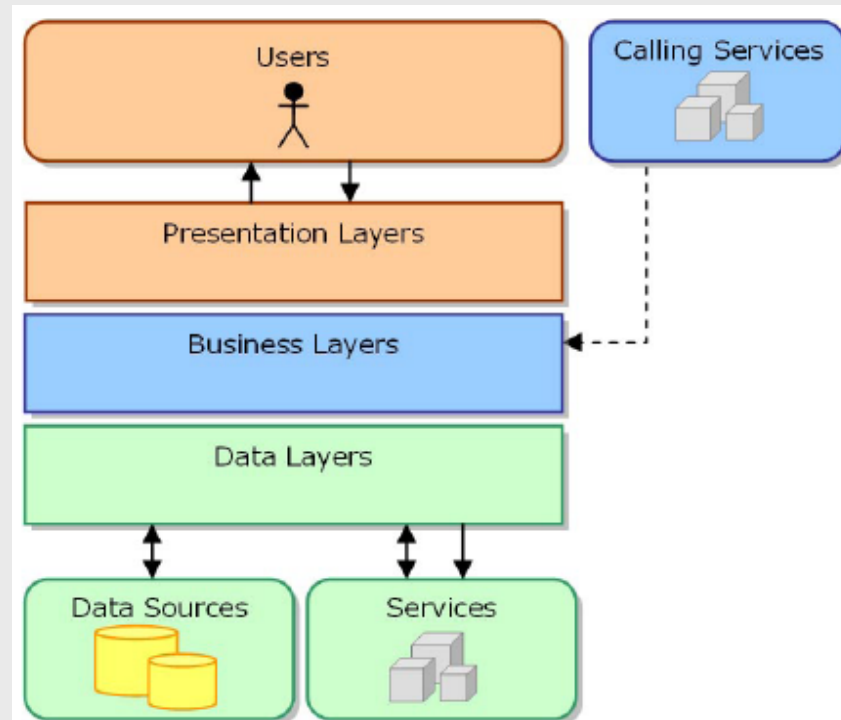
□ Aplikacija se može podijeliti u više razina – slojeva, npr.

- prezentacijski sloj (PL – Presentation Layer)
 - grafičko sučelje (GUI - Graphical User Interface) – Win / Web / PPC
- poslovni sloj (BL – Business Layer)
 - sloj poslovne logike (BLL - Business Logic Layer) – poslovne klase
- podatkovni sloj (Data Layer) –
 - sloj pristupa podacima (DAL - Data Access Layer) – npr. ADO.NET
- pohrana podataka (Data Storage) –
 - + kod na BP (stored procedure)
 - + pogledi na BP (view)

□ Između slojeva kolaju

- poslovni objekti (BE – Business Entity)

□ Primjer: FirmaWin

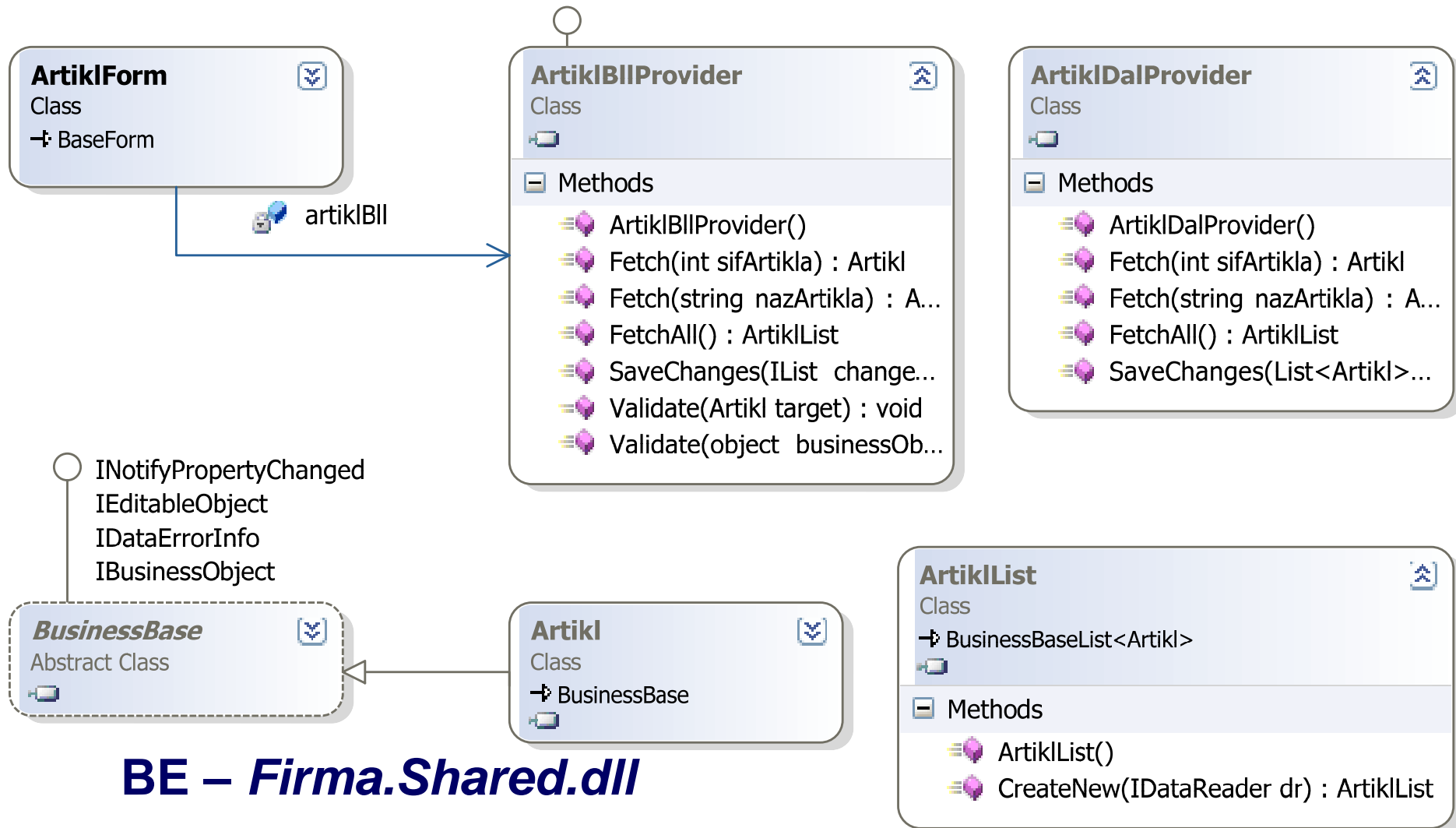


Slojevi aplikacije

PL - *Firma.exe*

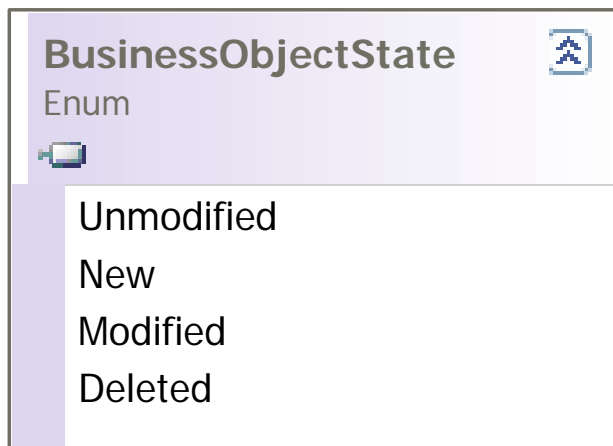
BL - *Firma.BLL.dll*

DL - *Firma.DAL.dll*

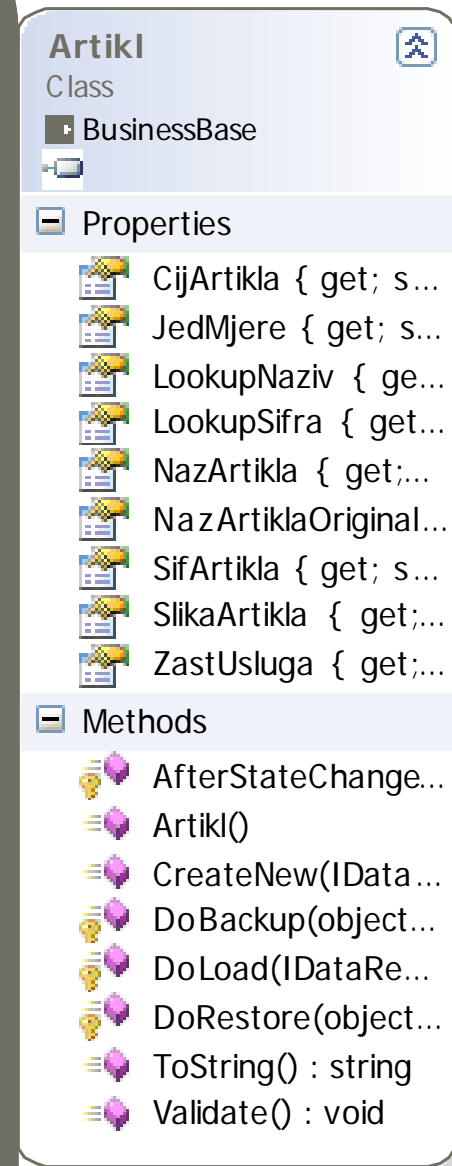
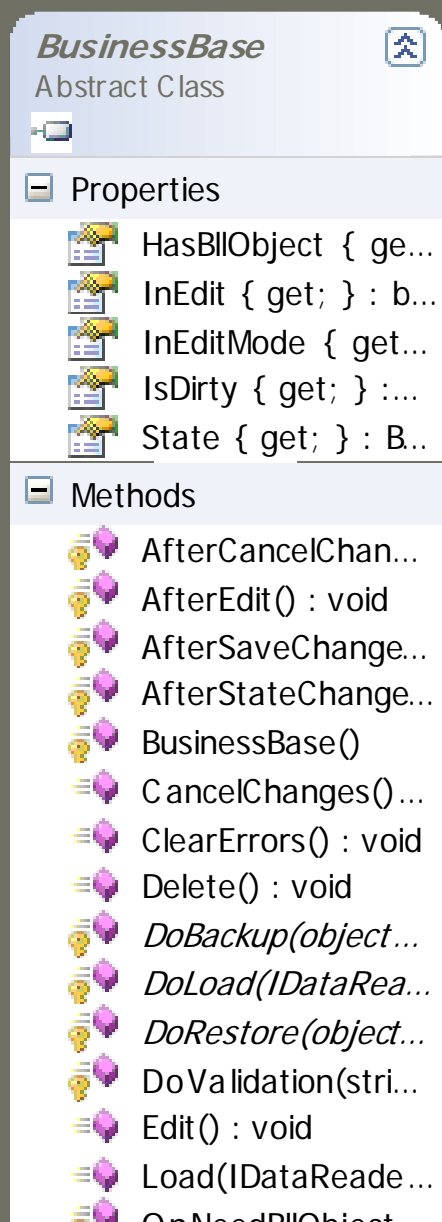


