Lab 1 Helper

Ioana Ciuciu,
ioana.ciuciu@ubbcluj.ro

Info about the lab ☺

- Lab requirements available here:
  - www.cs.ubbcluj.ro/~sabina
- 2 weeks delay = 1 point penalty
- Max 2 lab assignments / lab
- Final lab grade: ((GradeLab1-PenaltyLab1) + (GradeLab2-PenaltyLab2) + (GradeLab3-PenaltyLab3))/3
- No lab delivery during weeks 13, 14 and during the exams (sesiune)
- During retake session (restante): max 2 labs, with a penalty of 35%, only if the practical exam is retaken (except when the student has 10 p. for the practical exam)
- Attendance: 6 labs out of 7 (https://www.cs.ubbcluj.ro/wp-content/uploads/Hotarare-CDI-29.04.2020.pdf)
- Practical exam: weeks 13, 14 (in order to promote, a grade >= 5 is needed)

Prerequisites

- Visual Studio – installed

- For Linux users
  - Virtual machine, or
  - Mono Project (https://www.mono-project.com/)
    - Open source implementation of Microsoft's .NET Framework

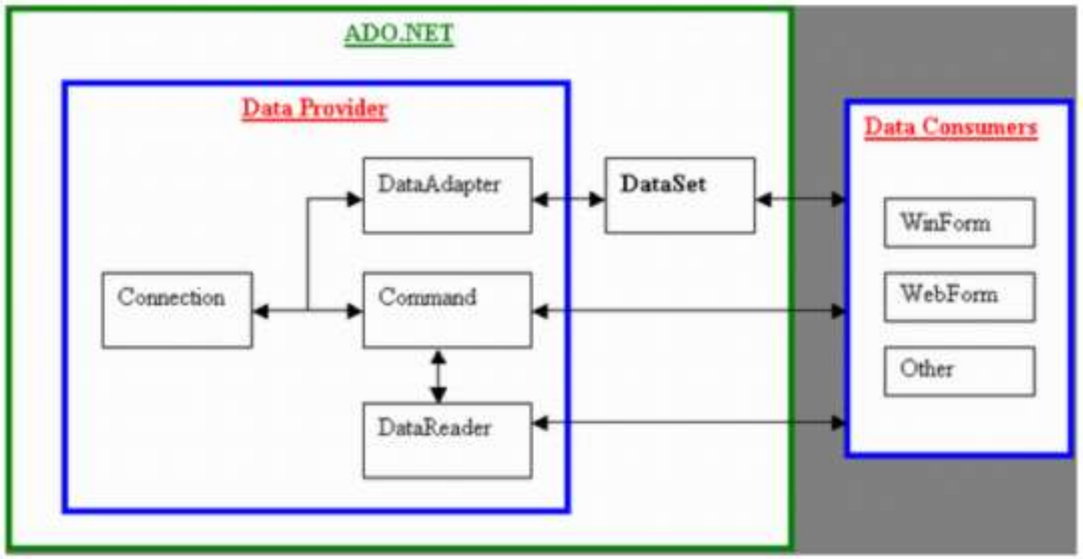
  - ! For the practical exam, an app using Windows Forms will be required!

  - Teams access code: **n9din1i**

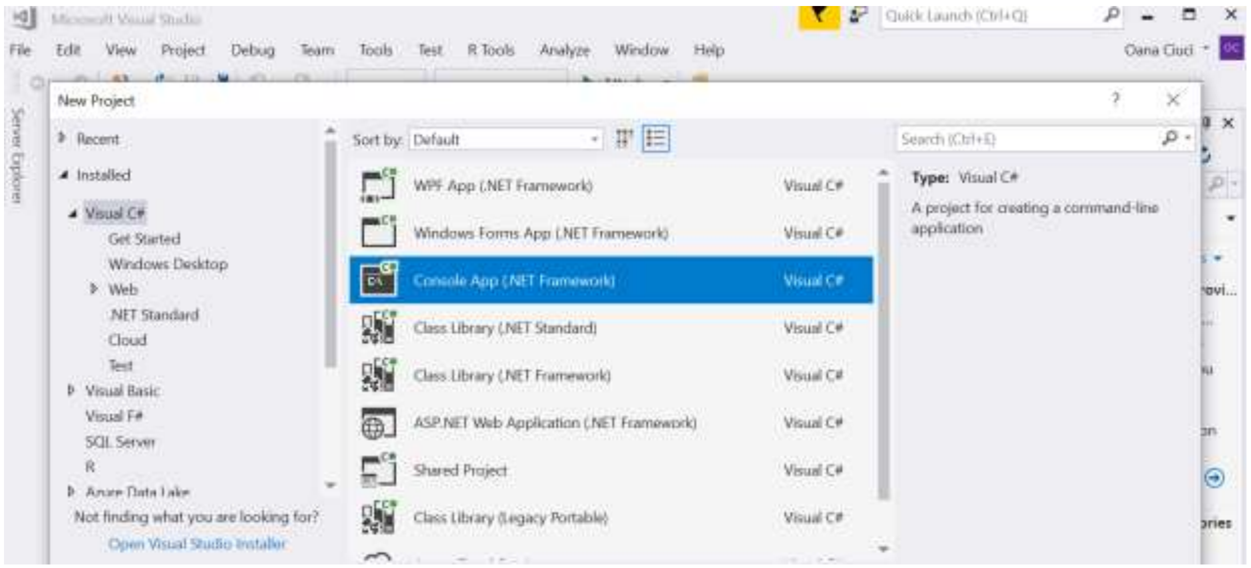
Part 1 - ADO.NET

The ADO.NET Object Model

- ADO.NET
  - Bridging between the front end controls and the back end DB
- 2 central components of ADO.NET classes
  - The .NET Framework Data Provider
  - The DataSet



Part 1 – create a console app

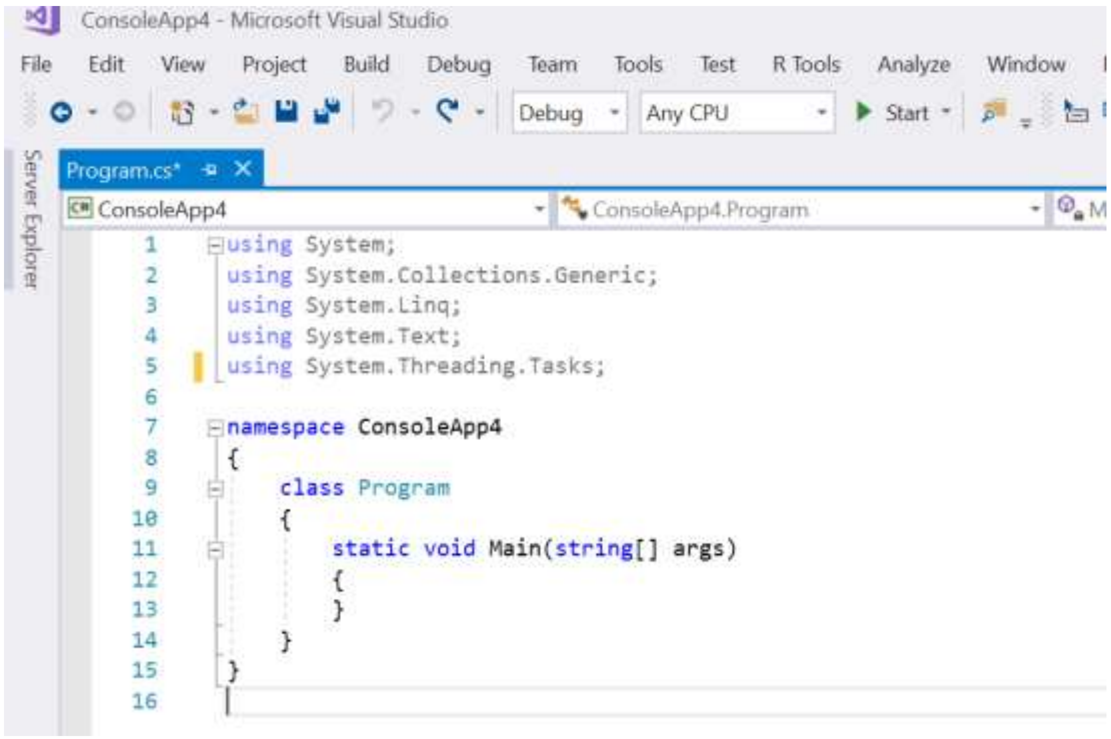


WPF App (.NET Framework)	Visual C#
Windows Forms App (.NET Framework)	Visual C#
Console App (.NET Framework)	Visual C#
Class Library (.NET Standard)	Visual C#
Class Library (.NET Framework)	Visual C#
ASP.NET Web Application (.NET Framework)	Visual C#
Shared Project	Visual C#
Class Library (Legacy Portable)	Visual C#

Type: Visual C#

A project for creating a command-line application

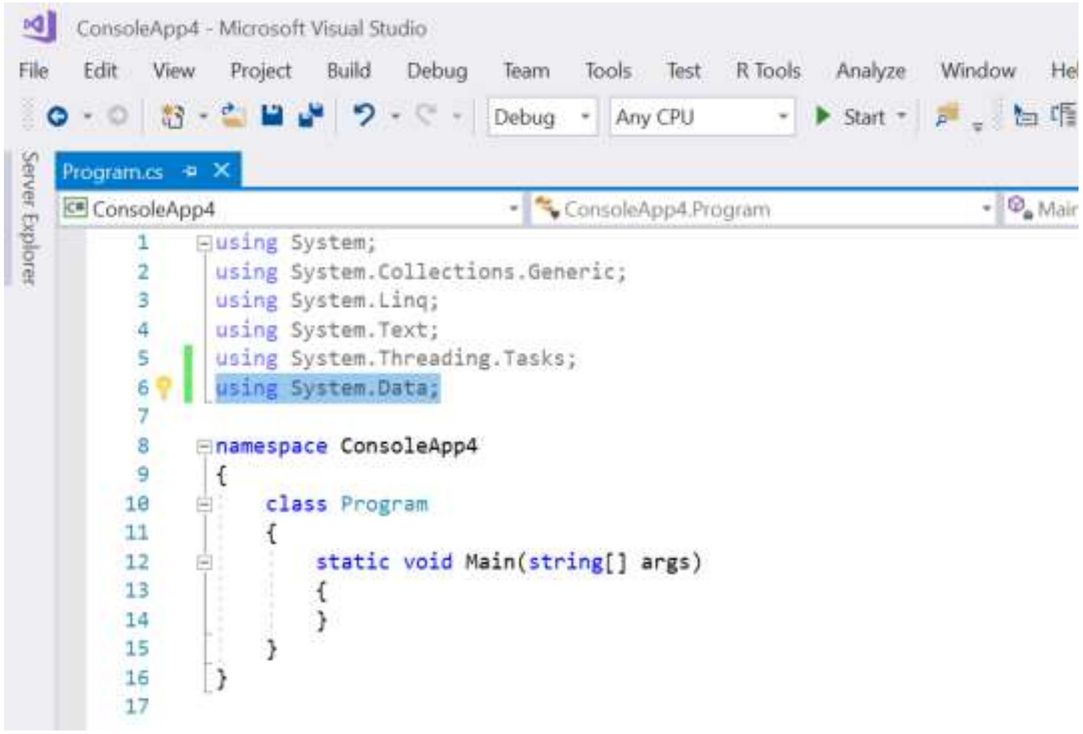
Part 1 – create a console app


ConsoleApp4 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test R Tools Analyze Window
Debug
Any CPU
Start
Program.cs*
ConsoleApp4
ConsoleApp4.Program
Server Explorer

