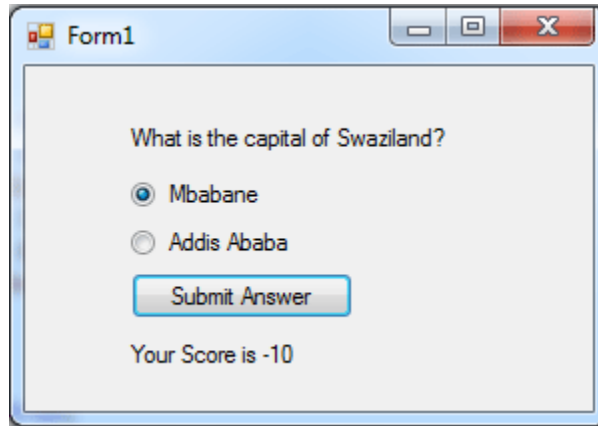


World Capitals – A new Exciting Game!

Our task is to create a simple game. The game will ask user the capital of a random country and provide two options to pick from. One of the options is the correct answer. If user submits the correct answer, his score is increased 10 points, decreased 10 points otherwise.

Here is a view of the game:



Form1

What is the capital of Swaziland?

☒ Mbabane

☐ Addis Ababa

Submit Answer

Your Score is -10

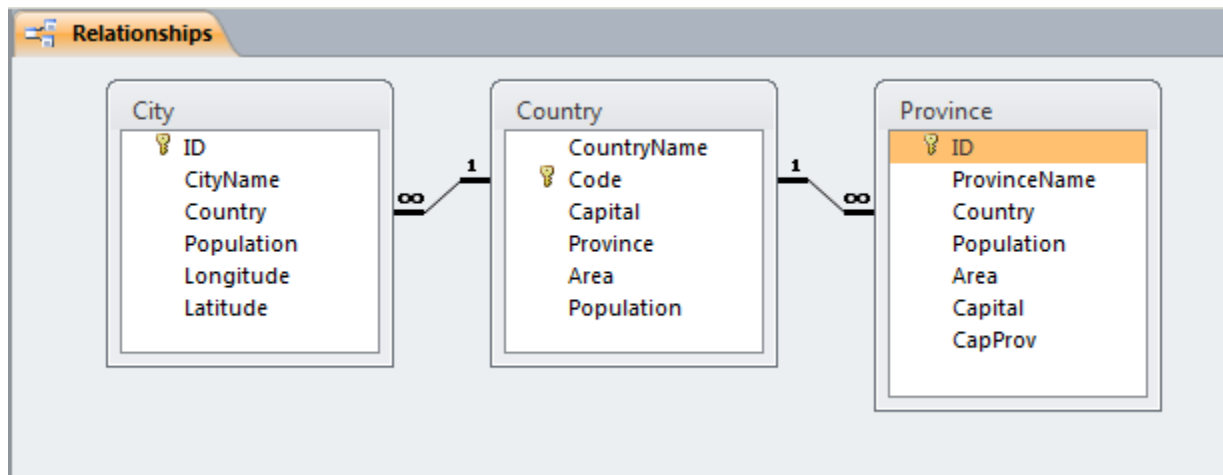
This exercise will give you further experience with C# Windows Forms applications. It will also demonstrate how to connect and extract information from an external database. In our case, we will use MS Access database.

Download the Database

There is a free database called MONDIAL on the web from about.com.

<http://databases.about.com/od/sampleaccessdatabases//MONDIAL.accdb>

Here is the relationship view of this database. We will only need to use Country table as the name of the country and the capital of the country is all we need for this game.