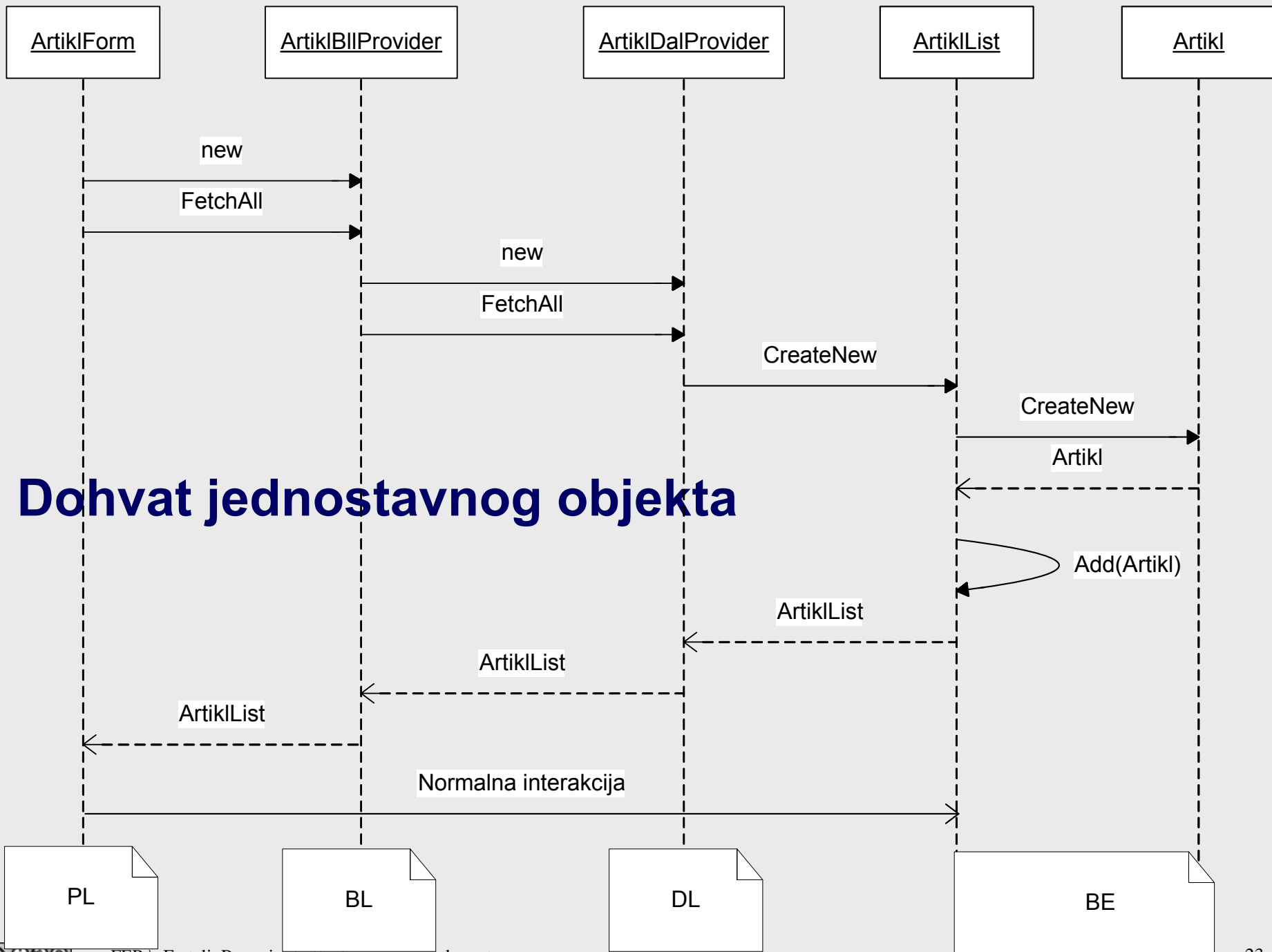
Poslovni objekt

- ❑ Posjeduje svojstva poslovnog entiteta
- ❑ Nosi stanja poslovnog entiteta
- ❑ Implementira postupke za provjeru i promjenu stanja, validaciju u srednjem sloju, pohranu, osvježavanje, itd.



INotifyPropertyChanged
IEditableObject
IDataErrorInfo
IBusinessObject





Zahtjev poslovnom sloju

❑ Primjer: FirmaWin - ArtiklForm

```
// referenca na BLL sloj
private ArtiklBllProvider artiklBll = new ArtiklBllProvider();

public ArtiklForm() {
    InitializeComponent();

    // Dohvat svih Artikala
    artiklBindingSource.DataSource = artiklBll.FetchAll();
}
```

❑ Primjer: FirmaWin - ArtiklBllProvider

```
public class ArtiklBllProvider : IBllObject {
    private ArtiklDalProvider dal = new ArtiklDalProvider();

    public ArtiklList FetchAll() {
        return dal.FetchAll();
    }
}
```


Kreiranje liste objekata

❑ **Primjer:**  **FirmaWin – ArtiklDalProvider ! FetchAll vraća listu ArtiklList**

```
public ArtiklList FetchAll()  
    using (SqlCommand cmd = db.CreateCommand()) {  
        SqlCommand cmd = db.CreateCommand()  
        cmd.CommandText = "[dbo].[ap_ArtiklList_R]";  
        cmd.CommandType = CommandType.StoredProcedure;  
        db.Open();  
        using (SqlDataReader dr = cmd.ExecuteReader()) {  
            return ArtiklList.CreateNew(dr);  
        }  
    }
```

❑ **Primjer:**  **FirmaWin – ArtiklList**

```
public static ArtiklList CreateNew(IDataReader dr)  
{  
    ArtiklList rez = new ArtiklList();  
    while (dr.Read())  
        // kreira i dodaje BE u listu  
        rez.Add(Artikl.CreateNew(dr));  
    return rez;  
}
```

Stvaranje pojedinačnog objekta

❑ Primjer: FirmaWin – Artikl

```
public static Artikl CreateNew(IDataReader dr)
{
    Artikl rez = new Artikl();
    rez.Load(dr);
    return rez;
}
```

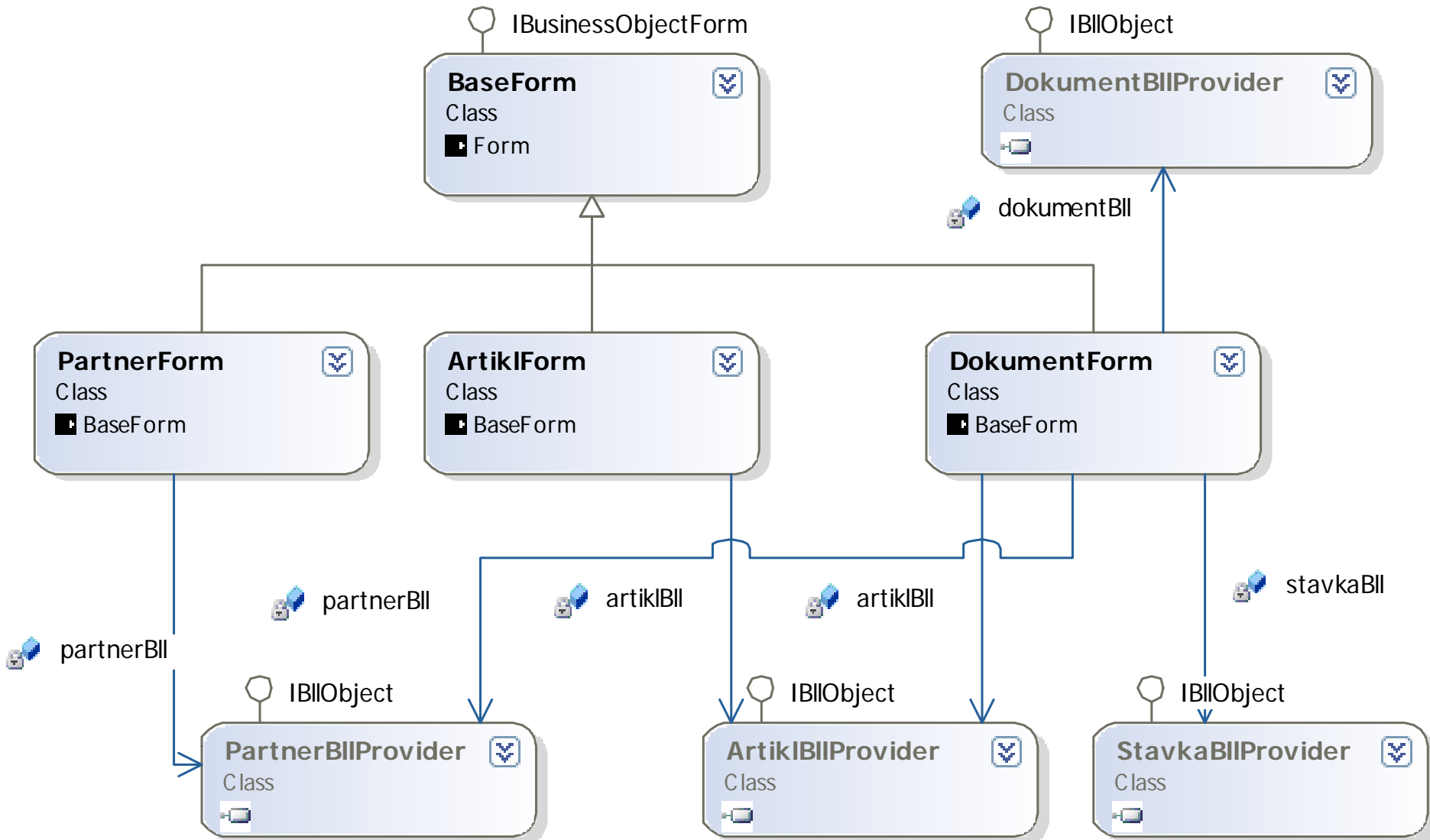
❑ ... nakon jednog međukoraka u *BusinessBase* ...

```
protected override void DoLoad(IDataReader dr)
{
    this.sifArtikla = dr["SifArtikla"] == DBNull.Value
        ? (int?)null : (int?)dr["SifArtikla"];
    this.nazArtikla = dr["NazArtikla"] == DBNull.Value
        ? string.Empty : (string)dr["NazArtikla"];
    ...
}
```

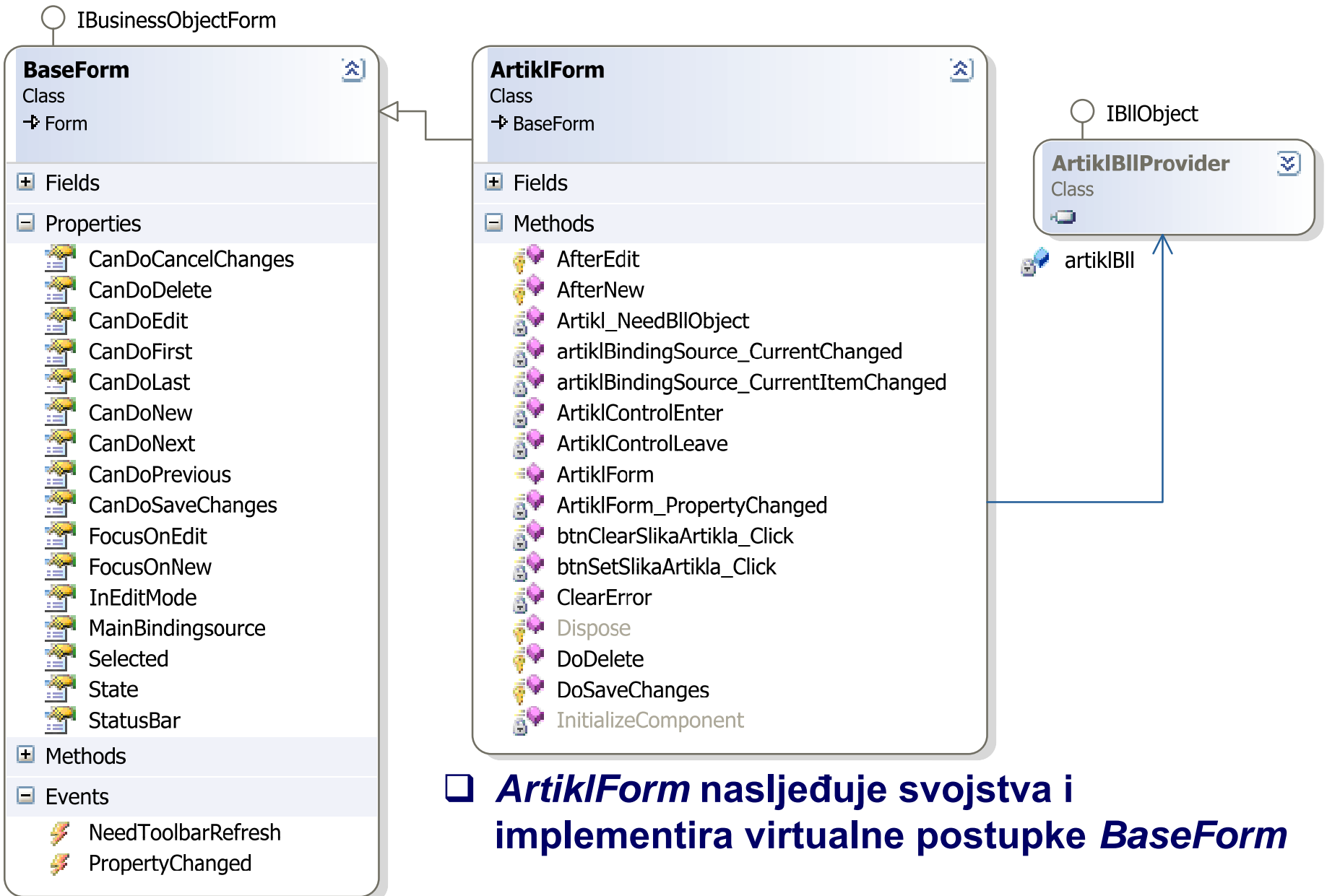
Prezentacijski sloj

Prezentacijski sloj

□ Razredi sloja i veza na poslovni sloj



Zaslonska maska



Povezivanje na objekte

❑ Primjer: FirmaWin – ArtikelForm

- *Data – Add New Data Source – Object ... Firma.Artikl*

❑ Povezivanje u dizajnu

- *artiklBindingSource.DataSource = Firma.Artikl*
- *nazArtiklaTextBox.Text* povezan na "*artiklBindingSource – NazArtikla*"