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

namespace ConsoleApp4
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Part 1 – create a console app

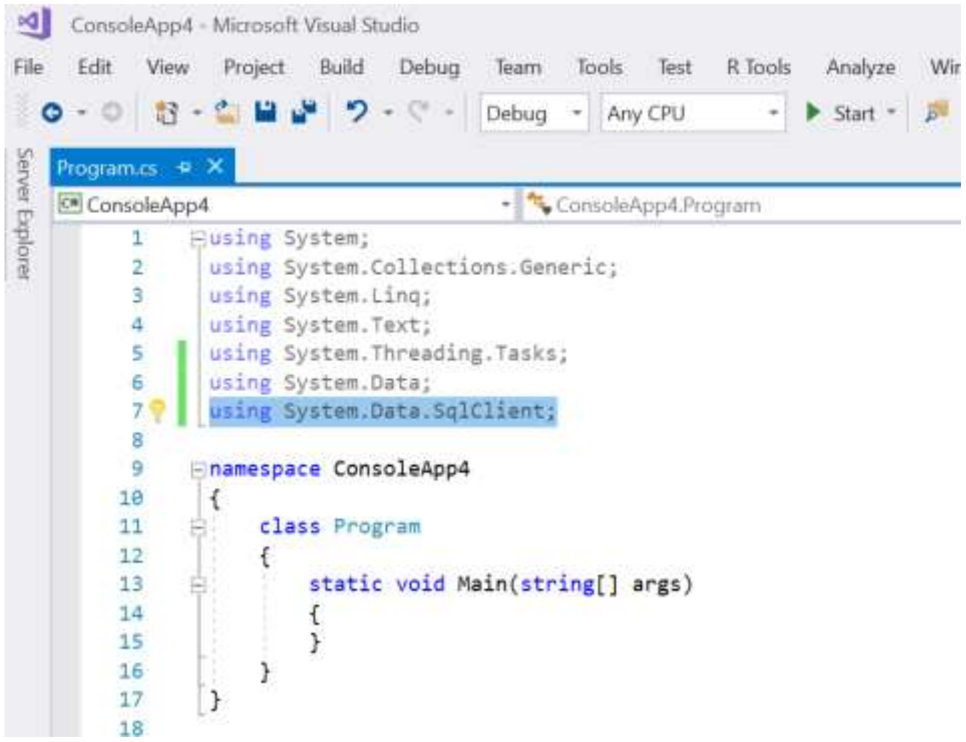
- Packages needed
  - System.Data



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

namespace ConsoleApp4
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Connection to an ADO.NET Database

- Before working with a database, you have to add the SqlClient .NET Data Provider namespace

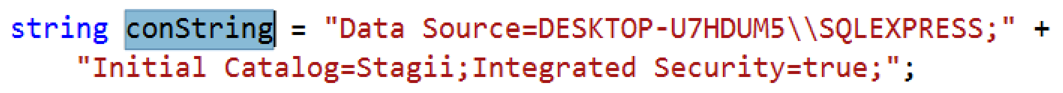


ConsoleApp4 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test R Tools Analyze
Debug
Any CPU
Start
Server Explorer
Program.cs
ConsoleApp4
ConsoleApp4.Program

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

namespace ConsoleApp4
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Connection to an ADO.NET Database

- Declare the **connection string** pointing to your database (SQL Server DB):

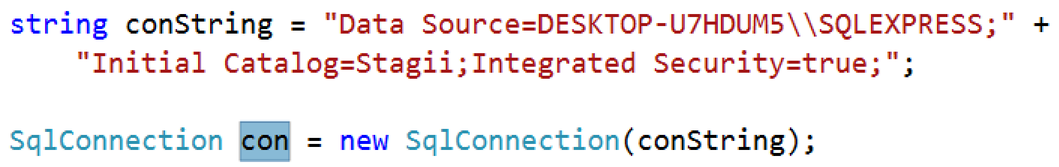


```
string conString = "Data Source=DESKTOP-U7HDUM5\\SQLEXPRESS;" +
    "Initial Catalog=Stagii;Integrated Security=true;";
```

  - Data Source – the DB server name (check on your computer)
  - Initial Catalog – the DB name
  - Integrated Security: true/false depending on the security level you choose for the DB server

Connection to an ADO.NET Database
The SqlConnection Object

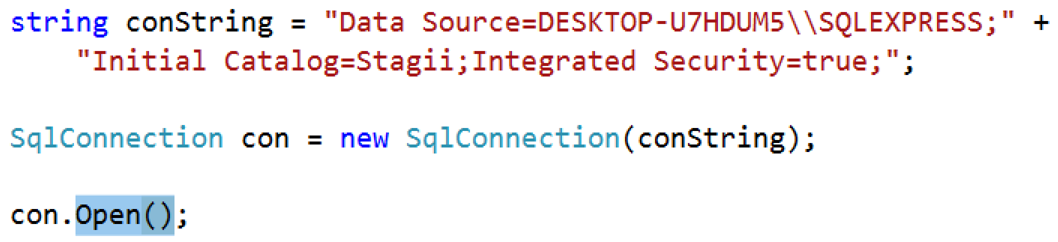
- Create an **SqlConnection object** via which you connect to the DB
- Then, pass the connection string to the SqlConnection object



```
string conString = "Data Source=DESKTOP-U7HDUM5\\SQLEXPRESS;" +
    "Initial Catalog=Stagii;Integrated Security=true;";

SqlConnection con = new SqlConnection(conString);
```

Connection to an ADO.NET Database

- **Open** the connection



```
string conString = "Data Source=DESKTOP-U7HDUM5\\SQLEXPRESS;" +
    "Initial Catalog=Stagii;Integrated Security=true;";

SqlConnection con = new SqlConnection(conString);

con.Open();
```

- Recommendation:
  - Open a connection when you need it, and
  - Close it as soon as you have finished with it

The SqlCommand Object

- You use a **command object** to send SQL statements to the database

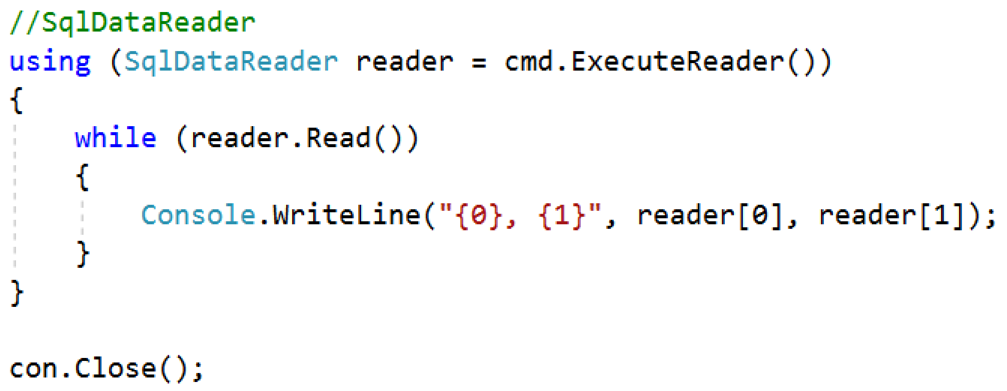
- The connection object is used by command objects so they will know which database to execute the command on

```
//SqlCommand
string strStagii = "SELECT * FROM Stagiu"; //Stagiu(id_stagiu,denumire,nr_ore,nr_credite, id_firma)
SqlCommand cmd = new SqlCommand(strStagii, con);
```

- A command object can be used
  - Alone, to execute a command directly, OR
  - Assign a reference to a command object to a SqlDataAdapter, which holds a set of commands that work on a group of data as described further (during lab & seminar)

The SqlDataReader Object

- The **data reader object** allows you to obtain the results of a SELECT statement from a command object



```
//SqlDataReader
using (SqlDataReader reader = cmd.ExecuteReader())
{
    while (reader.Read())
    {
        Console.WriteLine("{0}, {1}", reader[0], reader[1]);
    }
}

con.Close();
```

- The data returned from a data reader is a fast forward-only stream of data (you can only pull the data from a stream in a sequential manner)
  - If data manipulation is needed then a better alternative is a DataSet object (see further)

The DataSet Object

- DataSet objects are in-memory representations of data
- DataSet objects contain multiple DataTable objects
  - Contain columns and rows, just like normal database tables
  - You can even define relations between tables to create parent-child relationships
- DataSet is an object that is used by all of the Data Providers (does not have a Data Provider specific prefix)

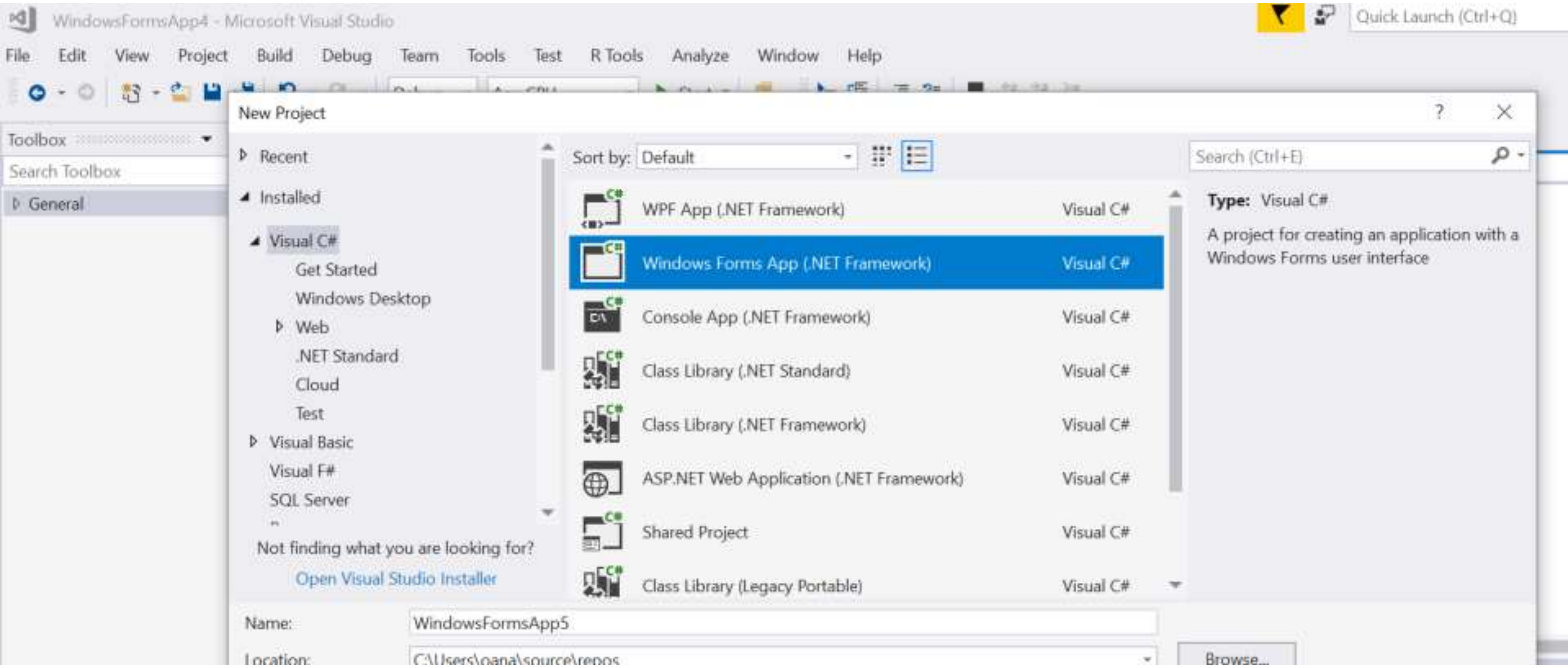
The DataAdapter Object

- The data adapter object
  - Fills a DataSet object when reading the data and
  - Writes in a single batch when persisting changes back to the database
- A data adapter
  - Contains a reference to the connection object
  - It opens and closes the connection automatically when reading from or writing to the database
- The data adapter contains command object references for SELECT, INSERT, UPDATE, and DELETE operations on the data
- You will have a data adapter defined for each table in a DataSet and it will take care of all communication with the database for you
- All you need to do is 'tell' the data adapter when to load from or write to the database