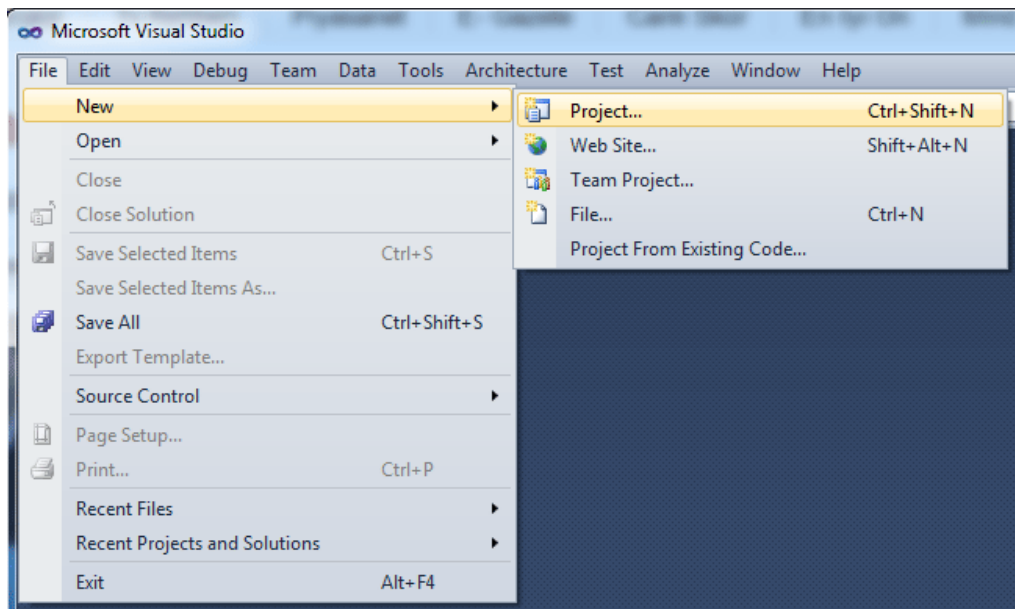


In order to connect to a MS Access database, you need to install “Microsoft Access Database Engine 2010 Redistributable” from Microsoft on your machine:

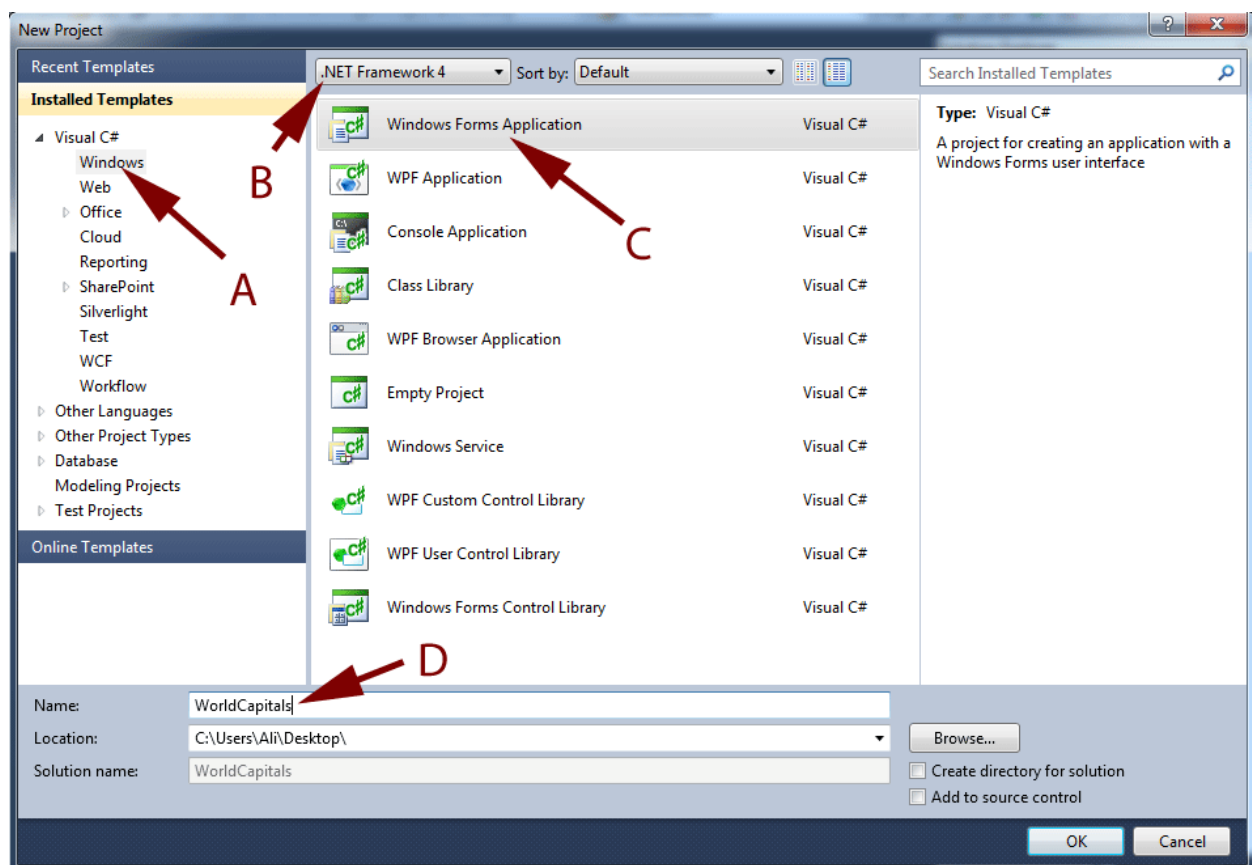
<http://www.microsoft.com/en-us/download/details.aspx?id=13255>