❑ Povezivanje programski - radi formata prikaza

```
cijArtiklaTextBox.DataBindings.Add(  
    new Binding(  
        "Text", artiklBindingSource, "CijArtikla",  
        true,    // formatiranje omogućeno  
        DataSourceUpdateMode.OnPropertyChanged, // ažuriranje izvora  
        string.Empty, "N2")); // vrijednost za null, dvije decimale
```

Rukovanje povezanim podacima (pogledati Oblikovanje korisničkog sučelja i dijaloga)

❑ Primjer: FirmaWin – BaseForm

- osnovna forma ima referencu na *BindingSource* kako bi rukovala podacima, što izvedena forma izloži u popisu svojstava u dizajnu
- (o atributima više naknadno)

```
private BindingSource mainBindingSource;  
  
[Browsable(true), Category("Data")]  
public BindingSource MainBindingsource  
{  
    get { return mainBindingSource; }  
    set { mainBindingSource = value; }  
}
```

```
public object Selected {  
    get {  
        if (mainBindingSource != null)  
            return mainBindingSource.Current;  
  
        return null;  
    }  
}
```

Navigacija

❑ Primjer: FirmaWin – BaseForm

```
public bool CanDoNext
{
    get { return mainBindingSource != null && !InEditMode
&& mainBindingSource.Position < mainBindingSource.Count - 1; }
}
```

❑ BaseForm.BaseForm_KeyDown

❑ ... case Keys.PageDown: Next();

```
public virtual void Next()
{
    // Odlazak na drugi zapis moguć je ako ne traje unos/izmjena
    if (mainBindingSource != null && !InEditMode)
    {
        mainBindingSource.MoveNext();
        OnNeedToolBarRefresh();
    }
}
```


Prijenos kontrole nad podacima

❑ Primjer: FirmaWin – FormToolbar

```
// Forma s kojom komunicira toolbar.  
// Property se postavlja u dizajnu forme.  
private IBusinessObjectForm f;
```

```
public void RefreshToolbar()  
{  
    this.Enabled = f != null;  
    ..  
    btnNext.Enabled = f != null ? f.CanDoNext : false;
```

```
private void btnNext_Click(object sender, EventArgs e)  
{  
    if (f != null)  
        f.Next();  
}
```

Prikaz informacije o aktualnom zapisu

❑ Postavljanje naslova forme kad se promijeni aktualni artikl

```
private void artiklBindingSource_CurrentItemChanged(  
    object sender, EventArgs e)  
{  
    this.Text = "Artikl";  
    if (artiklBindingSource.Current != null)  
    {  
        this.Text = "Artikl: "  
            + ((Artikl)artiklBindingSource.Current).ToString();  
    }  
}
```

❑ Izazove promjenu statusne trake

```
private void BaseForm_TextChanged(object sender, EventArgs e)  
{  
    ...  
    StatusBar.NazivModula = this.Text;  
}
```

Postavljanje reference na poslovni sloj

❑ Pridruživanje metode za BLL objekt poslovnem objektu

```
private void artikelBindingSource_CurrentChanged(...)  
{  
    Artikel a = artikelBindingSource.Current as Artikel;  
  
    // objekt koji još ne zna gdje mu je BLL objekt... ažuriraj.  
    if (a != null && !a.HasBllObject)  
    {  
        a.NeedBllObject  
        += new NeedBllObjectEventHandler(Artikel_NeedBllObject);  
    }  
}
```

❑ Metoda preko koje će poslovni objekt doći do BL sloja

```
private IBllObject Artikel_NeedBllObject()  
{  
    return artikelBll; // pogledati CallStack pri editu naziva  
}
```

Nalog poslovnom sloju za spremanje i brisanje

❏ Primjer: FirmaWin – ArtikelForm

```
// Spremanje svih izmjena
protected override void DoSaveChanges()
{
    artikelB11.SaveChanges (
        ((ArtiklList)artikelBindingSource.DataSource).GetChanges());
}
```

```
protected override void DoDelete() {
    // Uklanjanje poslovnog objekta iz liste dohvaćenih objekata
    artikelBindingSource.RemoveCurrent(); // označavanje objekta

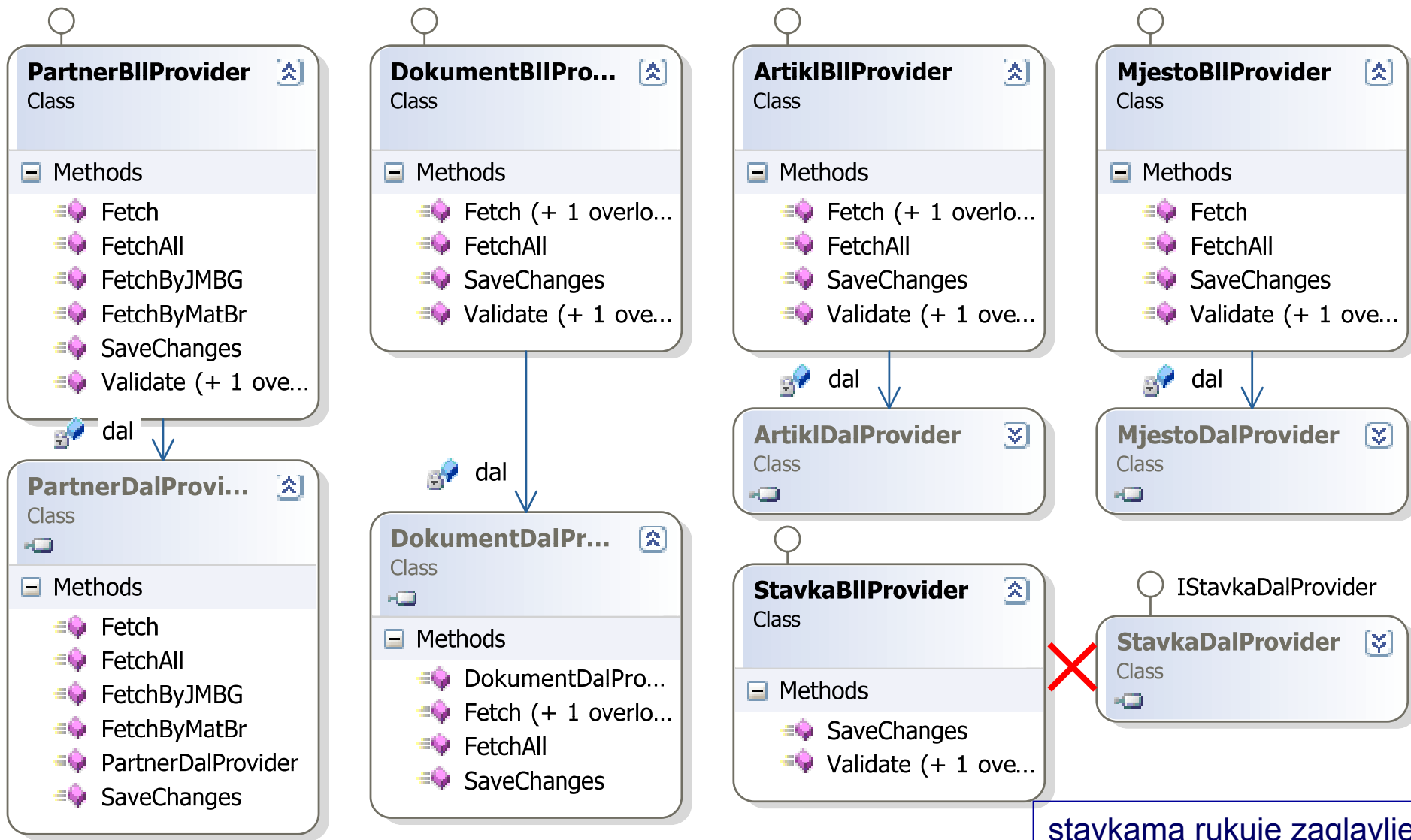
    // Sprema izmjene u bazu.
    artikelB11.SaveChanges (
        ((ArtiklList)artikelBindingSource.DataSource).GetChanges());
    ...
}
```

```
// ostale metode
```

Poslovni sloj

Poslovni sloj

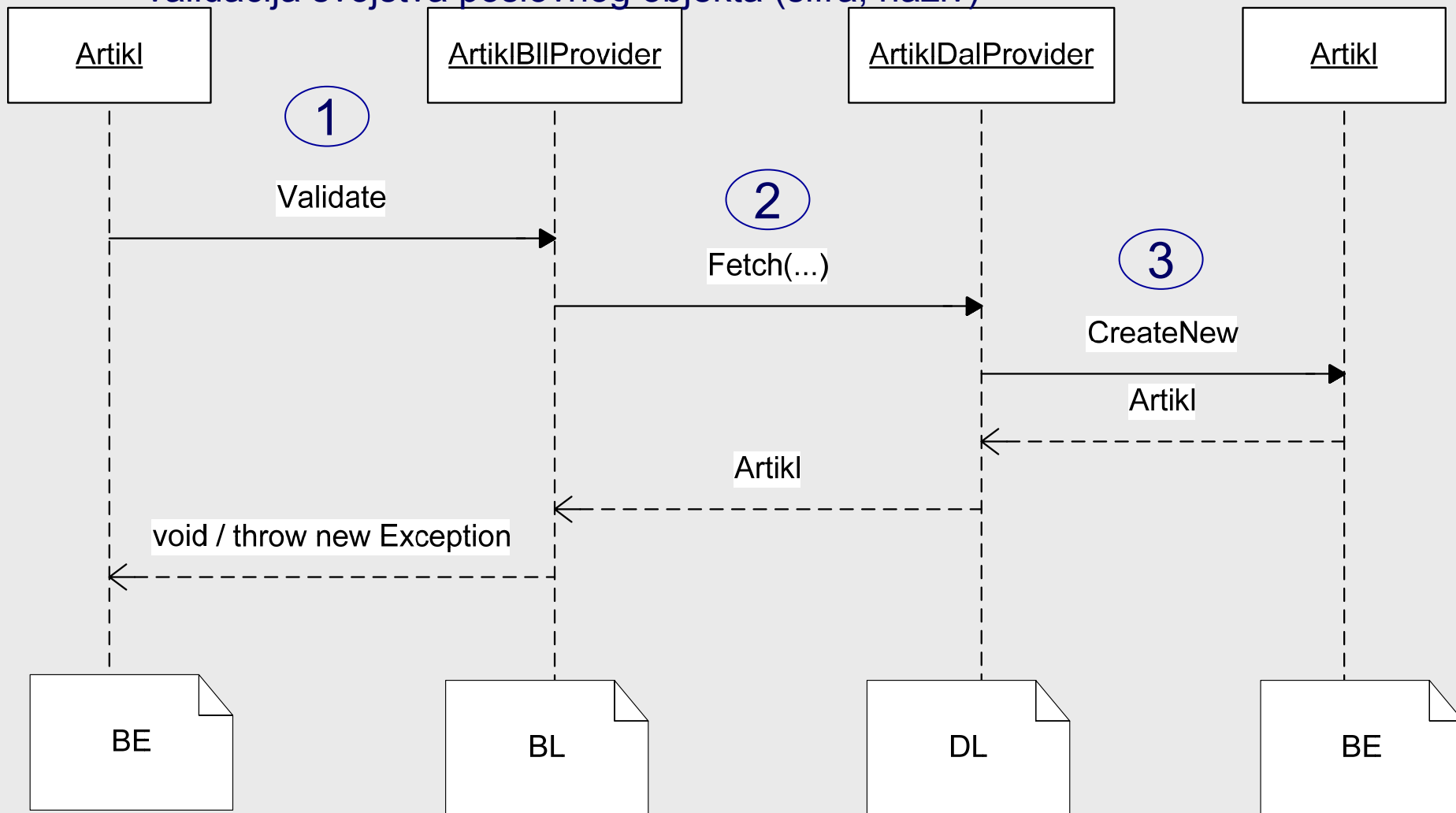
- Prosljeđuje objekte (poslovne entitete) između PL i DL
- Implementira provjeru (validaciju) poslovnih pravila (business rules)