DataSet and DataAdapter Objects

```
//SqlDataAdapter, DataSet
SqlDataAdapter daStagii = new SqlDataAdapter(strStagii, con);
DataSet dset = new DataSet();
//populate dset from the data adapter
daStagii.Fill(dset, "Stagii");
foreach (DataRow pRow in dset.Tables["Stagii"].Rows)
{
    Console.WriteLine("{0}, {1}", pRow["id_stagiu"], pRow["denumire"]);
}
```

Part 2 – create a WindowsForms app (Work in progress version)



WPF App (.NET Framework)	Visual C#
Windows Forms App (.NET Framework)	Visual C#
Console App (.NET Framework)	Visual C#
Class Library (.NET Standard)	Visual C#
Class Library (.NET Framework)	Visual C#
ASP.NET Web Application (.NET Framework)	Visual C#
Shared Project	Visual C#
Class Library (Legacy Portable)	Visual C#

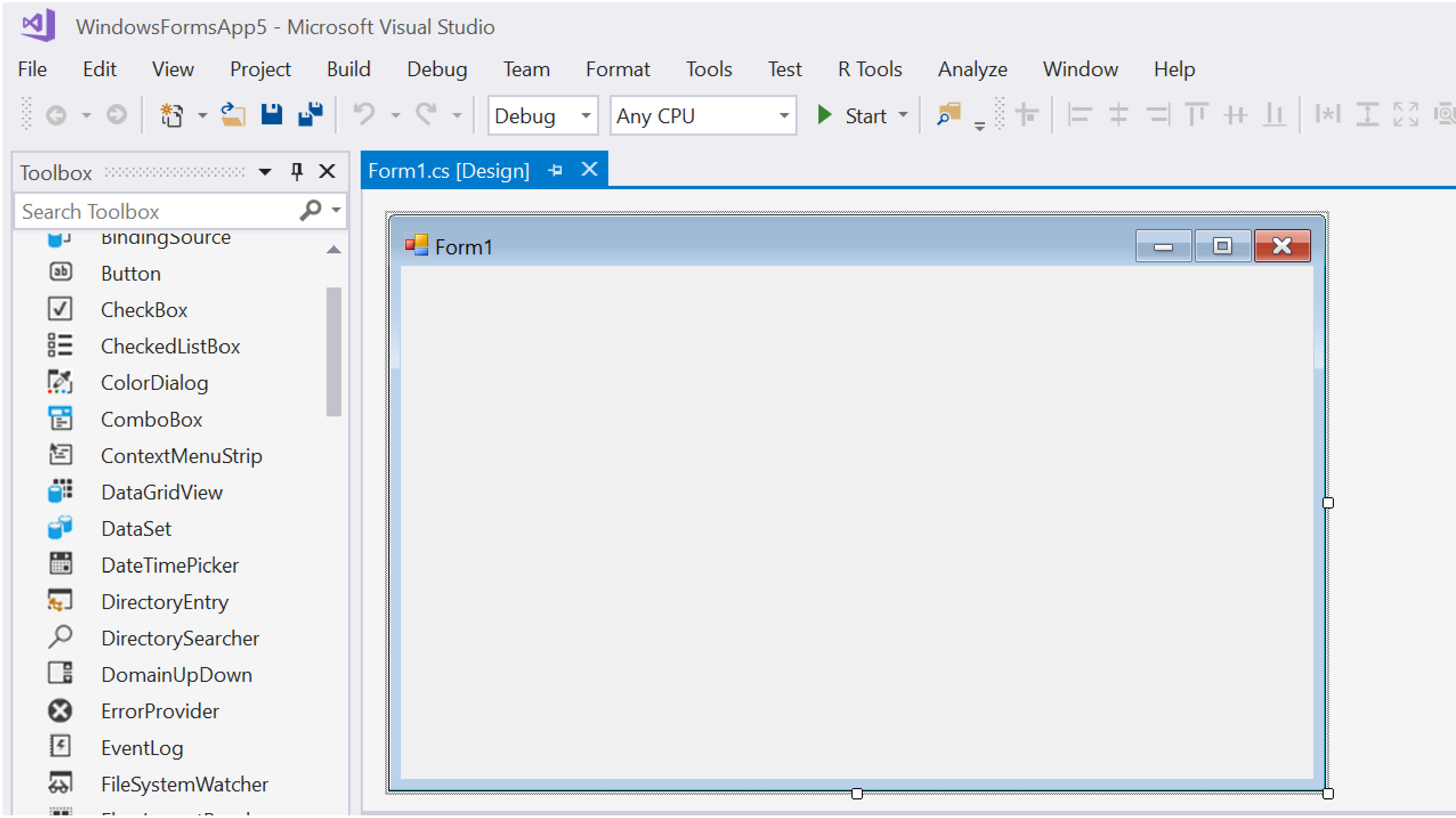
Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: WindowsFormsApp5