Step 1: Create a new Windows Forms Application

In Visual Studio, from the menu “File > New > Project”.



Select Visual C# Windows Forms Application. Name the Project WorldCapitals. Hit OK to create your project.



Step 2 – Design Your Applications User Interface

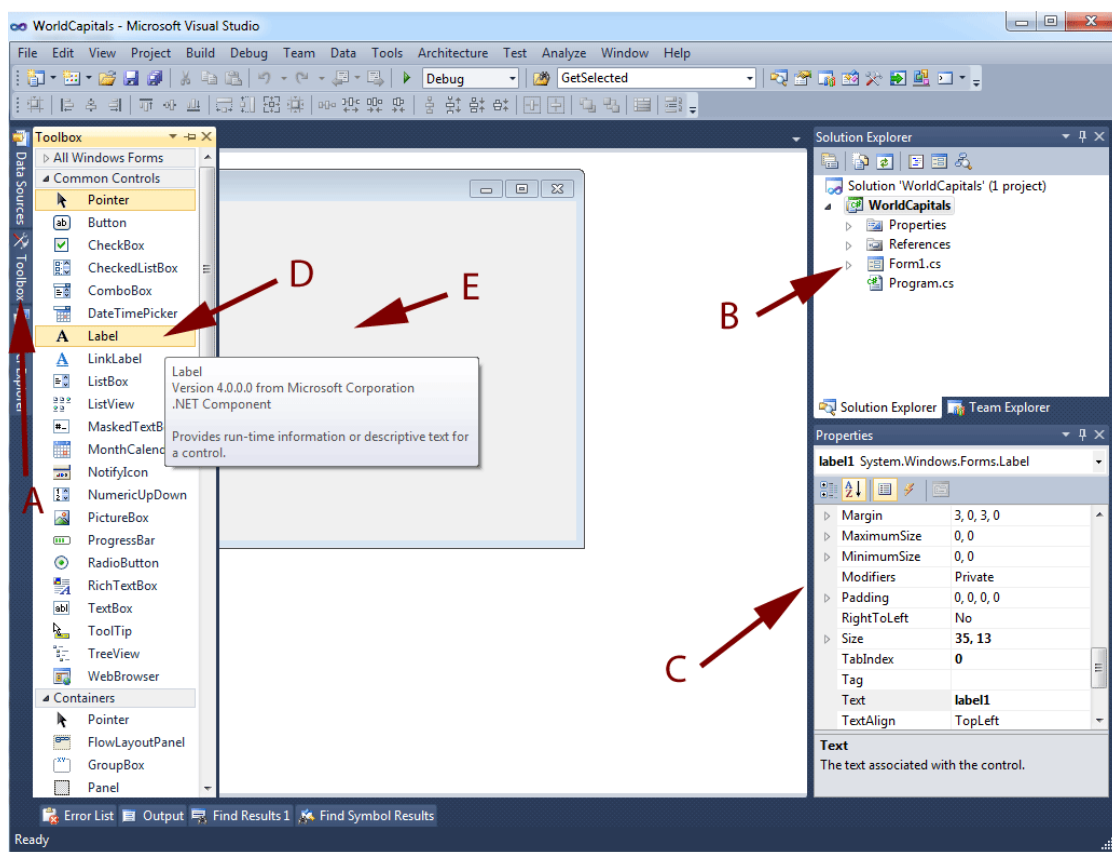
Once the project is created, you should have a view like the one shown below.