Provjera ispravnosti

❑ Primjer: FirmaWin – Artikl (edit : *Artikl.property.set*)

- validacija svojstva poslovnog objekta (šifra, naziv)



1: Pokretanje validacije

❑ Artikel objekt - promjenom svojstva putem forme (ili programski)

```
public int? SifArtikla
{
    get { return sifArtikla; }
    set
    {
        if (InEditMode)
        {
            sifArtikla = value;
            PropertyChanged(
                "SifArtikla");
        }
    }
    ...
}
```

```
protected void PropertyChanged(
    string propertyName)
{
    isDirty = true;
    // Osvježavanje data-binding-om
    OnPropertyChanged(propertyName);

    // Validacija
    DoValidation(propertyName);
}
```

❑ BusinessBase

```
protected void DoValidation(string propertyName)
{
    // BLL sloj
    IBllObject bll = OnNeedBllObject();
    ...
    bll.Validate(this, propertyName);
}
```


2: Validacija pojedinog svojstva objekta

❏ Primjer: FirmaWin – ArtikelBIIProvider

```
public void Validate(object businessObject, string propertyName)
{
    Artikel target = (Artikel)businessObject;
    switch (propertyName)
    {
        case "SifArtikla":
            {
                if (!target.SifArtikla.HasValue)
                    throw new Exception("Šifra artikla je obavezno polje!");

                if (target.State == BusinessObjectState.New)
                {
                    Artikel a = Fetch(target.SifArtikla.Value);
                    if (a != null)
                        throw new Exception(string.Format(
                            "Artikel {0} već postoji.", a.SifArtikla.Value));
                }
                break;
            }
    }
}
```

3: Dohvat podatka za provjeru

❑ Primjer: FirmaWin – ArtiklBllProvider

```
public Artikl Fetch(int sifArtikla)
{
    return dal.Fetch(sifArtikla);
}
```

❑ Primjer: FirmaWin – ArtiklDalProvider ! **Fetch vraća objekt Artikl**

```
public Artikl Fetch(int sifArtikla)
{
    return Fetch(sifArtikla, null);
}
```

```
private Artikl Fetch(object sifArtikla, object nazArtikla)
...
    cmd.CommandText = "[dbo].[ap_Artikl_R]";
...
    using (SqlDataReader dr = cmd.ExecuteReader())
...
        return Artikl.CreateNew(dr);
```

Stanja objekata

```
public enum BusinessObjectState
{
    Unmodified,
    New,
    Modified,
    Deleted
}
```

```
private BusinessObjectState state = BusinessObjectState.Unmodified;

[Browsable(false)]
public BusinessObjectState State
{
    get { return state; }
    protected set
    {
        state = value;
        AfterStateChanged();
        OnNeedToolBarRefresh();
        OnPropertyChanged("InEditMode");
        OnPropertyChanged("State");
    }
}
```

Poništavanje promjena

❏ Primjer: FirmaWin – BusinessBase

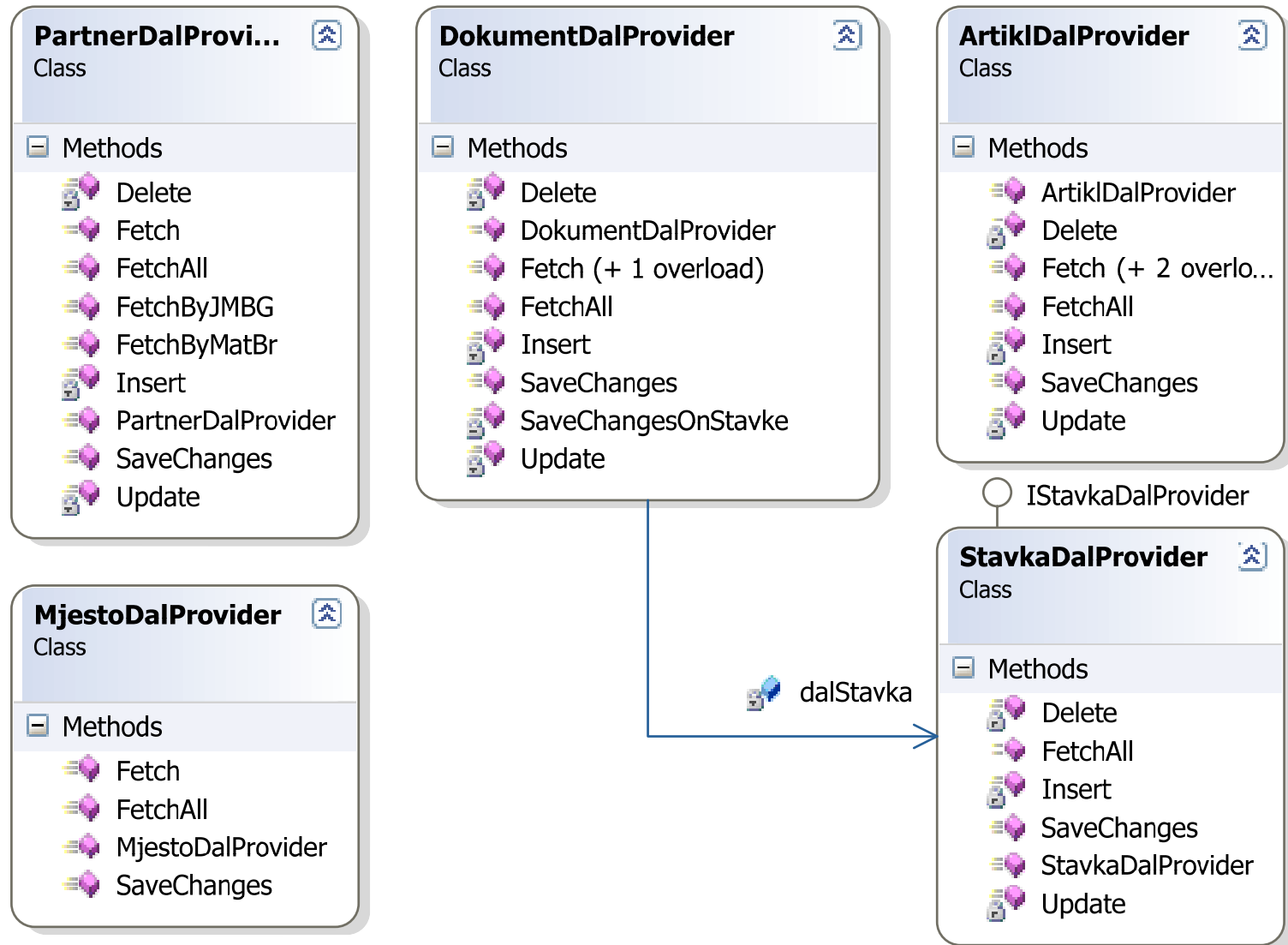
```
public void CancelChanges() {  
    (this as IEditableObject).CancelEdit();  
    if (State != BusinessObjectState.New) {  
        SetState(BusinessObjectState.Unmodified);  
    } else {  
        if (parent != null) {  
            parent.CancelNew(parent.IndexOf(this));  
            parent = null;  
        }  
    }  
    AfterCancelChanges();  
}
```

```
void IEditableObject.CancelEdit() {  
    if (!InEditMode)  
        return;  
  
    if (inEdit) {  
        Restore();  
        inEdit = false;  
    }  
}
```

Podatkovni sloj

Podatkovni sloj

❑ Osnovne funkcije za rukovanje podacima (CRUD)



Dohvat podataka iz baze podataka (još jednom)

❑ Primjer: FirmaWin – ArtiklDalProvider

- različite pohranjene procedure za traženje, odnosno dohvat pri validaciji

```
public ArtiklList FetchAll()  
    cmd.CommandText = "[dbo].[ap_ArtiklList_R]";  
    using (SqlDataReader dr = cmd.ExecuteReader())  
    ...  
    return ArtiklList.CreateNew(dr); // zove Artikl.CreateNew
```

```
private Artikl Fetch(object sifArtikla, object nazArtikla)  
    cmd.CommandText = "[dbo].[ap_Artikl_R]";  
    cmd.Parameters.Add(  
        new SqlParameter("@SifArtikla",  
            sifArtikla != null ? sifArtikla : DBNull.Value));  
    cmd.Parameters.Add(...  
    using (SqlDataReader dr = cmd.ExecuteReader())  
        if (dr != null && dr.Read())  
            return Artikl.CreateNew(dr);
```

Dohvat podataka – kreiranje objekta (već viđeno)

❏ Primjer: FirmaWin – Artikl

```
// Stvara novi objekt čitanjem iz podataka baze
public static Artikl CreateNew(IDataReader dr)
{
    Artikl rez = new Artikl();
    rez.Load(dr);
    return rez;
}
```

```
// Punjenje objekta iz čitača
protected override void DoLoad(IDataReader dr)
{
    this.sifArtikla = dr["SifArtikla"] == DBNull.Value
        ? (int?)null : (int?)dr["SifArtikla"];
    this.nazArtikla = dr["NazArtikla"] == DBNull.Value
        ? string.Empty : (string)dr["NazArtikla"];
    ...
}
```


Primjer korištene pohranjene procedure

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. On the left, the Object Explorer shows a tree view of a database named 'Firma'. The 'Stored Procedures' folder is expanded, and 'dbo.ap_Artikl_R' is selected. The right pane shows the SQL script for this procedure. The script sets ANSI_NULLS and QUOTED_IDENTIFIER to ON, then alters the procedure 'dbo.ap_Artikl_R' with parameters '@SifArtikla int = NULL' and '@NazArtikla nvarchar(255) = NULL'. The procedure body includes comments about NOCOUNT, a SET NOCOUNT ON statement, and logic to select articles based on the provided SifArtikla or NazArtikla parameters. If neither is provided, it raises an error. The status bar at the bottom indicates the server is 'Connected' and shows the instance name 'KLODOVIK\SQL05 (9.0 SP1)'.