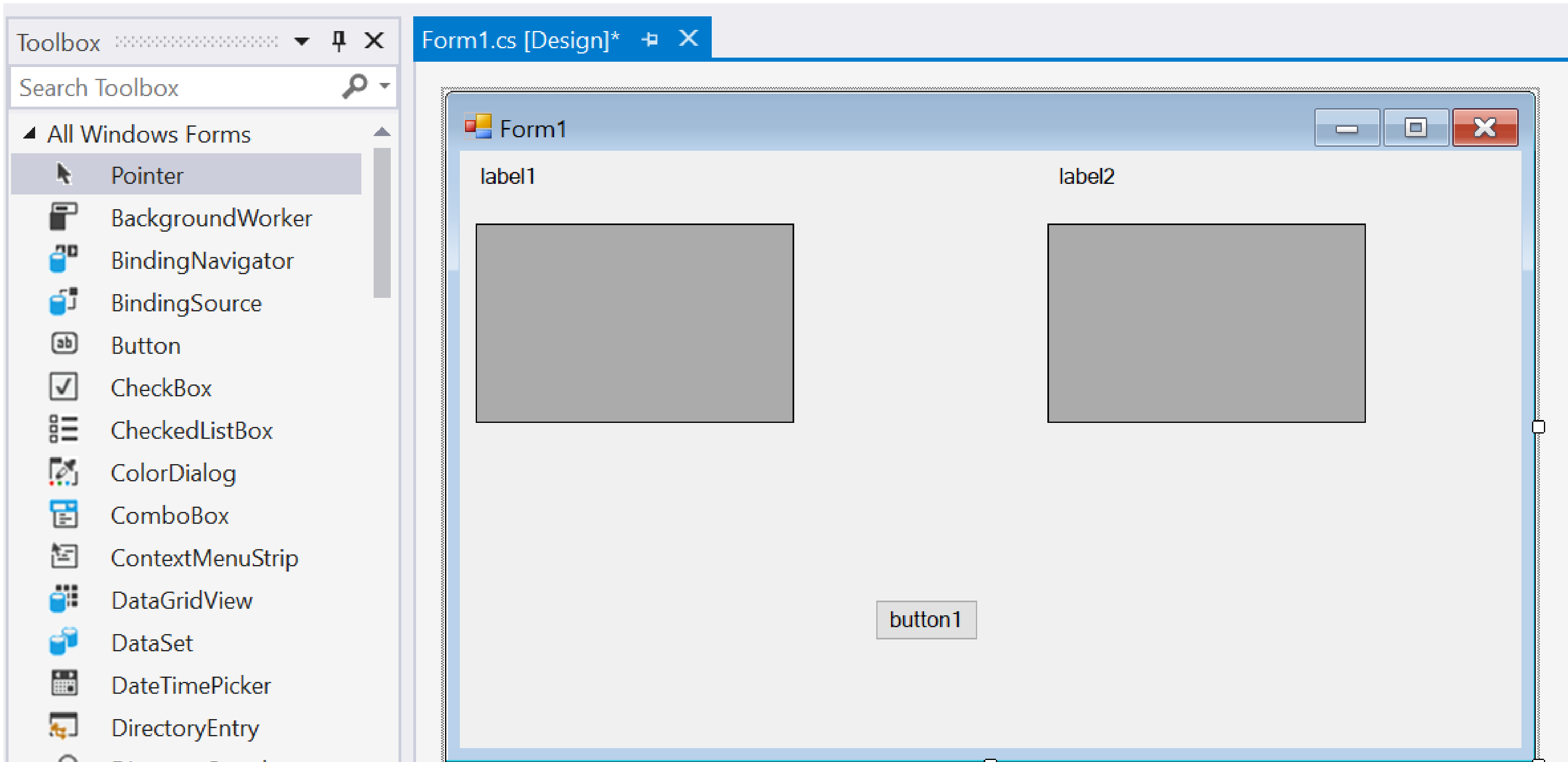
Location: C:\Users\oana\source\repos

Part 2 – create a WindowsForms app



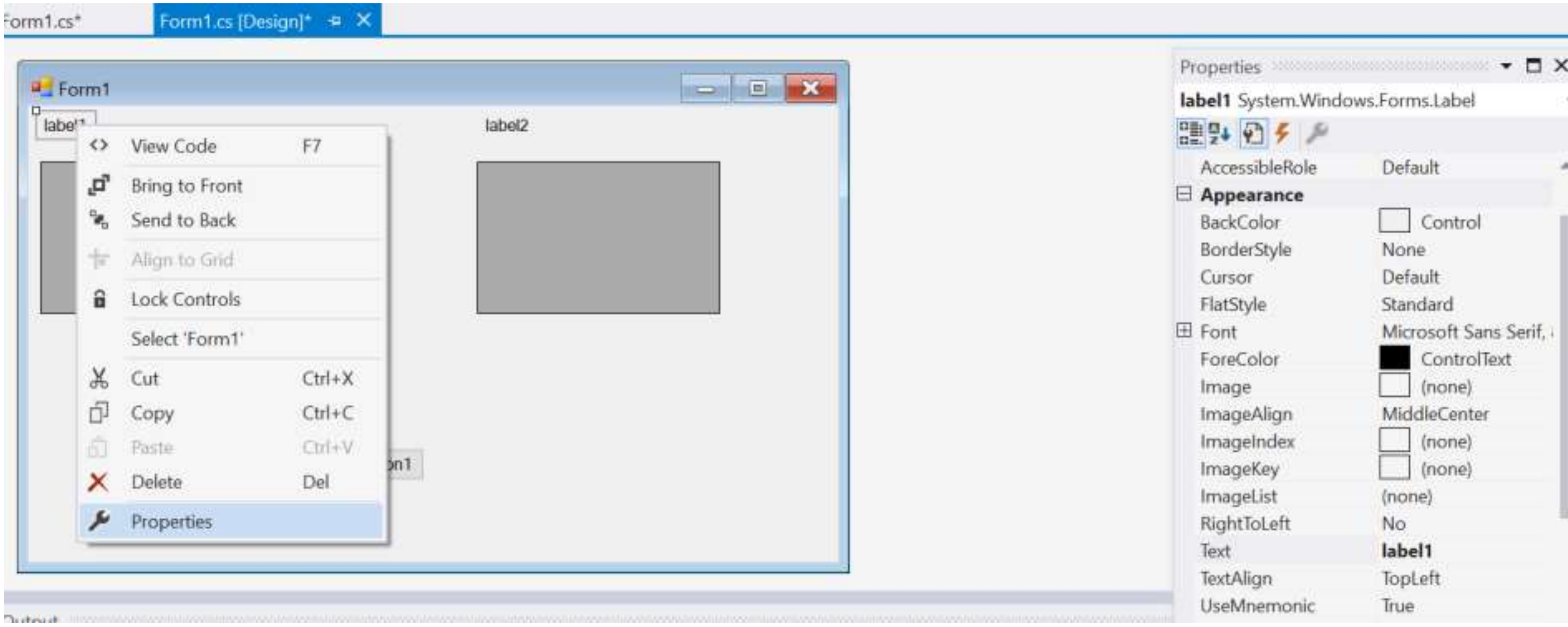
Part 2 – create a WindowsForms app

- Populate the form from the Toolbox: 2 Label controls, 2 DataGridView controls, one Button



Part 2 – create a WindowsForms app

- Customize your controls/GUI (Properties)

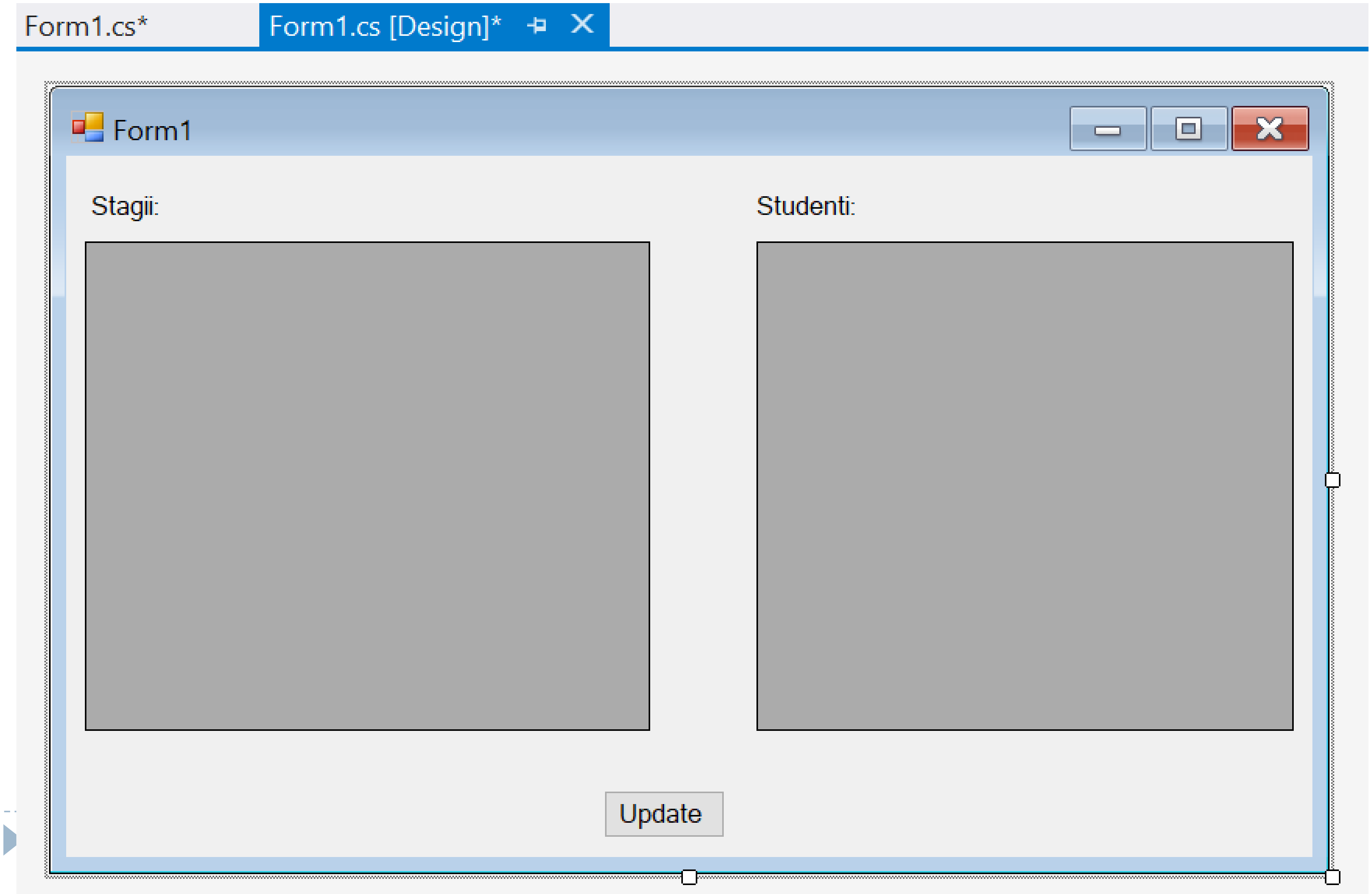


Properties

label1 System.Windows.Forms.Label

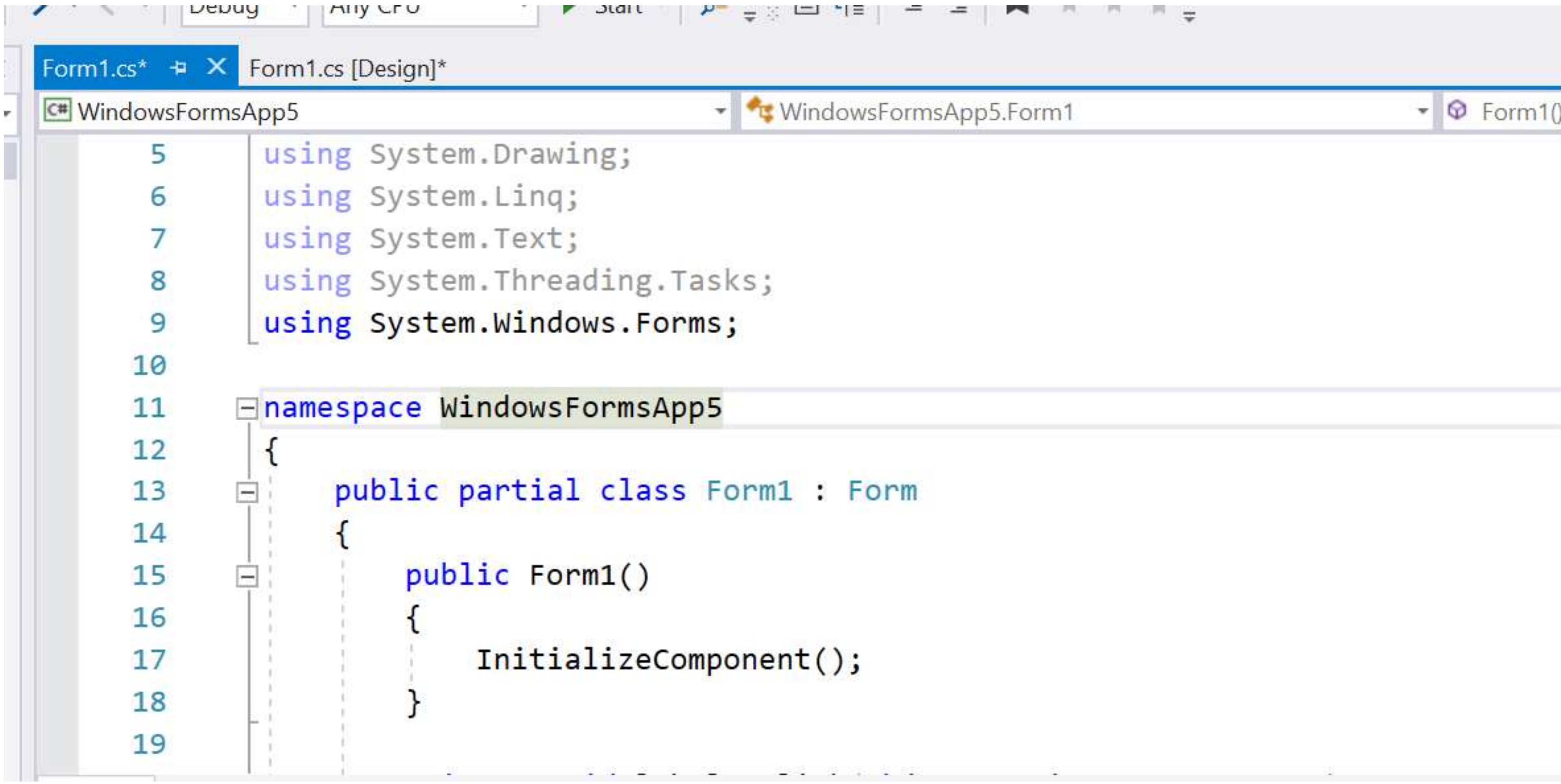
AccessibleRole	Default
Appearance	
BackColor	Control
BorderStyle	None
Cursor	Default
FlatStyle	Standard
Font	Microsoft Sans Serif,
ForeColor	ControlText
Image	(none)
ImageAlign	MiddleCenter
ImageIndex	(none)
ImageKey	(none)
ImageList	(none)
RightToLeft	No
Text	**label1**
TextAlign	TopLeft
UseMnemonic	True

Part 2 – create a WindowsForms app



Part 2 – create a WindowsForms app

- From the Designer, go to code (F7)