On the right hand side you see the Solution Explorer [B], which lists the files in your project. If you do not see Solution Explorer, from the menu, go to “View > Solution Explorer to activate it.

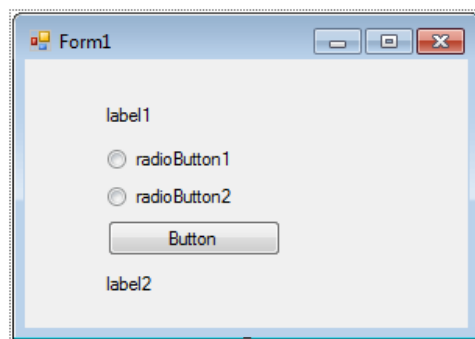
Again on the right hand side, at the bottom, is the Properties Explorer [C]. You can edit the properties of different user interface controls using this explorer. If you do not see Property Explorer, from the menu go to View > Properties Window to activate it.

On the left hand side, you see the Toolbox [A]. This pane contains various user interface controls commonly used in Windows programs. Our first task is to create two labels and two radio buttons on the window.

To add the first label, find the label control [D] in the toolbox [A], and drag and drop it onto the Windows Form [E].



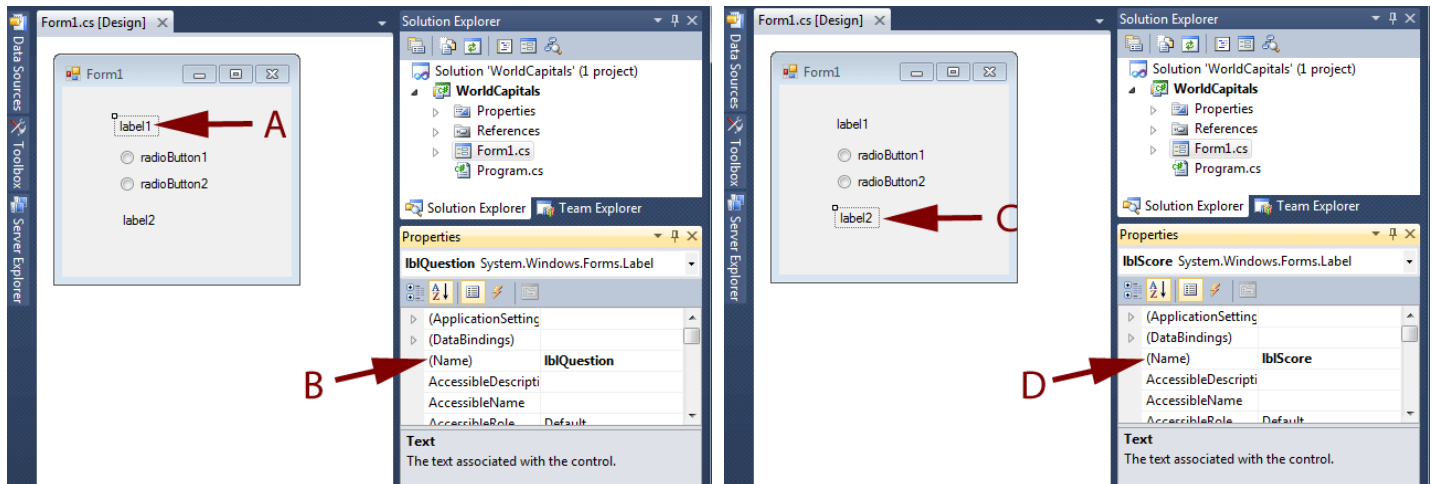
Then add one more label, two radio buttons and a button, one by one, from the Toolbox on to the Windows Form. After re-arranging and resizing the controls, your form should look like this:



Step 3 – Name the User Controls

Naming these two labels and the option box is very important. We will refer to these controls in our code using their names.