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
GO

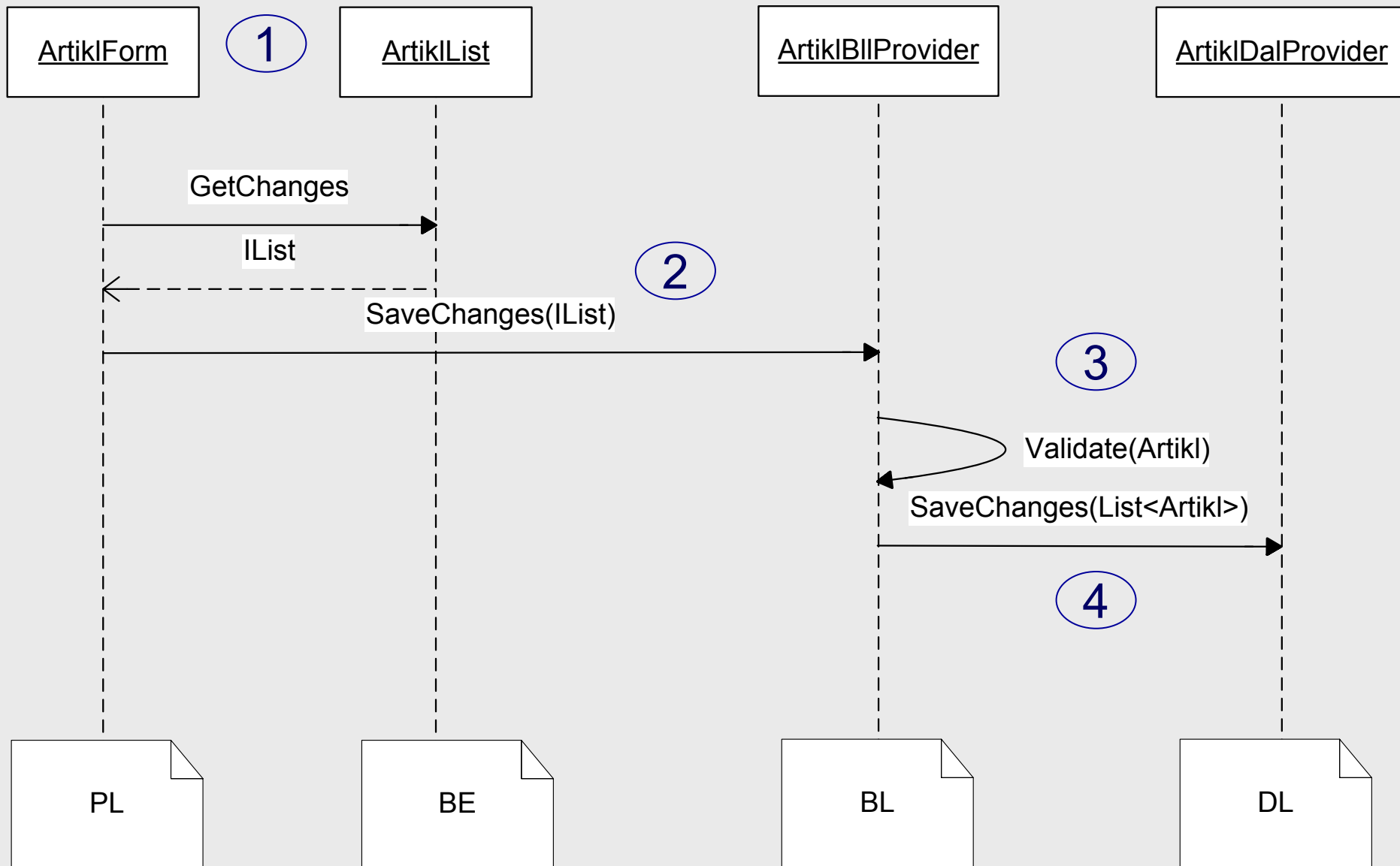
ALTER PROCEDURE [dbo].[ap_Artikl_R]
    @SifArtikla int = NULL
    ,@NazArtikla nvarchar(255) = NULL
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    IF @SifArtikla IS NOT NULL BEGIN
        SELECT * FROM Artikl WHERE SifArtikla = @SifArtikla;
    END ELSE IF @NazArtikla IS NOT NULL BEGIN
        SELECT * FROM Artikl WHERE NazArtikla = @NazArtikla;
    END ELSE BEGIN
        RAISERROR('Neispravni parametri poziva pohranjene proced...', 16, 1);
    END;

END
```

Ready

Spremanje promjena u bazu podataka



Nalog poslovnom sloju za spremanje i brisanje

❑ Primjer:  FirmaWin – ArtikelForm # pokazano uz prezentacijski sloj

```
// Spremanje svih izmjena
protected override void DoSaveChanges()
{
    artikelB11.SaveChanges (
        ((ArtiklList)artikelBindingSource.DataSource).GetChanges();
};
```

```
protected override void DoDelete()
{
    // Uklanjanje poslovnog objekta iz liste dohvaćenih objekata
    artikelBindingSource.RemoveCurrent(); // označavanje objekta
    {
        // Sprema izmjene u bazu.
        artikelB11.SaveChanges (
            ((ArtiklList)artikelBindingSource.DataSource).GetChanges());
    }
    ...
}
```

1: Stvaranje liste izmijenjenih objekata

❏ Primjer: FirmaWin – BusinessBaseList

```
public List<T> GetChanges()  
{  
    List<T> changes = new List<T>();  
    foreach (T item in deletedItems)  
    {  
        changes.Add(item);  
    }  
    foreach (T item in this)  
    {  
        if (item.IsDirty || item.State == BusinessObjectState.New)  
            changes.Add(item);  
    }  
  
    return changes;  
}
```

2: Spremanje izmjena

❏ Primjer: FirmaWin – ArtiklBIIProvider

```
public void SaveChanges(IList changedItems)
{
    foreach (Artikl item in changedItems)
    {
        // Ako je poslovni objekt izmijenjen validiraj ispravnost
        // Ako objekt nije ispravan bacit će se exception.
        if (item.IsDirty && (item.State == BusinessObjectState.New
            || item.State == BusinessObjectState.Modified))
            Validate(item); // item.Validate();
    }

    // Proslijedi DAL sloju na spremanje u bazu
    dal.SaveChanges( (List<Artikl>) changedItems );
}
```

3: Validacija poslovnog objekta

❑ Validacija svih svojstava poslovnog objekta

```
public void Validate(Artikl target)
{
    Validate(target, "SifArtikla");
    Validate(target, "NazArtikla");
    Validate(target, "JedMjere");
    Validate(target, "CijArtikla");
    Validate(target, "ZastUsluga");
    Validate(target, "SlikaArtikla");
}
```

❑ Validacija pojedinog svojstva (objašnjeno u opisu poslovnog sloja)

```
public void Validate(object businessObject, string propertyName)
{
    Artikl target = (Artikl)businessObject;
    switch (propertyName)
    {
        case "SifArtikla":
```

4: Pohrana u bazu podataka

❏ Primjer: FirmaWin – ArtiklDalProvider

```
public void SaveChanges(List<Artikl> changedItems)
{
    ...
    foreach (Artikl item in changedItems)
    {
        // Ako poslovni objekt nije mijenjan nemoj spremati
        if (!item.IsDirty) continue;

        using (SqlCommand cmd = db.CreateCommand())
        {
            cmd.Transaction = trans;
            switch (item.State)
            {
                case BusinessObjectState.New: Insert(item, cmd); break;
                case BusinessObjectState.Modified: Update(item, cmd); break;
                case BusinessObjectState.Deleted: Delete(item, cmd); break;
                default: break;
            }
        }
    }
    ...
}
```

Umetanje zapisa

```
private void Insert(Artikl item, SqlCommand cmd)
{
    cmd.CommandText = "[dbo].[ap_Artikl_C]";
    cmd.CommandType = CommandType.StoredProcedure;

    cmd.Parameters.Add(new SqlParameter(
        "@SifArtikla", SqlDbType.Int)).Value = item.SifArtikla.HasValue
        ? (object)item.SifArtikla.Value : DBNull.Value;
    ...
    cmd.ExecuteNonQuery();
}
```

```
ALTER PROCEDURE [dbo].[ap_Artikl_C]
    @SifArtikla int
    ,@NazArtikla nvarchar(255)
    ,@JedMjere nvarchar(5)
    ,@CijArtikla money
    ,@ZastUsluga bit
    ,@SlikaArtikla image
AS
BEGIN
    INSERT INTO Artikl (SifArtikla, ...)
    VALUES (@SifArtikla, ...)
```

END

Izmjena zapisa

```
private void Update(Artikl item, SqlCommand cmd)
{
    cmd.CommandText = "[dbo].[ap_Artikl_U]";
    cmd.CommandType = CommandType.StoredProcedure;

    cmd.Parameters.Add(new SqlParameter(
        "@SifArtikla", SqlDbType.Int)).Value = item.SifArtikla.HasValue
        ? (object)item.SifArtikla.Value : DBNull.Value;
    ...
    cmd.ExecuteNonQuery();
}
```

```
ALTER PROCEDURE [dbo].[ap_Artikl_U]
    @SifArtikla int
    ,@NazArtikla nvarchar(255)
    ,@JedMjere nvarchar(5)
    ,@CijArtikla money
    ,@ZastUsluga bit
    ,@SlikaArtikla image
AS
BEGIN
    UPDATE ARTIKL SET NazArtikla = @NazArtikla, ...
    WHERE SifArtikla = @SifArtikla;
```

END

Brisanje zapisa

```
private void Delete(Artikl item, SqlCommand cmd)
{
    cmd.CommandText = "[dbo].[ap_Artikl_D]";
    cmd.CommandType = CommandType.StoredProcedure;

    cmd.Parameters.Add(new SqlParameter(
        "@SifArtikla", SqlDbType.Int)).Value = item.SifArtikla.HasValue
        ? (object)item.SifArtikla.Value : DBNull.Value;

    cmd.ExecuteNonQuery();
}
```

```
ALTER PROCEDURE [dbo].[ap_Artikl_D]
    @SifArtikla int
AS
BEGIN
    DELETE FROM Artikl WHERE SifArtikla = @SifArtikla;
END
```

Složeniji primjeri

Konstruktor forme dokumenta

❏ Primjer: FirmaWin – DokumentForm

- inicijalizacija poslovnog sloja za referencirane podatke

```
public DokumentForm()  
{  
    InitializeComponent();  
  
    // Dohvat svih partnera, artikala i dokumenata za lookup  
    // Dohvat svih dokumenata  
    partnerInfoBindingSource.DataSource = partnerBll.FetchAll();  
    artiklBindingSource.DataSource = artiklBll.FetchAll();  
    dokumentBindingSource.DataSource = dokumentBll.FetchAll();  
  
    prethDokumentBindingSource.DataSource =  
        dokumentBindingSource.DataSource;
```

Poslovni objekt sa stavkama

❏ Primjer: FirmaWin – Dokument

```
public class Dokument : BusinessBase {
    public Dokument() {
        // Inicijalizacija praznih stavaka
        stavke = new StavkaList();
        stavke.ListChanged +=
            new ListChangedEventHandler(Stavke_ListChanged);
        datDokumenta = DateTime.Now;
    }