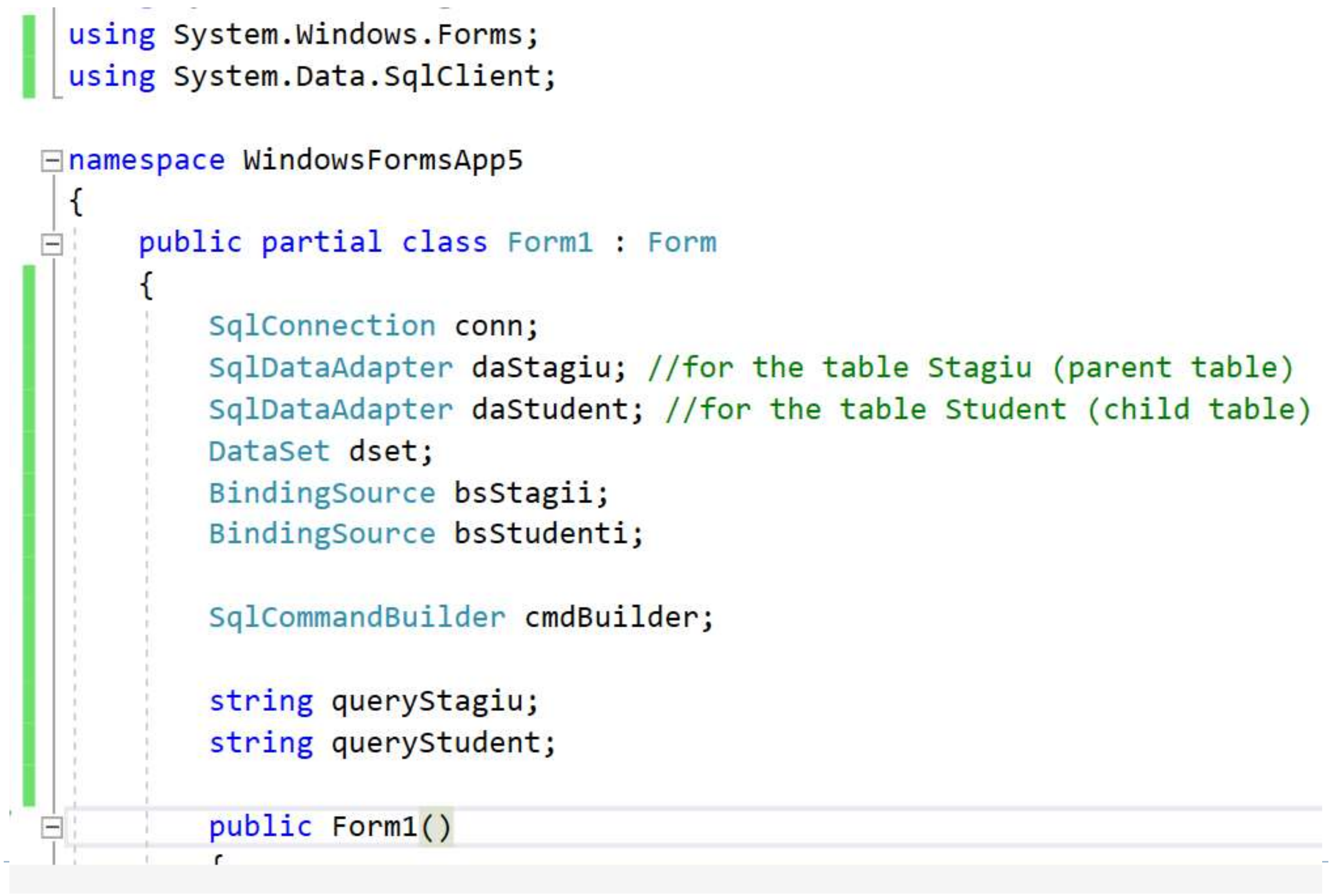
Debug | Any CPU | Start

Form1.cs* | Form1.cs [Design]*

WindowsFormsApp5 | WindowsFormsApp5.Form1 | Form1()

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp5
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

```

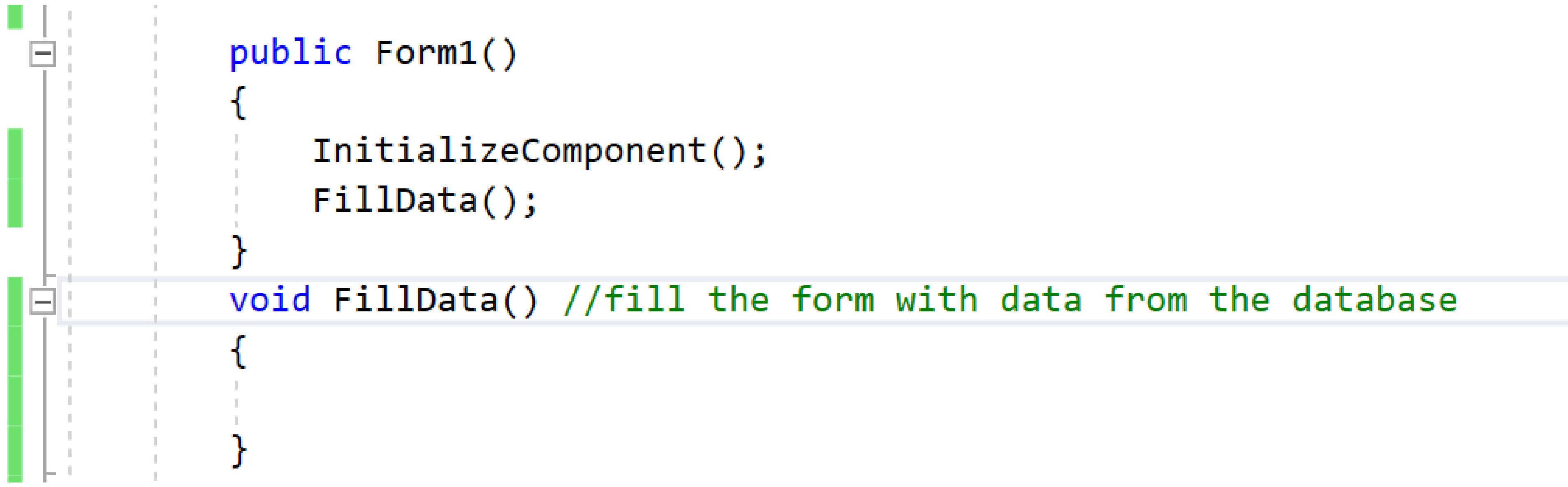
Part 2 – create a WindowsForms app



```
using System.Windows.Forms;
using System.Data.SqlClient;

namespace WindowsFormsApp5
{
    public partial class Form1 : Form
    {
        SqlConnection conn;
        SqlDataAdapter daStagiu; //for the table Stagiu (parent table)
        SqlDataAdapter daStudent; //for the table Student (child table)
        DataSet dset;
        BindingSource bsStagii;
        BindingSource bsStudenti;

        SqlCommandBuilder cmdBuilder;

        string queryStagiu;
        string queryStudent;

        public Form1()
```

Part 2 – create a WindowsForms app

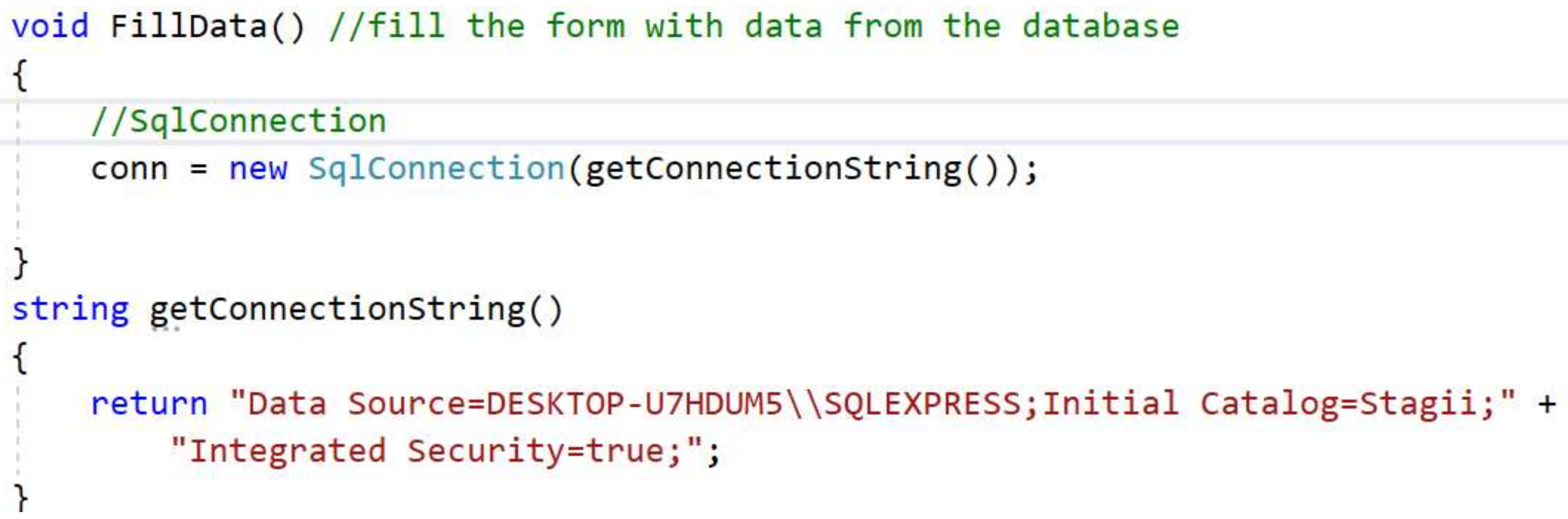
- Implement method FillData()
  - call it from the constructor !



```
public Form1()
{
    InitializeComponent();
    FillData();
}
void FillData() //fill the form with data from the database
{