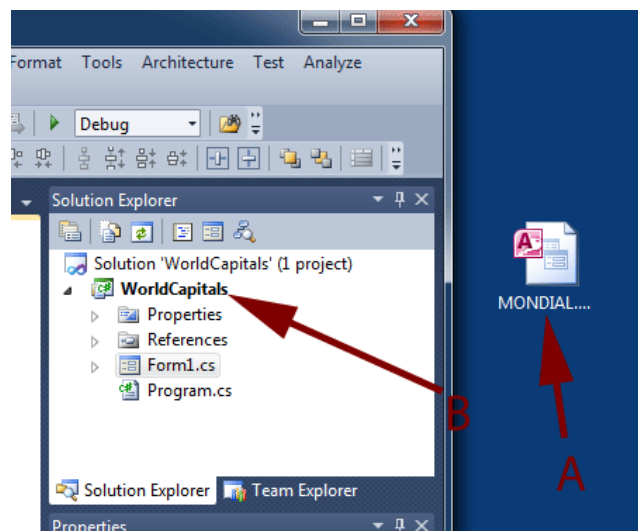
To name the first label, click on label1 [A] and then change its name in the properties explorer to **lblQuestion** [B]. Do the same for label2. Click on it [C], and set its name property to **lblScore** [D].



For the radio buttons, set the name of the first radio button to **optChoice1**. The second one should be named **optChoice2**. Change name property of the button to **btnSubmitAnswer**, also change the Text property of the button to Submit Answer.

Step 4 – Add the Database to Your Project

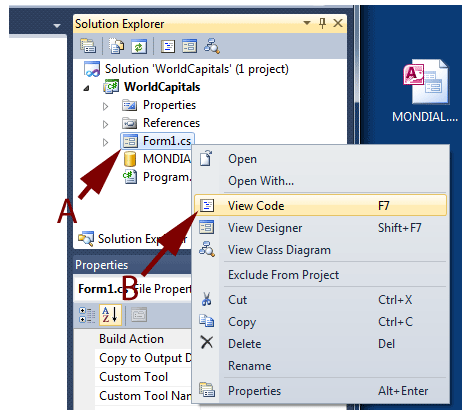
Add the Mondial database to your project. You can do that by dragging and dropping the database file [A] into your project [B]. Once the database is added, Data Source Configuration Wizard will show up. Simply cancel out of this wizard. We will do our connection to the database programmatically.



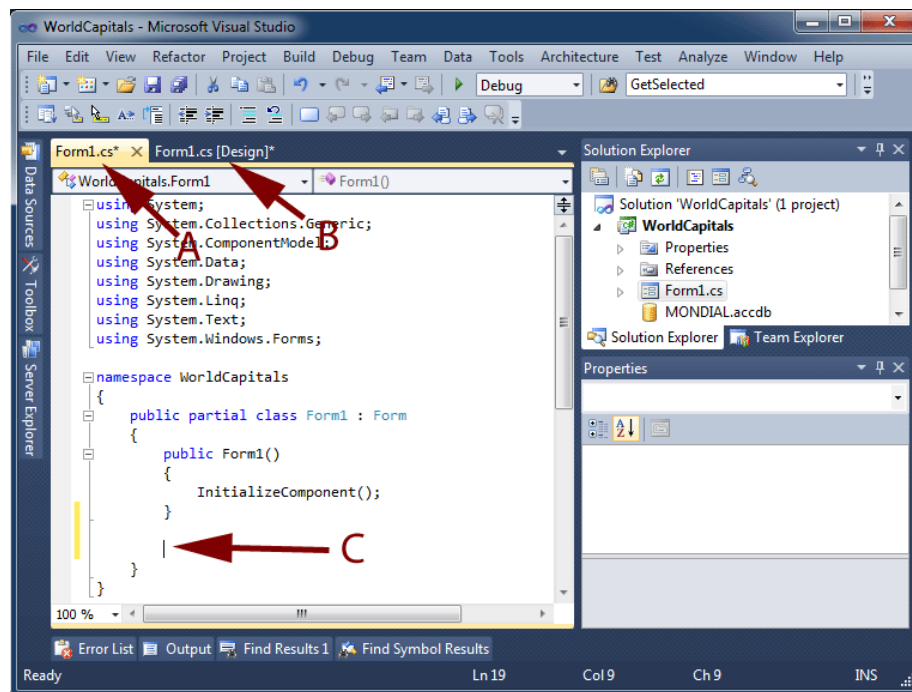
Step 5 – Time to Do Some Coding!