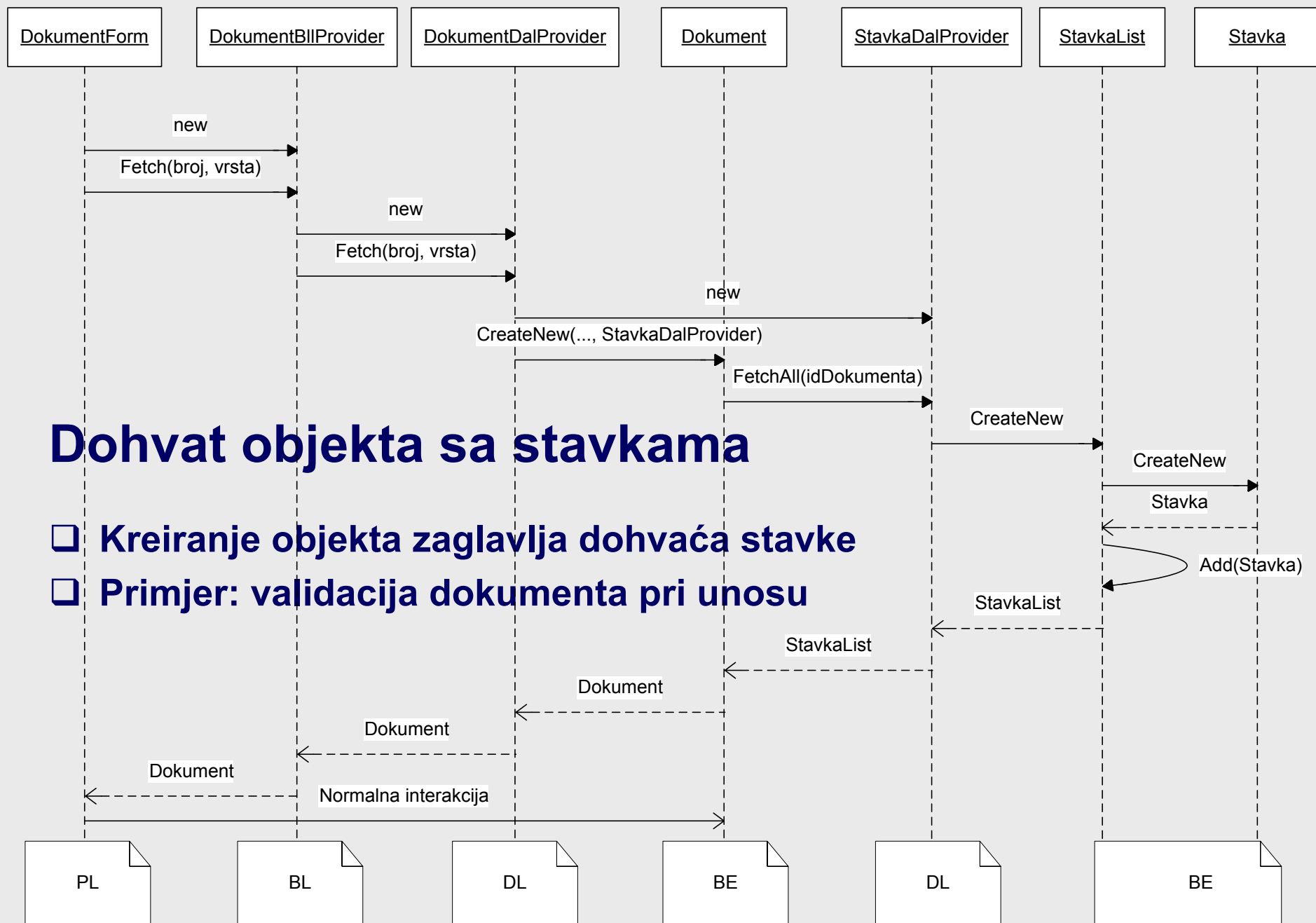
    private StavkaList stavke;
    public StavkaList Stavke
    {
        get { return stavke; }
    }
    private void Stavke_ListChanged(...) {
        // Kad se promijeni stavka osvježiti iznos dokumenta...
        OnPropertyChanged("IznosDokumenta");
    }
}
```

Povezivanje stavaka

❏ Primjer: FirmaWin – DokumentForm

- dinamičko povezivanje pri promjeni aktualnog zapisa

```
private void dokumentBindingSource_CurrentChanged(  
    object sender, EventArgs e)  
{  
    Dokument d = dokumentBindingSource.Current as Dokument;  
    if (d != null)  
    {  
        if (!d.HasBllObject)  
            d.NeedBllObject += new  
                NeedBllObjectEventHandler(Dokument_NeedBllObject);  
  
        stavkaBindingSource.DataSource = d.Stavke;  
    }  
}
```



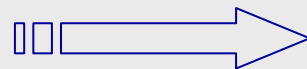
Padajuće liste referentnih podataka

❑ Partner

- `partnerInfoBindingSource.DataSource = Firma.Partner`
- `idPartneraComboBox`
 - `DataSource = partnerInfoBindingSource`
 - `DisplayMember = Naziv`
 - `ValueMember = IdPartnera`
 - `SelectedValue = dokumentBindingSource - IdPartnera`

❑ Prethodni dokument

- `prethDokumentBindingSource.DataSource = Firma.Dokument`
- `idPrethDokumentaComboBox`
 - `DataSource = prethDokumentBindingSource`
 - `DisplayMember = LookupText`
 - `ValueMember = IdDokumenta`
 - `SelectedValue = dokumentBindingSource - IdPrethDokumenta`



Izračunata vrijednost za odabir iz padajuće liste

❏ Primjer: FirmaWin – Dokument

```
public string LookupText
{
    get { return this.ToString(); }
}
public override string ToString()
{
    if (brDokumenta.HasValue
        && !string.IsNullOrEmpty(vrDokumenta))
    {
        return vrDokumenta
            + " " + brDokumenta.Value.ToString()
            + " - " + datDokumenta;
    }
    else
    {
        return "Dokument";
    }
}
```

Izračunata vrijednost iznosa dokumenta

❑ Primjer: FirmaWin – DokumentForm

```
iznosDokumentaTextBox.DataBindings.Add(new Binding("Text",  
    dokumentBindingSource, "IznosDokumenta", true,  
    DataSourceUpdateMode.OnPropertyChanged, string.Empty, "C2"));
```

❑ Primjer: FirmaWin – Dokument

```
public decimal? IznosDokumenta {  
    get {  
        decimal rez = 0;  
        foreach (Stavka s in this.stavke) {  
            if (s.Iznos.HasValue)  
                rez += s.Iznos.Value;  
        }  
  
        if (postoPorez.HasValue) {  
            rez *= (1m + postoPorez.Value);  
        }  
        return rez;  
    }  
}
```

Ulančavanje referentnog podatka sa zaglavlja

❑ Primjer: FirmaWin – DokumentForm : PartnerForm

```
protected override void Zoom()
{
    if (InEditMode)
    {
        if (idPartneraComboBox.Focused)
        {
            PartnerForm f = new PartnerForm();
            f.StartPosition = FormStartPosition.CenterScreen;
            if (f.ShowDialog() == DialogResult.OK)
            {
                // Potrebno zbog bindinga na lookup novounesenih
                partnerInfoBindingSource.DataSource =
                    partnerBll.FetchAll();

                Partner p = (Partner)f.Selected;
                (dokumentBindingSource.Current
                    as Dokument).IdPartnera = p.IdPartnera;
            }
        }
    }
}
```

Ulančavanje referentnog podatka sa stavke

❏ Primjer: FirmaWin – DokumentForm : ArtiklForm

```
if (InEditMode &&
    stavkaDataGridView.SelectedCells[0].ColumnIndex == 0 && ...) {
ArtiklForm f = new ArtiklForm();
if (f.ShowDialog() == DialogResult.OK) {
    artiklBindingSource.DataSource = artiklBll.FetchAll();
    ...
if (stavkaDataGridView.Rows...IsNewRow) { // SKRAĆENO
    stavkaBindingSource.CancelEdit();
    Dokument dok = (Dokument)dokumentBindingSource.Current;
    s = new Stavka();
    dok.Stavke.Add(s);
} else
    s = (Stavka)stavkaBindingSource.Current;

s.SifArtikla = a.SifArtikla;
s.NazArtikla = a.NazArtikla;
...
}
```

Dohvat složenog objekta iz baze podataka

❑ Primjer: FirmaWin – DokumentDalProvider

- kreatoru liste i kreatoru objekta predaje se i DAL objekt stavaka

```
public DokumentList FetchAll() {  
    ...  
    cmd.CommandText = "[dbo].[ap_DokumentList_R]";  
    using (SqlDataReader dr = cmd.ExecuteReader()) {  
        return DokumentList.CreateNew(dr, dalStavka);  
    }  
}
```

```
public Dokument Fetch(int brDokumenta, string vrDokumenta)  
...  
    cmd.CommandText = "[dbo].[ap_Dokument_R]";  
    cmd.Parameters.Add(  
        new SqlParameter("@BrDokumenta", brDokumenta));  
    ...  
    using (SqlDataReader dr = cmd.ExecuteReader())  
    ...  
        return Dokument.CreateNew(dr, dalStavka);  
}
```

Dohvat stavki dohvaćenog zaglavlja

❑ Primjer: FirmaWin – Dokument (također iz *DokumentList*)

```
public static Dokument CreateNew(...) {  
    Dokument rez = new Dokument();  
    rez.Load(dr, stavkaDalProvider);  
    return rez;  
}
```

❑ Primjer: FirmaWin – Dokument

```
public void Load(  
    IDataReader dr, IStavkaDalProvider stavkaDalProvider)  
{  
    Load(dr);  
    if (idDokumenta.HasValue)  
    {  
        stavke = stavkaDalProvider.FetchAll(idDokumenta.Value);  
    }  
    else  
    {  
        stavke = new StavkaList();  
    }  
}
```

Ostali postupci podatkovnog sloja zaglavlja

❑ Primjer: FirmaWin – DokumentDalProvider

❑ Dodavanje

- `cmd.CommandText = "[dbo].[ap_Dokument_C]";`
 - `ap_Dokument_C : SET @IdDokumenta = SCOPE_IDENTITY();`
- dohvat OUTPUT vrijednosti serijskog primarnog ključa
 - `item.IdDokumenta =`
`(int?)cmd.Parameters["@IdDokumenta"].Value;`
- `SaveChangesOnStavke(item, ...);`
 - `dalStavka.SaveChanges(target.Stavke.GetChanges(), db,`
`trans);`

❑ Izmjena

- `cmd.CommandText = "[dbo].[ap_Dokument_U]";`
- `SaveChangesOnStavke(item, ...);`

❑ Brisanje

- `cmd.CommandText = "[dbo].[ap_Dokument_D]";`

Povezivanje na tip specijalizacije

❑ Primjer: FirmaWin – PartnerForm

- automatski prikaz panela prikladnog tipu partnera
 - povezivanjem na kontrole osobaCheck, odnosno tvrtkaCheck
- kontrole mijenjaju vrijednost
 - navigacijom na zapis (partnerBindingSource_CurrentChanged)
 - interaktivno, tijekom uređivanja

```
osobaPanel.DataBindings.Add(  
    new Binding("Visible", osobaCheck, "Checked"));  
  
tvrtkaPanel.DataBindings.Add(  
    new Binding("Visible", tvrtkaCheck, "Checked"));
```

❑ Primjer: FirmaWin – TipPartnera

- enum TipPartnera
- class TipPartneraConverter

```
public enum TipPartnera  
{  
    Osoba,  
    Tvrtka,  
    Nedefinirano  
}
```


Prikaz tipa specijalizacije pri navigaciji

❏ Primjer: FirmaWin – PartnerForm

```
private void partnerBindingSource_CurrentChanged(...)
{
    Partner p = partnerBindingSource.Current as Partner;

    if (p == null) // TipPartnera.Nedefinirano
    {
        osobaCheck.Checked = false;
        tvrtkaCheck.Checked = false;
    }
    else if (p.TipPartnera == TipPartnera.Tvrtka)
    {
        osobaCheck.Checked = false;
        tvrtkaCheck.Checked = true;
    }
    else // TipPartnera.Osoba
    {
        osobaCheck.Checked = true;
        tvrtkaCheck.Checked = false;
    }
}
```

Promjena tipa specijalizacije pri uređivanju

❑ Primjer: FirmaWin – PartnerForm