}
```

Part 2 – create a WindowsForms app

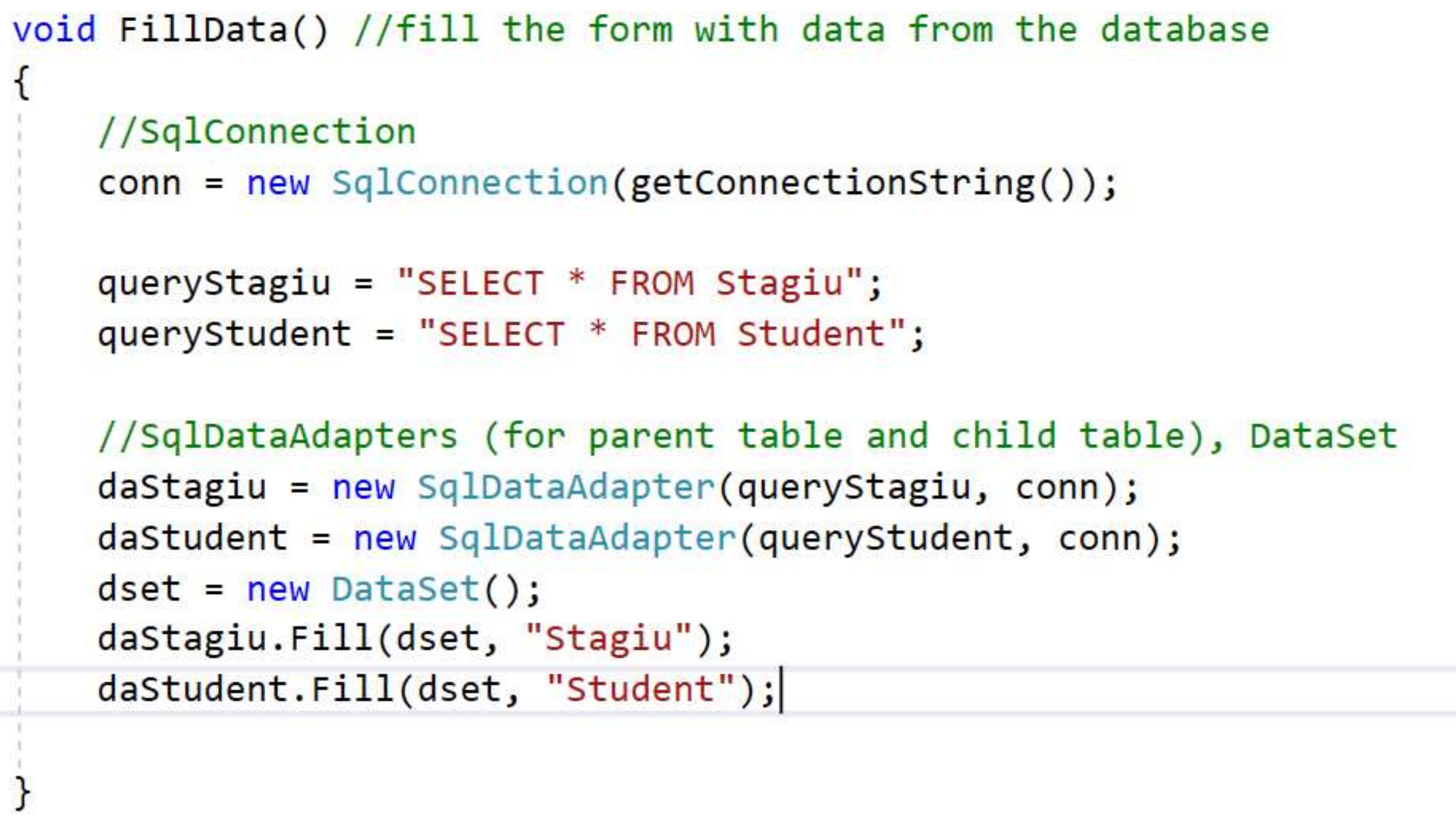
- Implement method FillData()



```
void FillData() //fill the form with data from the database
{
    //SqlConnection
    conn = new SqlConnection(getConnectionString());

}
string getConnectionString()
{
    return "Data Source=DESKTOP-U7HDUM5\\SQLEXPRESS;Initial Catalog=Stagii;" +
        "Integrated Security=true;";
}
```

Part 2 – create a WindowsForms app

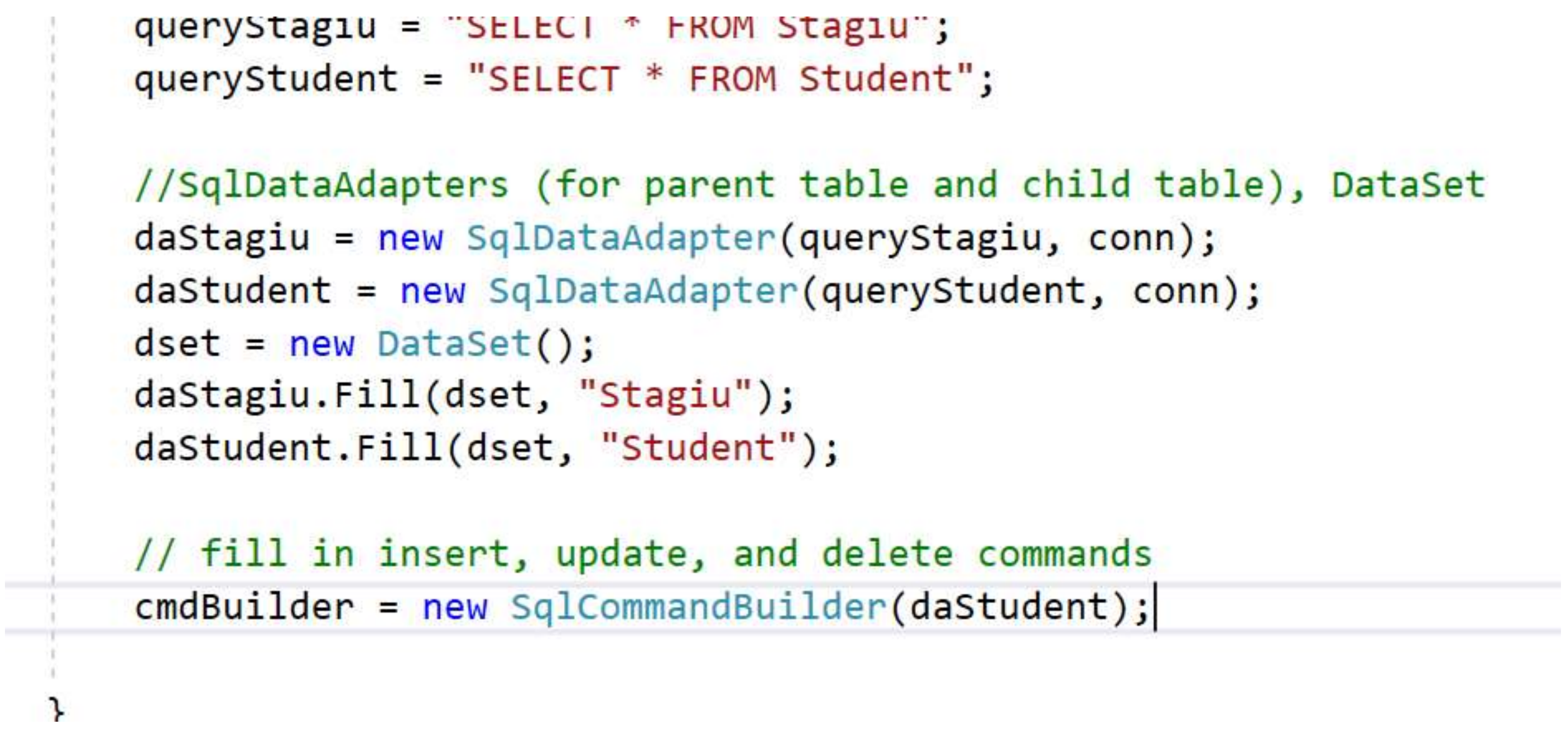
- Implement method FillData()



```
void FillData() //fill the form with data from the database
{
    //SqlConnection
    conn = new SqlConnection(getConnectionString());

    queryStagiu = "SELECT * FROM Stagiu";
    queryStudent = "SELECT * FROM Student";

    //SqlDataAdapters (for parent table and child table), DataSet
    daStagiu = new SqlDataAdapter(queryStagiu, conn);
    daStudent = new SqlDataAdapter(queryStudent, conn);
    dset = new DataSet();
    daStagiu.Fill(dset, "Stagiu");
    daStudent.Fill(dset, "Student");

}
```

Part 2 – create a WindowsForms app

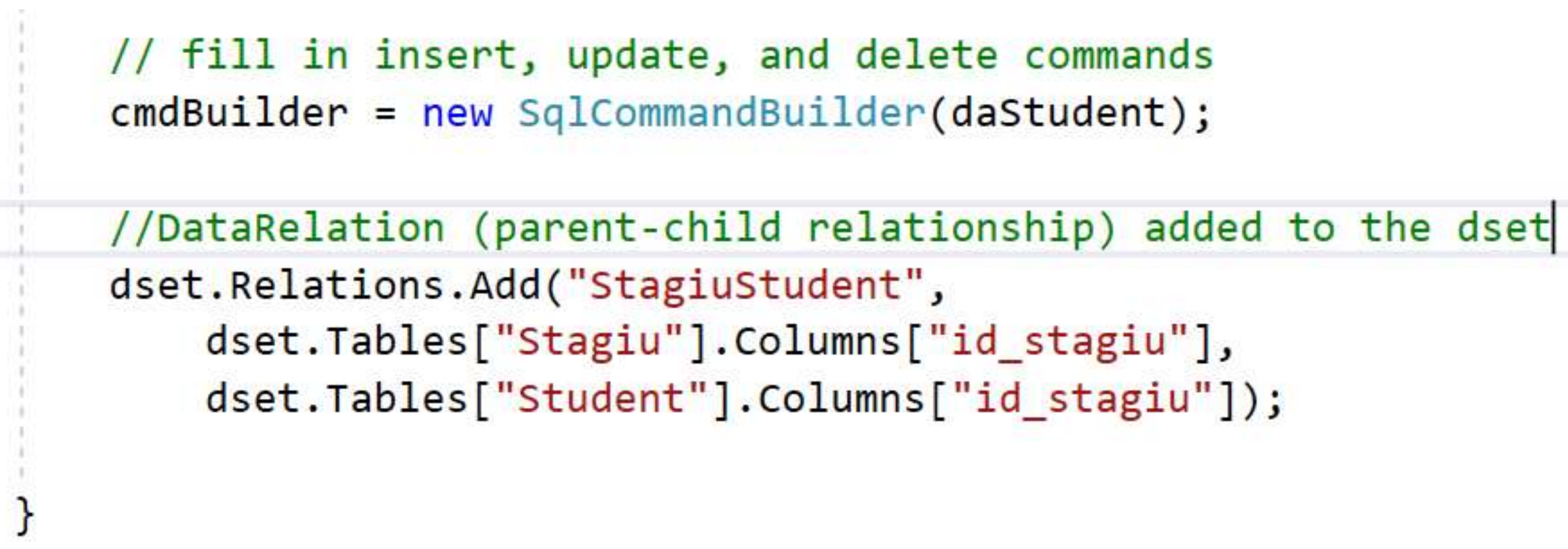
- Implement method FillData()



```
    queryStagiu = "SELECT * FROM Stagiu";
    queryStudent = "SELECT * FROM Student";

    //SqlDataAdapters (for parent table and child table), DataSet
    daStagiu = new SqlDataAdapter(queryStagiu, conn);
    daStudent = new SqlDataAdapter(queryStudent, conn);
    dset = new DataSet();
    daStagiu.Fill(dset, "Stagiu");
    daStudent.Fill(dset, "Student");

    // fill in insert, update, and delete commands
    cmdBuilder = new SqlCommandBuilder(daStudent);