We have the user interface and the data. Let's see if we can put this whole thing together. There are two parts to this program. First part is to pose a question to the user based on the information in the database. The second part is to evaluate user's answers and keep a score based on that.

We will write two functions to handle each part. First, let's get to the code view of this Windows Form. Right click on Form1 in Solution Explorer and hit "View Code".



Now you have the code view of the form. Notice that you can always switch between the code view [A] and the design view [B]. We will write our code right after Form1's constructor [C].



The first function displays a question to the user. Let's call this function AskQuestion(). We don't need to send any input arguments into this function, nor are we looking for an output, so it will be a void function. We will write the code that displays a question to the user inside AskQuestion() function.

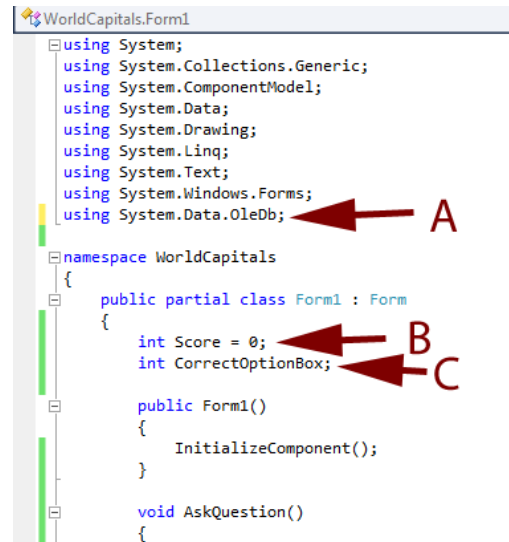
```
namespace WorldCapitals
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        void AskQuestion()
        {
            |
        }
    }
}
```

A red arrow labeled 'A' points to the line of code immediately following the 'AskQuestion()' method, indicating where to write the code.

Before we start writing the code inside AskQuestion function, let's add couple of convenience code. First let's add the namespace of the data access objects at the very beginning of the code file [A]. Let's also declare two global variables. I know I had told you many times that global variables are evil – that you should avoid at all cost. Well, sometimes they are necessary evils.

Let's introduce two global variables. First one is to hold on to the score [B]. This number will increase or decrease based on user's answer. The second one is to tell me which one of the option box has the correct answer [C]. When we display the question to user, we will set this variable so that the function that evaluates user's answer would know which one of the option boxes contained the correct answer.



```
WorldCapitals.Form1
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb; ← A

namespace WorldCapitals
{
    public partial class Form1 : Form
    {
        int Score = 0; ← B
        int CorrectOptionBox; ← C

        public Form1()
        {
            InitializeComponent();
        }

        void AskQuestion()
        {
```

Step 6 – AskQuestion() Function

Now we are ready for AskQuestion() function. In order to ask a question, we need to take a look at the countries in the database and pick a random country. User is to choose the capital of this country among the two options provided. So the two options we provide are: 1- the capital of the country we randomly picked, 2- the capital of another randomly picked country as a decoy.