```
private void tvrtkaCheck_CheckedChanged(...)
{
    Partner p = partnerBindingSource.Current as Partner;
    if (p != null)
    {
        if (tvrtkaCheck.Checked)
        {
            p.TipPartnera = TipPartnera.Tvrtka;
        }
        else
        {
            p.TipPartnera = TipPartnera.Osoba;
        }
    }
}
```

Pristup podacima koji sudjeluju u "jest" vezi

❑ Primjer: FirmaWin – PartnerDalProvider

- pozivaju se modificirane pohranjene procedure i pogledi

❑ Dohvat (ap_PartnerList_R)

- `SELECT ... FROM Osoba UNION SELECT ... FROM Tvrtka`

❑ Dodavanje (ap_Partner_C)

- `INSERT INTO Partner`
- uvjetno `INSERT INTO Osoba \ INSERT INTO Tvrtka`
- **// Dohvat OUTPUT vrijednosti**
 - `SET @IdPartnera = SCOPE_IDENTITY();`

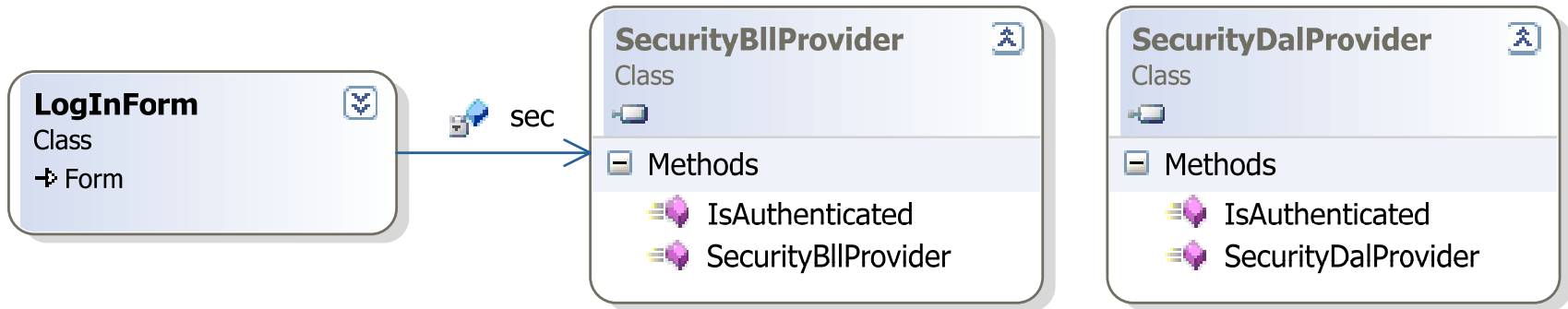
❑ Izmjena (ap_Partner_U)

- `UPDATE Partner SET ...`
- uvjetno `UPDATE Osoba \ UPDATE Tvrtka`

❑ Brisanje (ap_Partner_D)

- `DELETE FROM Partner WHERE IdPartnera = @IdPartnera;`
- alternativa je obrisati specijalizaciju a onda Partner

Provjera korisnika



```
public class SecurityDalProvider
{
    public bool IsAuthenticated(string username, string password)

    using (SqlCommand cmd = db.CreateCommand())
    {
        cmd.CommandText = "[dbo].[ap_Login]";
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.Add(new SqlParameter("@Username", username));
        cmd.Parameters.Add(new SqlParameter("@Password", password));

    try
    {
        return Convert.ToInt32(cmd.ExecuteScalar()) == 1;
    }
}
```


Pohranjena procedura za provjeru korisnika

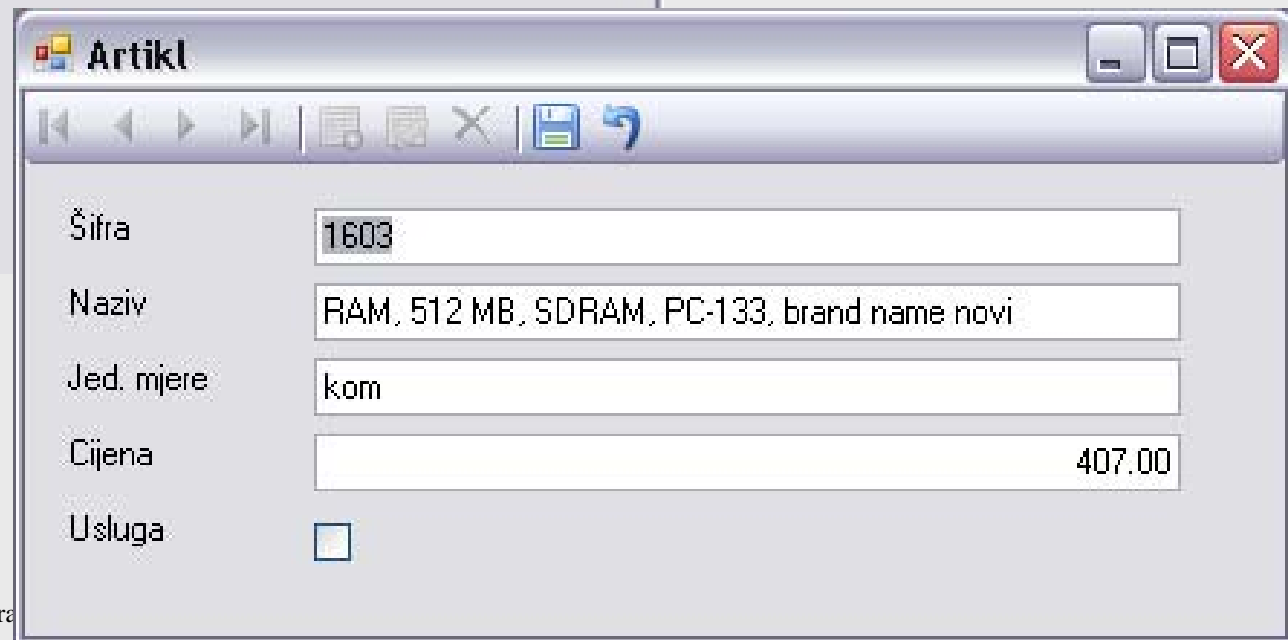
- ❑ U stvarnoj aplikaciji branimo se šifriranjem zaporke, kriptiranjem mrežnog prometa, ...

```
ALTER PROCEDURE [dbo].[ap_Login]
    @Username nvarchar(15)
    ,@Password nvarchar(15)
AS
BEGIN
    SELECT COUNT(*) FROM Korisnik
        WHERE Username = @Username
        AND Password = @Password;
END
```

Univerzalni i samoprilagodljivi programski moduli

Univerzalna forma

- ❑ Preddefinirana forma koja se pri pokretanju prilagodi podacima koje treba obraditi.
- ❑ Primjer:  FirmaWin \ Core – GenericForm.cs

A screenshot of a software window titled "Artiki". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar is a toolbar with several icons: a double left arrow, a single left arrow, a single right arrow, a double right arrow, a document icon, a document with a magnifying glass icon, a close icon (X), a save icon (floppy disk), and a refresh icon (circular arrow). The main area of the window contains a form with the following fields:

| | |
|------------|---|
| Šifra | 1603 |
| Naziv | RAM, 512 MB, SDRAM, PC-133, brand name novi |
| Jed. mjere | kom |
| Cijena | 407.00 |
| Usluga | <input type="checkbox"/> |

Pokretanje univerzalne forme

❑ Umjesto instanciranja namjenski oblikovanje forme

- npr. `ArtiklForm f = new ArtiklForm();`
- koja pri pokretanju prikupi i poveže potrebne podatke

❑ Kreira se općenita forma

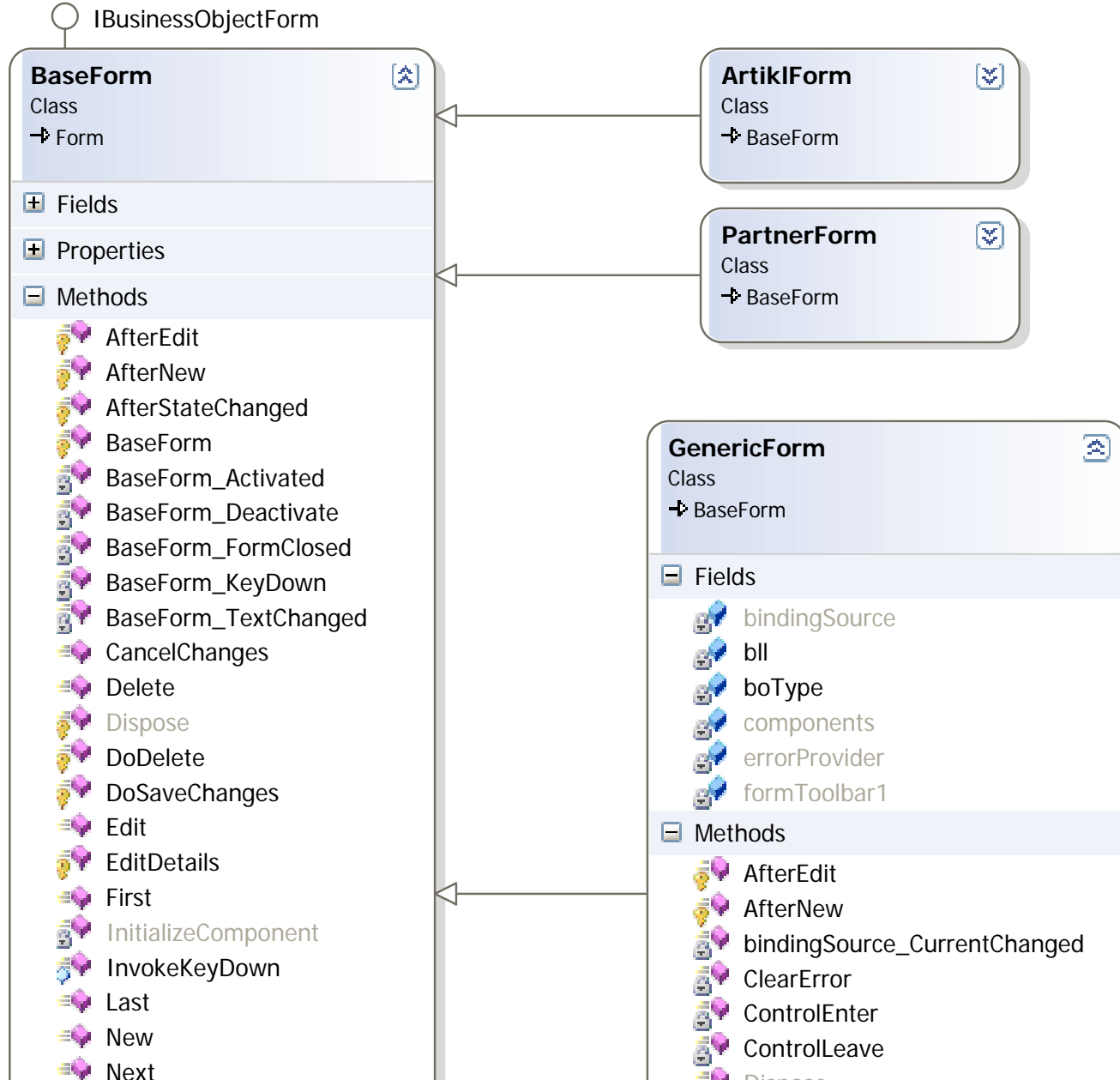
- koja se prilagodi predanom skupu podataka i odgovarajućem tipu

❑ Primjer: FirmaWin – MainForm - Šifrnici


