Department of Computer Science

08120 Programming 2
Week 30 Practical 2013/2014
Dodgy Dice

This week we are going to experiment with objects. We care going to use an object which has been pre-written and see if there is anything wrong with it. In programming this is often called writing a "test harness" for the code. In the case of the object we have been given there are some *features* which have been placed there by a sneaky programmer, normally you would be looking for programming faults.

Honest Fred and his Dice object

We are going to be writing some programs which allow us to control moves in various board games. We have been given a class called **`DiceClass`** by "Honest Fred". We have not been given the source for the **`DiceClass`**, instead Fred has just given us the assembly file. For more details on this class, and the lab work, you can visit the course SharePoint site:

http://intra.net.dcs.hull.ac.uk/student/modules/08120/default.aspx

The Dice class

You will need to obtain the **`DiceLibrary.dll`** assembly for this practical. The assembly contains a namespace called **`GameClasses`**. Inside the namespace is a class called **`DiceClass`**. Your program must make an instance of this class and then use a method called **`Roll`** which can be called to make it print out a dice roll.

Below is a simple class called **`Chucker`** which creates an instance of a **`Dice`** object and then rolls it once:

```
using System;

using DiceLibrary;

class Chucker
{
    public static void Main ()
    {
        DiceClass dice = new DiceClass();
        Console.WriteLine ( dice.Roll());
    }
}
```

Before you go any further; perform the following:

1. Download the **`DiceLibrary.zip`** file from the laboratory web page on SharePoint. Copy it into the same folder as your program source file.
2. You now need to extract the library file (**`DiceLibrary.dll`**) from the archive and copy it into the same directory.
3. Create a copy of the program above in a source file called **`Chucker.cs`**.
4. Use the following command to compile the program. This makes use of the `DiceLibrary.dll` assembly.

   ```
   csc /reference:DiceLibrary.dll Chucker.cs
   ```

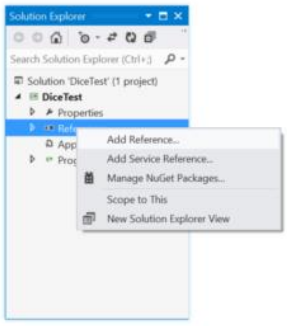
   The reference option tells the compiler to look in the specified assembly when building the program. If you are using Visual Studio to create the program you will have to add a reference to the **`DiceLibrary.dll`** file into the References for your project.
5. Compile and run the program.

Using the Assembly with Visual Studio

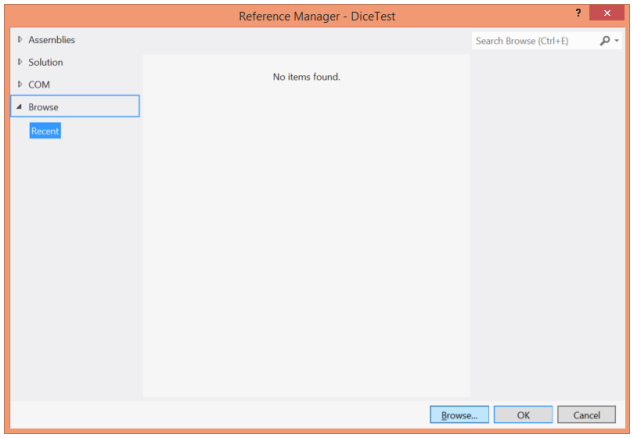
If you want to use the `DiceLibrary.dll` assembly with a Visual Studio project you need to include it as a reference in your project.

