Laborator 4.2

1. Mecanisme de tip “factory”

a. Design pattern-ul Factory method

Sursa: Wikipedia, http://en.wikipedia.org/wiki/Factory_method

```
public interface ImageReader {
    public DecodedImage getDecodedImage();
}

public class GifReader implements ImageReader {
    public DecodedImage getDecodedImage() {
        return decodedImage;
    }
}

public class JpegReader implements ImageReader {
    // ....
}
```

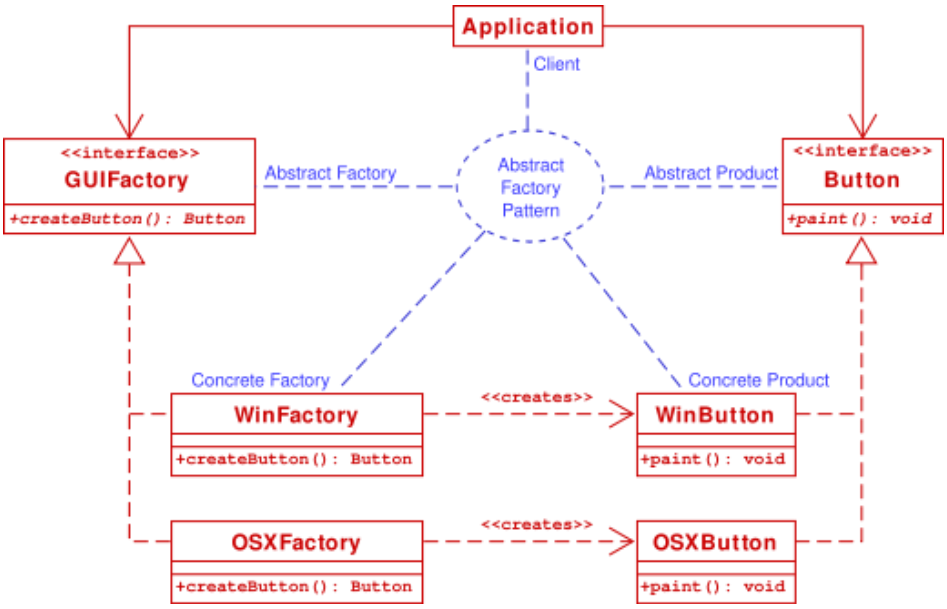
Factory method:

```
public class ImageReaderFactory {
    public static ImageReader getImageReader(InputStream is) {
        int imageType = determineImageType(is);//e.g. use file extension
        //or read file's header

        switch(imageType) {
            case ImageReaderFactory.GIF:
                return new GifReader(is);
            case ImageReaderFactory.JPEG:
                return new JpegReader(is);
            // etc.
        }
    }
}
```

b. Design pattern-ul Abstract Factory

Scop: grupare de obiecte de tip factory inrudite [Wikipedia].




```
interface Button is
  method paint()

interface GUIFactory is
  method createButton()
      output:  a button

class WinFactory implementing GUIFactory is
  method createButton() is
      output:  a Windows button
    Return a new WinButton

class OSXFactory implementing GUIFactory is
  method createButton() is
      output:  an OS X button
    Return a new OSXButton

class WinButton implementing Button is
  method paint() is
    Render a button in a Windows style

class OSXButton implementing Button is
  method paint() is
    Render a button in a Mac OS X style

class Application is
  constructor Application(factory) is
      input:  the GUIFactory factory used to create buttons
    Button button := factory.createButton()
    button.paint()

Read the configuration file
If the OS specified in the configuration file is Windows, then
  Construct a WinFactory
  Construct an Application with WinFactory
else
  Construct an OSXFactory
  Construct an Application with OSXFactory
```

2. Data access application block

Caracteristici

- Utilizat pentru accesarea datelor din baza de date
- Reduce considerabil catitatea de cod necesara
- Codul este cu grad mare de independenta fata de serverul de baze de date utilizat

Exemplu: cod classic ADO.NET:

```
SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["myCS"].ConnectionString);
SqlCommand cmd = new SqlCommand("GetCustomerList", con);
cmd.CommandType = CommandType.StoredProcedure;
try
 {
    con.Open();
    SqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
          textBox1.Text += reader["ContactName"] + Environment.NewLine;
    }
    reader.Close();
 }
 finally
 {
    con.Close();
 }
```

Comentarii:

- cod mult, dependent de serverul de baze de date;
- chiar daca ADO.NET introduce mecanismul de furnizor de date (data provider = independenta fata de furnizor, cod generic), cantitatea de cod ce trebuie scrisa nu scade; a se vedea explicarea mecanismului de ADO.NET data provider in [2], pag 170, sectiunea ADO.NET 2.0 Provider Factories.