```
private void artiklToolStripTablice_Click(...)
{
    using (new StatusBusy())
    {
        ArtiklBllProvider bll = new ArtiklBllProvider();
        GenericForm f = new GenericForm("Artikl",
            bll.FetchAll(), bll, typeof(Artikl));
        f.MdiParent = this;
        f.Show();
    }
}
```


Dizajn univerzalne forme

- ❑ Funkcionalnost izvedena iz osnovne forme, koju nasljeđuju i druge forme.

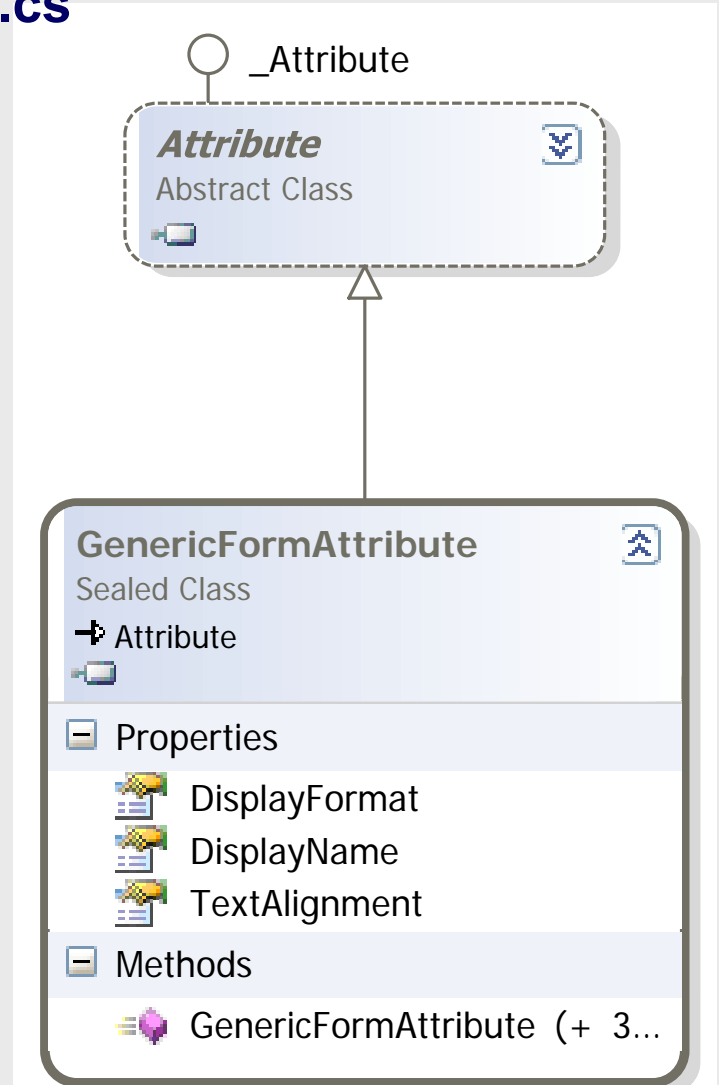


Prilagodba ponašanja

- ❑ Prilagodba korisnički definiranim atributima poslovnog entiteta
- ❑ Primjer:  FirmaWin \ Shared – Artikl.cs

```
[GenericForm("Šifra")]  
public int? SifArtikla {  
    get { return sifArtikla; }  
    set { ... }  
}  
...  
GenericForm("Cijena", "N2",  
    HorizontalAlignment.Right)]  
public decimal? CijArtikla {  
    get { return cijArtikla; }  
    set { ... }  
}
```

? iza naziva tipa označava tzv. nulabilni tip podataka



Atributi

❑ Atributi – deklaracija opisa nalik na onu ključnim riječima

- Standardni način preciziranja je onaj ključnim riječima (*public*, *private*, ...)
- Atributi su nadodane oznake tipovima, poljima, postupcima i svojstvima.
- Definirani uglatim zagradama [] prije deklaracije entiteta koji opisuju
- Dostupni programski tijekom izvođenja programa

❑ Primjeri preddefiniranih atributa

- `Browsable` – vidljivost svojstva ili događaja u *Properties* prozoru
- `Category` – kategorija u kojoj se svojstvo ili događaj prikazuje u *Properties*
 - Neke preddefinirane kategorije: *Data*, *Behavior*, *Design*, *Action*, *Misc*
 - Moguće je definirati i vlastite kategorije
- `Description` - opis svojstva ili događaja

❑ Primjer: 📁 **FirmaWin – BaseForm.cs / ArtiklForm (dizajn)**

```
[Browsable(true), Category("Data")]  
public BindingSource MainBindingSource {  
    get { return mainBindingSource; }  
    set { mainBindingSource = value; }  
}
```

System.Attribute

❑ **Razred *System.Attribute* – osnovni razred za atribut**

- `GetCustomAttribute` – vraća atribut danog tipa *Type* primijenjenog na asemblij, član razreda, ...
 - npr. `GetCustomAttribute(MemberInfo, Type)`
- `GetCustomAttributes` – vraća polje atributa
- `IsDefined` – određuje da li je ijedan atribut zadanog tipa *Type* definiran
 - npr. `IsDefined(MemberInfo, Type)`

❑ **Vlastiti atributi nasljeđuju *System.Attribute***

Definiranje vlastitih atributa

❑ Primjer: FirmaWin \ Firma.Framework – GenericFormAttribute

- atribut za označavanje svojstava koja će se pokazati na univerzalnoj formi

```
public sealed class GenericFormAttribute : Attribute {  
    private string displayName = string.Empty;  
    private string displayFormat = string.Empty;  
    private HorizontalAlignment ha = HorizontalAlignment.Left;  
    ...  
    public GenericFormAttribute(string displayName) {
```

❑ Primjer: FirmaWin \ Shared – Artikl.cs

```
[GenericForm("Cijena", "N2", HorizontalAlignment.Right)]  
public decimal? CijArtikla {  
    get { return cijArtikla; }  
    set { ... }  
}
```

Refleksija

- ❑ **Proces kojim program može pregledavati i modificirati vlastitu strukturu tijekom izvođenja**
- ❑ **Refleksija se upotrebljava za:**
 - Dohvat metapodataka (sadržanim u atributima)
 - Otkrivanje tipa podatka
 - Pristup informacijama o učitanim asemblijima i tipovima definiranim unutar njih (pregled interakcija i instanciranje tipova)
 - Dinamičko povezivanje na svojstva i postupke
 - Pozivanje svojstava ili postupaka dinamički instanciranog objekta na temelju otkrivenog tipa (*dynamic invocation*), npr. bind fonta ili boje
 - Stvaranje i korištenje novih tipova za vrijeme izvršavanja programa
- ❑ **Primjeri upotrebe**
 - Razvoj aplikacija za reverzno inženjerstvo
 - Razvoj preglednika razreda
 - Razvoj editora svojstava razreda (kao *Properties* prozor)
- ❑ **Prostor imena *System.Reflection* i razred *System.Type***

Prostor imena *System.Reflection*

□ Prostor imena *System.Reflection*

- Sadrži razrede i sučelja za dohvat informacija o tipovima i članovima programskog koda pri izvođenju, kao i dinamičko kreiranje tipova

□ Razredi

- `Assembly` – predstavlja asemblij
 - Postupci `Load`, `LoadFile`, `GetName`, `GetModules`, ...
- `MemberInfo` (apstraktni razred)
 - pruža pristup metapodacima članovima razreda
- `ConstructorInfo`, `PropertyInfo`, `MethodInfo`
 - Pristup metapodacima konstruktora, svojstava, odnosno postupaka
 - Nasljeđuju `MemberInfo` razred
 - `ConstructorInfo` i `MethodInfo` imaju još i postupke `IsPrivate`, `IsPublic`, `IsStatic`, ...

Razred *System.Type*

□ Predstavlja deklaraciju tipova (razreda, sučelja, polja, ...)

- Osnova za refleksiju, za pristup metapodacima
- `typeof()` – vraća objekt razreda *Type* za zadani tip podataka
 - `typeof(string) -> String`
- `Object.GetType()` – vraća objekt razreda *Type* za zadani objekt
 - Za *string* `s, s.GetType() -> String`

□ Postupci

- `GetConstructor(type[])` – dohvaća konstruktor čiji tipovi parametara odgovaraju zadanom polju tipova
- `GetProperty(string)` – dohvaća svojstvo određenog naziva
- `GetProperty(string, BindingFlags)` – dohvaća svojstvo uz filter kontrole povezivanja
- `GetMethod(string), GetMethod(string, BindingFlags)`
- `GetConstructors, GetProperties, GetMethods` – dohvaća konstruktore, svojstva, odnosno postupke danog tipa podatka
 - `GetProperties(BindingFlags)`

Obrada atributa refleksijom

❑ Primjer: FirmaWin \ Core – GenericForm (SetupForm)

- Dohvat svih javnih svojstava danog poslovnog objekta
- Provjera da li je svojstvo označeno atributom za prikaz na univerzalnoj formi
- Kreiranje prikladne kontrole za svojstvo

```
private Type boType;  
PropertyInfo[] props = boType.GetProperties(  
    BindingFlags.Instance | BindingFlags.Public);  
  
foreach (PropertyInfo prop in props) {  
    if (Attribute.IsDefined(prop, typeof(GenericFormAttribute))) {  
        GenericFormAttribute atr =  
            (GenericFormAttribute)Attribute.GetCustomAttribute(prop,  
                typeof(GenericFormAttribute));  
  
        ... //kreiranje kontrole za svojstvo  
    }  
    ...  
}
```

Kreiranje kontrole prema tipu svojstva

- ❑ Za tip svojstva *bool*, *DateTime* ili *string* kreira se kontrola *CheckBox*, *DateTimePicker*, odnosno *TextBox*

```
Control c = null;
if (prop.PropertyType.Equals(typeof(bool)) ||
    prop.PropertyType.Equals(typeof(bool?))) {
    c = new CheckBox();
    c.DataBindings.Add(new Binding(
        "CheckState", bindingSource, prop.Name, true));
}
else if (prop.PropertyType.Equals(typeof(DateTime)) ||
        prop.PropertyType.Equals(typeof(DateTime?))) {
    c = new DateTimePicker();
    c.DataBindings.Add(new Binding("Value", bindingSource, prop.Name));
}
else {
    c = new TextBox();
    ...
}
```

Dinamičko kreiranje kontrola na formi

```
// Kreiranje labele za svojstvo
Label l = new Label();
l.Text = atr.DisplayName;

// Postavljanje kontrole i labele na formu
this.Controls.Add(l);
l.Location = labelPos;
labelPos.Y += 26;

c.Width = 300;
this.Controls.Add(c);
c.Location = controlPos;
controlPos.Y += 26;

l.SendToBack();
c.Enter += new EventHandler(ControlEnter);
c.Leave += new EventHandler(ControlLeave);
```

Rukovanje svojstvom temeljem atributa

❑ Artiki.cs

```
[GenericForm("Šifra")]
public int? SifArtikla
{
    get { return sifArtikla; }
    set
    {
        if (InEditMode)
        {
            sifArtikla = value;
            PropertyChanged(
                "SifArtikla");
        }
    }
}
```

❑ BusinessBase.cs

```
protected void PropertyChanged(
    string propertyName)
{
    isDirty = true;
    // Osvježavanje data-binding-om
    OnPropertyChanged(propertyName);

    // Validacija
    DoValidation(propertyName);
}
```



Još jedan primjer primjene refleksije

❑ Primjer: FirmaWin – Utils,

- Postupak `SetNull` – postavlja *null* na kontrole povezane na izvor podataka

❑ Primjer: FirmaWin – **GenericForm(ControlLeave)**

```
public static class Utils {  
    public static void SetNull(Control c,  
        string bindedProperty, object businessObject)    {  
        Binding b = c.DataBindings[bindedProperty];  
        PropertyInfo p = businessObject.GetType().GetProperty(  
            b.BindingMemberInfo.BindingField,  
            BindingFlags.Instance | BindingFlags.Public);  
  
        if (p.PropertyType.Equals(typeof(string))) {  
            p.SetValue(businessObject, string.Empty, null);  
        }  
        else {  
            p.SetValue(businessObject, null, null);  
        }  
    }  
}
```



Zadaci za vježbu

- ❑ **Ugraditi generičku višeslojnu komponentu za šifrarnik *Drzava* po uzoru na generičko rješenje za *Artikl*, koje se sastoji od**
 - *DrzavaBIIProvider*
 - *DrzavaDalProvider*
 - *Drzava*
 - *DrzavaList*
 - prateće pohranjene procedure
 - instanciranje generičke forme u glavnom izborniku

- ❑ **Ugraditi višeslojnu komponentu *Mjesto*, koja ulančava poslovni objekt i formu prethodno ugrađene komponente *Drzava***
 - *DrzavaForm*
 - *DrzavaBIIProvider*
 - *DrzavaDalProvider*
 - *Drzava*
 - *DrzavaList*
 - prateće pohranjene procedure
 - instanciranje forme u glavnom izborniku

Reference

❑ K.Hyatt: N-Tier Application Development with Microsoft.NET

- <http://www.microsoft.com/belux/msdn/nl/community/columns/hyatt/ntier1.mspx>

❑ Rockford Lhotka, CSLA.NET Framework

- <http://www.lhotka.net/cslanet/>

❑ MSDN

- Attribute Class
- Reflection (C# Programming Guide)
- Accessing Attributes With Reflection (C# Programming Guide)