To use an assembly in a Visual Studio project do the following:

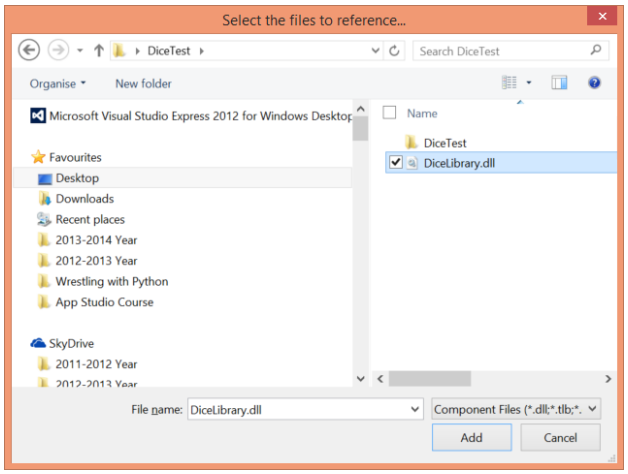
1. Create a new Visual Studio Console project (call it Chucker).
2. Download and extract the **`DiceLIbrary.dll`** file from the laboratory web page on SharePoint. Copy it into the same folder as your project.



3. Right click the References part of your solution and select the Add Reference option, as shown above. This will cause the Reference Manager to appear:

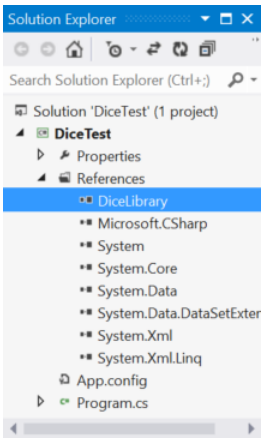


4. Select the Browse tab at the left of this dialog and press the Browse button. Find your DiceLibrary.dll library and select it.



5. Click Add and then click OK to close the Reference Manager window.

6. The Dice library should now be part of your project.



7. Add a line at the top of your program to use the DiceLibrary resources.

```
using DiceLibrary;
```

Dodgy Dice

If you run the **`Chucker`** program a few times you may decide that there is something very suspect about this dice. What you now need to do is get the value from the dice and do some testing of its performance. **`DiceClass`** also provides a method which returns the number of spots as an integer:

```
DiceClass dice = new DiceClass();
Console.WriteLine(dice.IntSpots());
```

The code above creates an instance of the DiceClass and uses the **`IntSpots`** method of the object to read the number of spots as an integer, and then print out this value.

8. Create a program that creates a dice and throws it.
9. Modify your program so that it prints out 20 spots values. You will need to use a **`for`** loop to call **`intSpots`** 20 times and print out its result. Run the program and comment on its performance.

Dice Testing

You are now going to do some statistics on the dice rolls. Over a large number of rolls an unbiased dice will give an average score of 3.5. If the dice class which Fred has supplied produces an average score which is different from this, then we have a dodgy dice.

10. Modify your program so that it prints out the average of 1000 spots values. You will need to use a **`for`** loop to call **`IntSpots`** 1000 times. Each time round the loop you add the latest number of spots to a total. At the end you divide the total number of spots by 1000 to get the average roll value.

Naming the Dice Player

Fred has provided another method which you can use to tell a **`Dice`** who is throwing it. The dice provides the method **`setPlayer`** which is supplied with the name of the person using it:

```
using System;

using DiceLibrary;

class ChuckValue
{
    public static void Main ()
    {
        dice.SetPlayer("Rob");
        Console.WriteLine ( dice.IntSpots());
    }
}
```

The program above sets the player to Rob and then throws the dice.

11. Modify your program so that it sets the name of the player to your name. Run the tests and see what happens.
12. Modify your program so that it sets the name of the player to Fred. Run the program and see what happens. Do you notice anything suspicious?
13. Write some test programs to see if you can explain what is happening.

14. Show a demonstrator the program working and explain what you think is happening.

Creating a Fair Dice

You might want to create a replacement for the dodgy dice that behaves properly no matter who is using it.

Creating Windowed Dice Application

You can use the dice library in a Windows application. Make one that displays a dice value each time the user presses a “throw” button.

Rob Miles
February 2014