We are using ADO.NET to connect to the Access database. ADO.NET can be used for wide variety of databases using a connection string. You can always Google for connection strings for the specific database system you are working with. This one instructs ADO.NET to connect to a Microsoft Access database named MODIAL.accdb:

```
Provider=Microsoft.ACE.OLEDB.12.0;Data Source=MONDIAL.accdb;Jet OLEDB:Database Password=password
```

Using a DataSet and OleDbDataAdapter objects, we can retrieve the data we need from the Access database. Below, you can see the full source code for AskQuestion() function. Notice the SELECT query executed on the database, and the returned data from this query is stored in the DataSet called data.

At **step C**, we get the total number of countries in the database. We need this beforehand so that we can create random numbers between 1 and the number of countries. **Step D** does exactly this. Lists/Arrays in C# are zero based, so the number we generate is between 0 and number of countries - 1.

At **Step E**, we are getting the country name and the capital of the correct answer along with another decoy capital. Now that we have all the information we need, at **step F** we display the question; and **Step G**, we set the options for user to choose. Notice that we need to randomize which option box will have the correct answer and setup the option boxes accordingly. At the same time, we set CorrectOptionBox for later use.

```

void AskQuestion()
{
    // Step A - build a connection string to be used to connect to the database file
    string connStr = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=MONDIAL.accdb;Jet OLEDB:Database Password=password";

    // Step B - Using the .NET ADO connect to the database
    // and retrieve all the countries using a SELECT query
    DataSet data = new DataSet();
    OleDbDataAdapter adap = new OleDbDataAdapter("SELECT * FROM Country", connStr);
    adap.Fill(data);

    // Step C - get the number of countries in the database
    int countryCount = data.Tables[0].Rows.Count;

    // Step D - randomly pick a country for which the question
    // will be formed, also pick two countries for the wrong answers
    Random random = new Random();
    int correctAnswer = random.Next(0, countryCount - 1);
    int wrongAnswer = random.Next(0, countryCount - 1);

    // Step E - Based on the random numbers, get the CountryName and its correct capital
    // also load the names of the capitals of countries to be used as wrong answer.
    string questionCountryName = data.Tables[0].Rows[correctAnswer]["CountryName"].ToString();
    string questionCorrectCapital = data.Tables[0].Rows[correctAnswer]["Capital"].ToString();
    string questionWrongCapital = data.Tables[0].Rows[wrongAnswer]["Capital"].ToString();

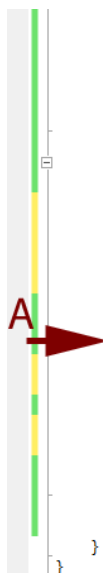
    // Step F - form the question using the country name
    lblQuestion.Text = "What is the capital of " + questionCountryName + "?";

    // Step G - we want to make one of the option boxes the correct answer
    // let's decide which box is the correct one based on the current milliseconds.
    // if current millisecond is an even number then the first option box is correct
    // the second option box is set as correct otherwise.
    if (DateTime.Now.Millisecond % 2 == 0)
    {
        optChoice1.Text = questionCorrectCapital; // first box has the right choice
        optChoice2.Text = questionWrongCapital;
        CorrectOptionBox = 1;
    }
    else
    {
        optChoice1.Text = questionWrongCapital;
        optChoice2.Text = questionCorrectCapital; // second box has the right choice
        CorrectOptionBox = 2;
    }
}