Data access application block ofera:

- mai multa independenta fata de furnizorul de date
- cele mai des intalnite comenzi sunt factorizate astfel incat sa reduca mult codul scris
- se poate folosi parte de caching pentru parametri; parametrii se pot reutiliza in mod transparent .

Varianta cu Data Access Application Block:

```
//linia de mai jos se poate executa o singura data, de exemplu din metoda Main()
DatabaseFactory.SetDatabaseProviderFactory(new DatabaseProviderFactory());

Database database = DatabaseFactory.CreateDatabase("myConStr");
using (IDataReader reader = database.ExecuteReader(CommandType.StoredProcedure,
"[GetCustomerList]"))
{
```

```
    while (reader.Read())
    {
        txtCustomers.Text += reader["CompanyName"] + " from " + reader["City"] +
Environment.NewLine;
    }
}
```

Stringul de conexiune este scris astfel:

```
<add connectionString="server=.\sqlexpress; database=northwind; uid=sa; pwd=1q2w3e; "
name="myConStr" providerName="System.Data.SqlClient"/>
```

Fisierul de backup al bazei de date northwind este db_backup\northwind.bak.

Esential pentru utilizarea bibliotecii este atributul *providerName*, specificand care este tipul de furnizor de date ADO.NET prin care se face accesul la date.

Pentru valori ale atributului providerName suportate de .NET Framework: http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource.providername.aspx; pentru alte surse de date valoarea stringului este dependenta de implementarea pusa la dispozitie (e.g. MySQL: providerName="MySql.Data.MySqlClient"; trebuie sa fie distribuite si fisierele dll care implementeaza furnizorul de date MySQL).

Pentru utilizare, trebuie inclus spatiul de nume Microsoft.Practices.EnterpriseLibrary.Data si adaugata referinta la Microsoft.Practices.EnterpriseLibrary.Common.dll si la Microsoft.Practices.EnterpriseLibrary.Data.dll. Alternativ, se foloseste Nuget pentru aducerea de pachete cu dependinte.

NOTA: daca obtineti eroare de compilare de forma: "The type or namespace name 'Data' does not exist in the namespace 'Microsoft.Practices.EnterpriseLibrary' (are you missing an assembly reference?)" trebuie ca proiectul sa fie compilat cu target .NET Framework 4 (in loc de .NET Framework 4 Client profile). Acest lucru se seteaza astfel: click dreapta pe proiectul care foloseste referinta la Microsoft.Practices.EnterpriseLibrary.Data.dll, properties, in tab-ul Application se alege la lista "Target framework" optiunea ".Net Framework 4".

Se remarca:

- cod sensibil mai putin
- lucrul pe interfete si clase agnostice de bazele de date (obiectele db, reader)
- se folosesc concepte din ADO.NET, dar nu atat de multe ca la exemplul precedent
- obiectul DataReader inca trebuie inchis manual (are sens, deoarece numai programatorul stie cand a terminat de lucrat cu el); asta duce la inchiderea automata a conexiunii utilizate (aspect ce este prezent si in ADO.NET, dar cu cod scris explicit)

Cod demonstrativ: DemoDataAccessApplicationBlock, baza de date Northwind se poate restaura din .\db_backup\northwind.bak

Tutorial:

1. (concis, focusat pe versiunea 4.1, inca util) http://aspnet.4guysfromrolla.com/articles/030905-1.aspx
2. Developer's Guide to Microsoft Enterprise Library, cap 2.

Tema:

1. Folosind metode de tip DataSet, sa se lucreze cu un obiect de tip DataSet populat/salvat din/in baza de date; documentatie: Developer’s Guide to Microsoft Enterprise Library, cap 2

2. Functioneaza Data Access Application Block pentru accesarea datelor dintr-un fisier Excel?

3. Urmarind sectiunea “Retrieving data as objects” din Developer’s Guide to Microsoft Enterprise Library, cap 2, transformati rezultatul unei interogari intr-o colectie de obiecte. Cum se face asocierea intre coloanele aduse din baza de date si proprietatile ce trebuie populate?