}
```

Part 2 – create a WindowsForms app

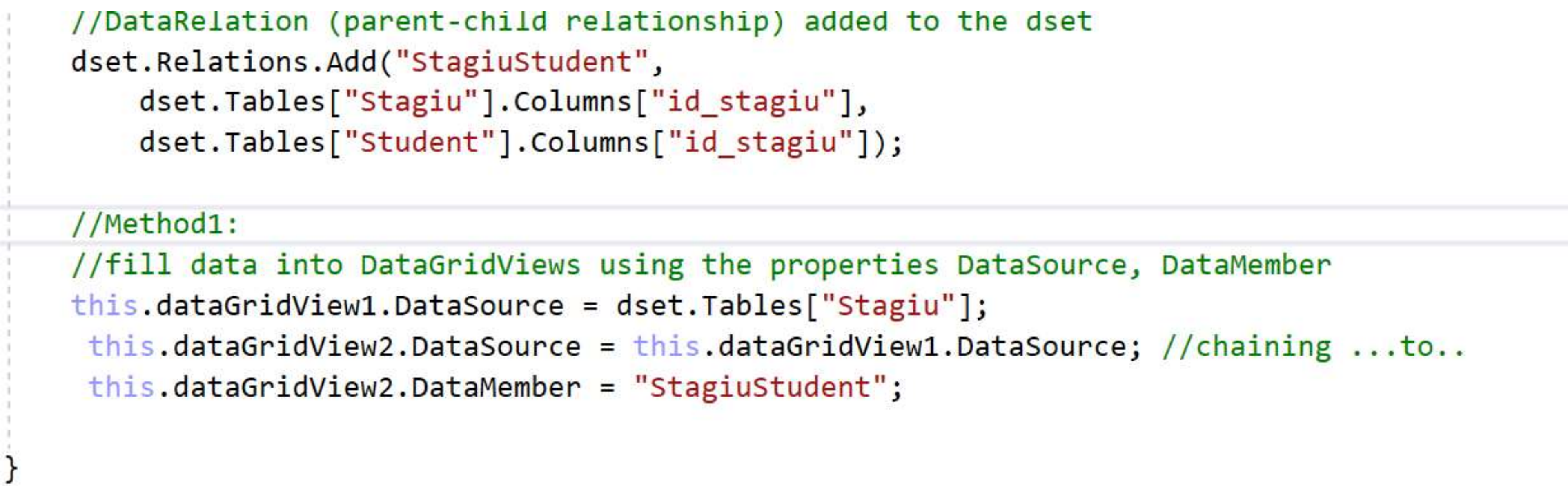
- Implement method FillData()



```
    // fill in insert, update, and delete commands
    cmdBuilder = new SqlCommandBuilder(daStudent);

    //DataRelation (parent-child relationship) added to the dset
    dset.Relations.Add("StagiuStudent",
        dset.Tables["Stagiu"].Columns["id_stagiu"],
        dset.Tables["Student"].Columns["id_stagiu"]);

}
```

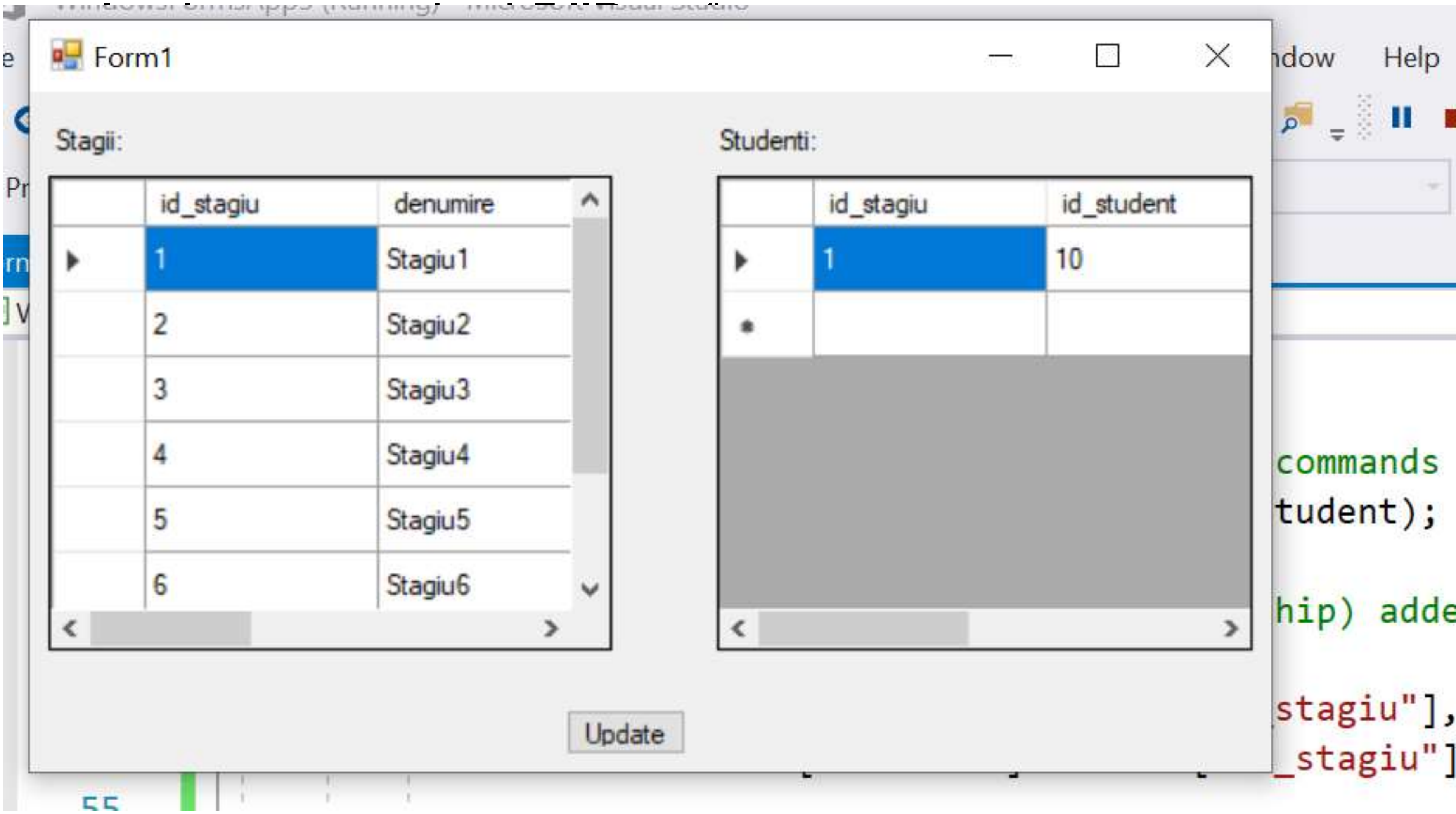
Part 2 – create a WindowsForms app

- Implement method FillData()



```
    //DataRelation (parent-child relationship) added to the dset
    dset.Relations.Add("StagiuStudent",
        dset.Tables["Stagiu"].Columns["id_stagiu"],
        dset.Tables["Student"].Columns["id_stagiu"]);

    //Method1:
    //fill data into DataGridViews using the properties DataSource, DataMember
    this.dataGridView1.DataSource = dset.Tables["Stagiu"];
     this.dataGridView2.DataSource = this.dataGridView1.DataSource; //chaining ...to..
     this.dataGridView2.DataMember = "StagiuStudent";

}
```

Part 2 – create a WindowsForms app

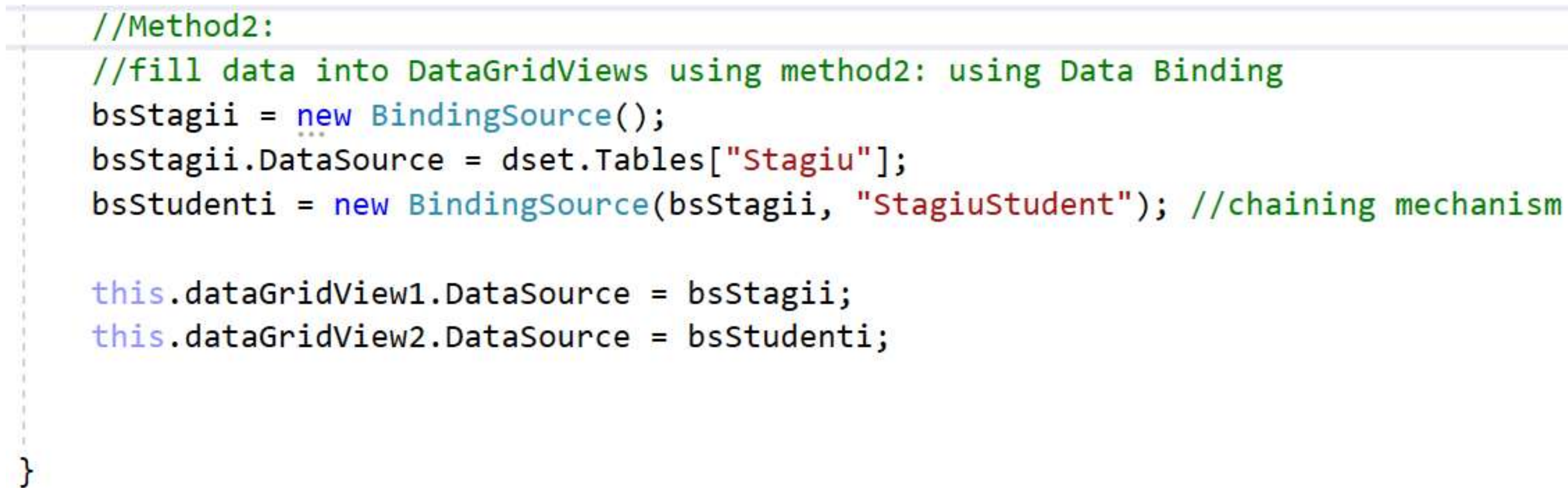


	id_stagiu	denumire
▶	1	Stagiu1
	2	Stagiu2
	3	Stagiu3
	4	Stagiu4
	5	Stagiu5
	6	Stagiu6

	id_stagiu	id_student
▶	1	10
*		

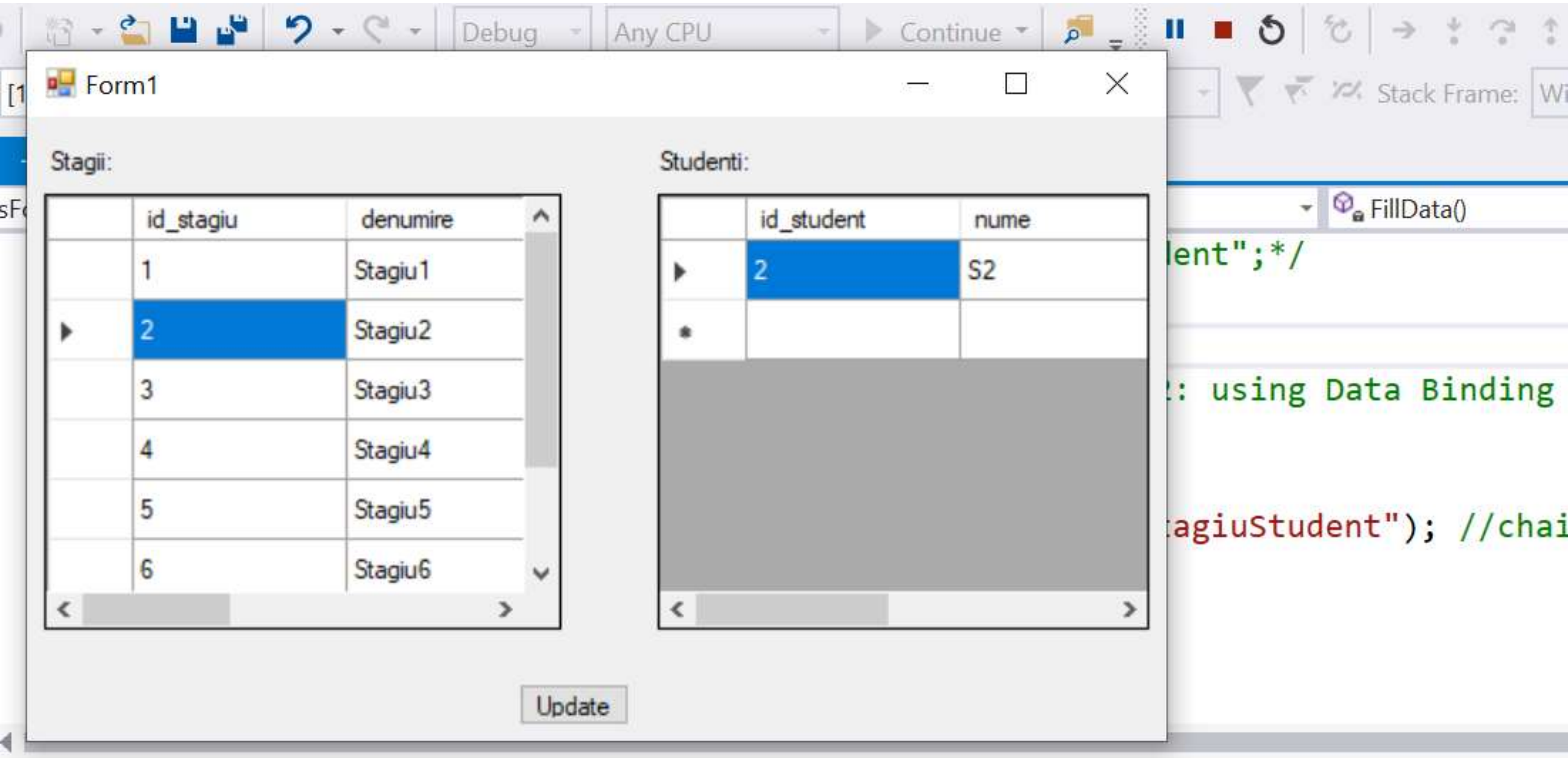
Part 2 – create a WindowsForms app

- Implement method FillData()