```

Step 7 – EvaluateAnswer() Function

The second part is to evaluate user's answer. We will write a new function named EvaluateAnswer(), right after AskQuestion() function.



```

{
    optChoice1.Text = questionWrongCapital;
    optChoice2.Text = questionCorrectCapital; // second box has the right choice
    CorrectOptionBox = 2;
}

void EvaluateAnswer()
{
    // if the correct optionbox is the first one and user checked the first optionbox,
    // or if the correct optionbox is the second one and user checked the second optionbox
    // then user answered correctly. we increment the score by 10 points.
    // otherwise, we decrement the score by 10
    if ((CorrectOptionBox == 1 && optChoice1.Checked) || (CorrectOptionBox == 2 && optChoice2.Checked))
        Score += 10;
    else
        Score -= 10;

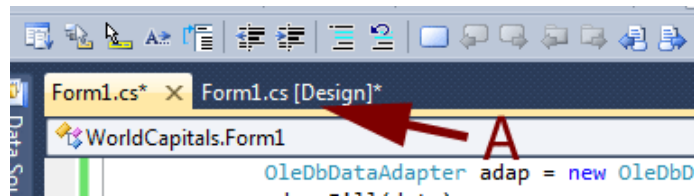
    // display the current score of the user
    lblScore.Text = "Your Score is " + Score.ToString();

    // and ask another question
    AskQuestion();
}
}

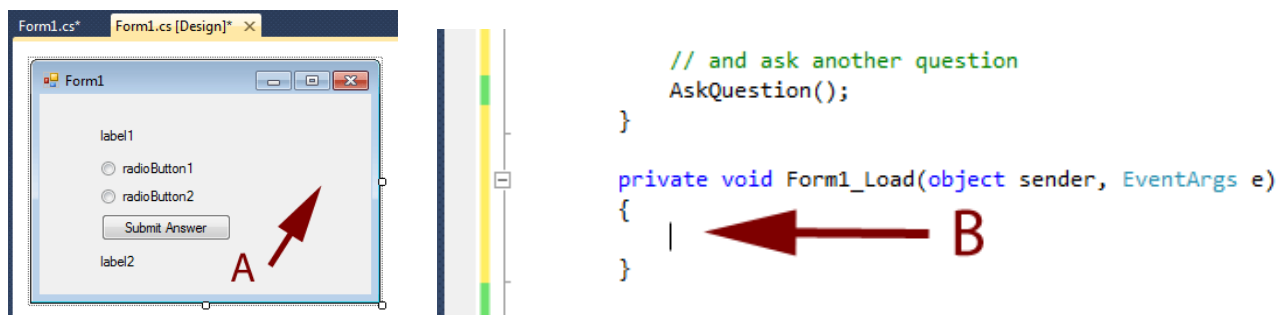
```

Step 8 – Setting up User Control Events

Visual Windows Forms based programs are event driven applications. That means user interacts with the textboxes, button, option boxes etc. And based on this interaction, our code responds. In our case, when the form loads – which means the first time application is run -- we would like to ask question to user. So we need to handle Load event of the Form. The easiest way to do this is to switch to design view [A].



Double click on the form [A] (but not on any of the controls inside the form). This will open up the code window and automatically add the function that will be executed upon form loading.

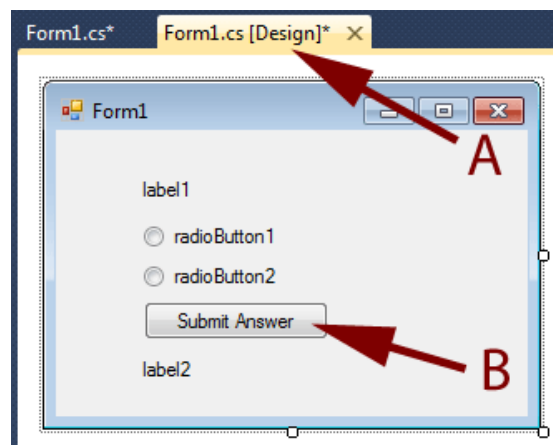


We want to ask question to user as soon as form loads, so call AskQuestion() function inside Form1_Load().

```
// and ask another question
AskQuestion();
}

private void Form1_Load(object sender, EventArgs e)
{
    AskQuestion(); ← A
}
```

Form Load event is fine. As soon as user clicks on Submit Answer button, we would like to evaluate user's answer. That means we need to handle click event of the button control. Go back to Design View [A] and double click on the button [B].



This will open up the code window and automatically add btnSubmitAnswer's Click event [A]. Call EvaluateAnswer() function to evaluate user's answer [B].

```
        // and ask another question
        AskQuestion();
    }

A private void Form1_Load(object sender, EventArgs e)
{
    AskQuestion();
}

private void btnSubmitAnswer_Click(object sender, EventArgs e)
{
    EvaluateAnswer(); B
}
```

You are ready to run your program. From the menu, Debug > Start Debugging. If you run into any problems, drop me an email.