THE UNIVERSITY OF NEWCASTLE AUSTRALIA

School of Information and Physical Sciences

Inft2012 Application Programming – Notes for week 5

Pair programming

Remember, programming in pairs – and taking turns at the keyboard – is known to be beneficial when learning to program and to enhance productivity in actual programming. You are strongly encouraged to work with a partner on the exercises each week.

Lab exercises

Remembering that learning to program takes lots of practice, you are strongly advised to finish these exercises before next week's lab class, bringing to the class any problems you encounter.

1. Reading and understanding program code is an essential step on the way to writing program code. Read the following code and explain what it does. Try to explain what the purpose or outcome of the code might be, rather than its individual steps. Do this just by reading the code – not by entering it into a program and trying it.

   If you can't tell just by reading the code, desk-check the program – use pen and paper to work out what it would do with some small value of *iLimit*.

   Students often skip the read-understand-explain question, either because it looks too hard or because the other questions look like more fun. There will be read-understand-explain questions on your exam, so you might as well get some practice in!

   As illustrated in this week's lecture and in Lec5Demo1LoopTests, the sequence "\r\n" in a string will cause the display to go to a new line when printing that string.

```
TbxDisplay.Clear();
for (i = 1; i < iLimit; i++)
{
    for (j = 1; j <= iLimit - i; j++)
    {
        TbxDisplay.AppendText(" ");
    }
    for (j = 1; j <= 2 * i; j++)
    {
        TbxDisplay.AppendText("*");
    }
    TbxDisplay.AppendText("\r\n");
}
```

2. Adjust your die-throwing program from weeks 2 and 3 so that when a button is clicked it throws the die 10 times. At each throw, the number thrown is to be displayed in one textbox and the progressive total of the throws is to be displayed in another.

   If you simply run this program, you'll see the end result but not the individual throws. How might you slow or stop the program between throws so that each result can be seen? There are several possibilities. You might use message boxes; you might use a timer. Here's another option that we haven't yet met. The following lines will pause the program, where iMillisecs is the delay time in milliseconds:

   ```
   Application.DoEvents();
   System.Threading.Thread.Sleep(iMillisecs);
   ```

3. Write a program that accepts two integer inputs from textboxes. In one label on your form, list all the numbers from 1 to the larger of the two inputs (which might be in either textbox). In another label, list all of the values from the first label that are evenly divisible by the smaller number. For example, with the inputs 2 and 6 . . .

One label contains the numbers from 1 to 6	The other label contains the ones that are divisible by 2
1	2
2	4
3	6
4	
5	
6	

4. Write a program that, when a button is pressed, checks the contents of a textbox. If the textbox does not contain a number, the program informs the user of this and does nothing more. Once a number has been entered, the program will do something trivial such as displaying double the number in a message box. For a general number, you should probably use the type double.

   Notice that this is not a looping exercise, even though the user might repeatedly enter non-numbers. Because it is an event-driven program, each 'iteration' is started when the user clicks the button. The repetition is driven by the user, not by a looping statement.