```
    //Method2:
    //fill data into DataGridViews using method2: using Data Binding
    bsStagii = new BindingSource();
    bsStagii.DataSource = dset.Tables["Stagiu"];
    bsStudenti = new BindingSource(bsStagii, "StagiuStudent"); //chaining mechanism

    this.dataGridView1.DataSource = bsStagii;
    this.dataGridView2.DataSource = bsStudenti;


}
```

Part 2 – create a WindowsForms app



Form1

Stagii:

	id_stagiu	denumire
	1	Stagiu1
▶	2	Stagiu2
	3	Stagiu3
	4	Stagiu4
	5	Stagiu5
	6	Stagiu6

Studenti:

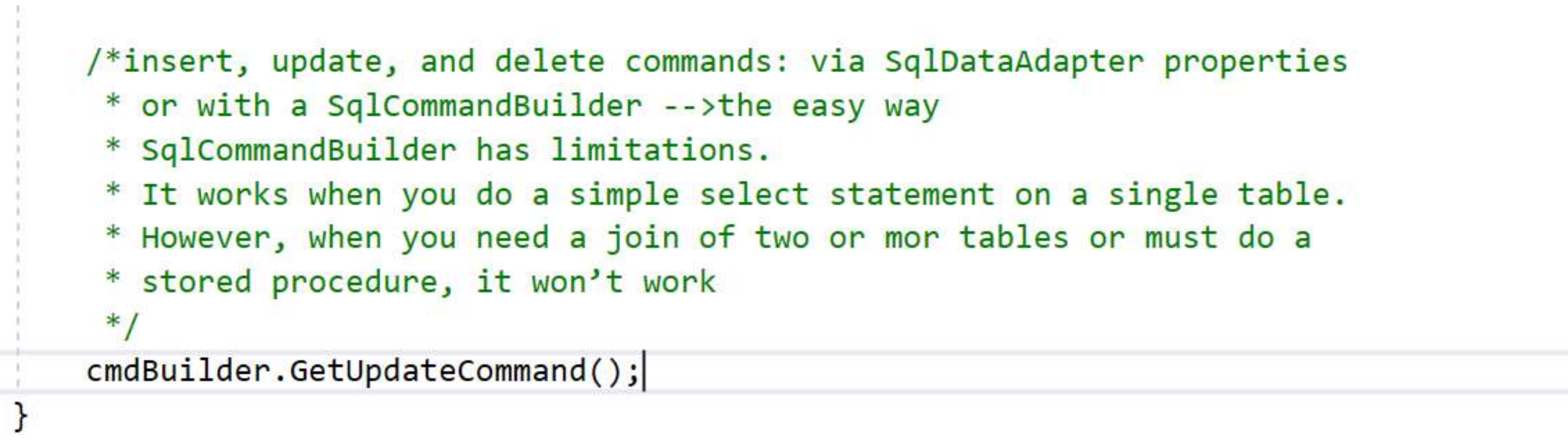
	id_student	nume
▶	2	S2
*		

Update

```
lent";*/

: using Data Binding

agiuStudent"); //chai
```

Part 2 – create a WindowsForms app

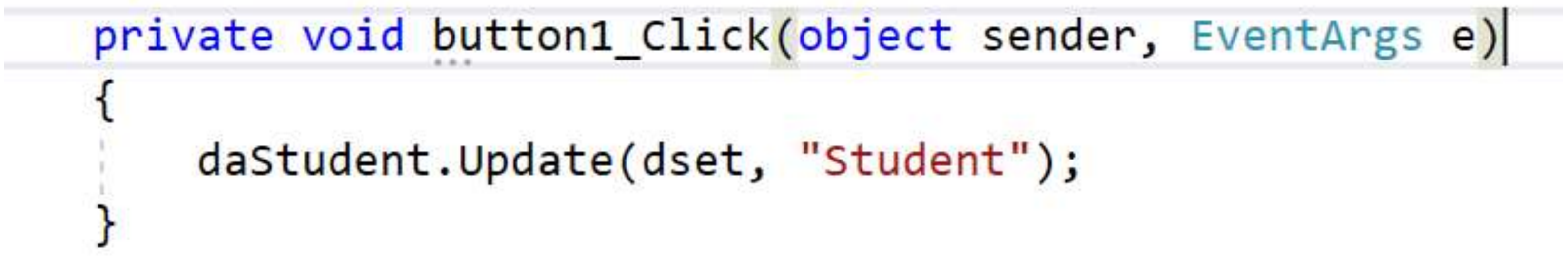
- Implement method FillData()



```
    /*insert, update, and delete commands: via SqlDataAdapter properties
     * or with a SqlCommandBuilder -->the easy way
     * SqlCommandBuilder has limitations.
     * It works when you do a simple select statement on a single table.
     * However, when you need a join of two or mor tables or must do a
     * stored procedure, it won’t work
     */
    cmdBuilder.GetUpdateCommand();
}
```

Part 2 – create a WindowsForms app

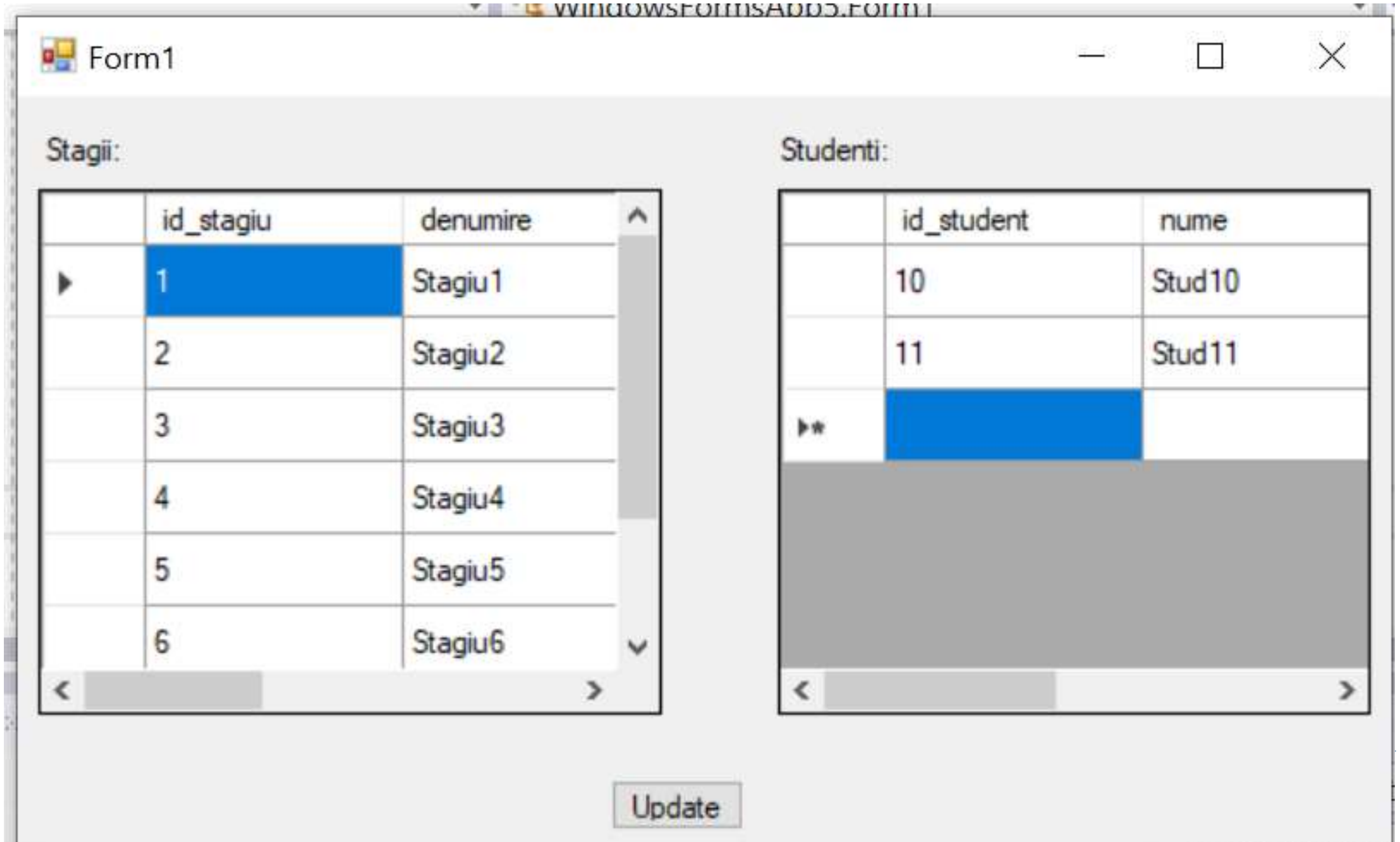
- Implement "Update" button action



```
private void button1_Click(object sender, EventArgs e)
{
    daStudent.Update(dset, "Student");
}
```

Part 2 – create a WindowsForms app

- Check how "Update" button is working (for Insert, Update, Delete)



WindowsFormsApp5.Form1

Form1

Stagii:

	id_stagiu	denumire
▶	1	Stagiu1
	2	Stagiu2
	3	Stagiu3
	4	Stagiu4
	5	Stagiu5
	6	Stagiu6

Studenti:

	id_student	nume
	10	Stud10
	11	Stud11
▶*		

References

- *http://csharp-station.com/Tutorial/AdoDotNet/Lesson01*
- *http://www.codeproject.com/Articles/8477/Using-ADO-NET-for-beginners*
- *http://www.codeproject.com/Articles/24656/A-Detailed-Data-Binding-Tutorial*