Chapter Six: Thinking Procedurally (Strings & User Functions)

String

This is a new data type. A string is a sequence of characters. Any variable we have used must be declared before we use it.

```
String    text = “this is a string”;
```

One of the most important things that we can do with strings is to stick them together (concatenation). We do this with the + operator. We have already seen this in the output statements.

Functions

A function is a self-contained program that can be called from another part of your program. It can be used over and over again. Functions can be given information to work with and they can give back answers. We want a program to print out the square according to the number that you enter.

```
*******
*******
*******
*******
*******
*******
*******

    public static void main(String[] args)
    {    int num   = IBIO.inputInt("enter number of lines ");
         String aa = stars(num);

         for (int i = 0; i < num; i++)
              IBIO.output(aa);
    }

    static String stars(int n)
    {    String xx = "";
         for (int i = 0; i < n; i++)
              xx = xx + "*";
         return xx;
    }
```

The important thing in this piece of code is the function stars():

static This would be explained in depth later, but for now you should always include it.
String This is the return value. That is the routine creates a String.
stars The name of the function.
() The argument list, the data that the function needs as input.
int The type of the data.
n The name of the data.

The function makes a String of stars by first making an empty string and then gradually adding on more and more stars until we have enough.

In the function ‘i’ is used and in the main program ‘i’ is also used. These two do not conflict because the ‘i’ in the function only has existence during the time the program is running. This is called the scope of the variable.

Pr 6.1 Change the program so that it prints the square 10 spaces out. Do this by changing only the function and not changing the main routine.

Problem Solving

Below is another shape that your program is to print. This time there are a different number of stars to print in each line. As before you enter the number of lines in the triangle.

```
*
**
***
****
*****
******
```

Do not erase your subroutine. You will use it again in this program. The first step is to write a program that will print the number of stars in each line.

```
int  n = input("how many lines ");
for (int i = 0; i < n; i++)
{    IBIO.output(i + 1);
}
```

This program will not print out the stars but will print out the number of stars needed for each line. Now replace the output in this program with the star routine as before to get the shape as above.

Pr 6.2 Write a program that will print out the following shape. The width depends on the number you first input.

```
*
**
***
****
*****
******
*****
****
***
**
*
```

Pr 6.3 Write a program that will print out the following shape. The width depends on the number you first input. This is when you input 3 and there are 3 stars in each line, separated by spaces and there are 3 lines, a middle line and then more lines. To do this you can create another routine that makes a blank String.

```
 * * *
 * * *
* * *
* * *
* * *
 * * *
 * * *
```

Chapter Seven: Thinking Ahead

You will notice that it is hard to print numbers nicely on the screen. Nothing seems to line up. In the first work sheet you printed out the numbers and squares and cubes of the number from 1 to 20. It would have been nice if they had lined up like

```
1       1       1
2       4       8
3       9      27
4      16      81
```

In the second work sheet you went to a lot of effort using the **if** statement to print out extra space so that the numbers would line up.

Examine the program below which prints numbers lined up on the right.

```
public static void main(String[] args)
{
    for (int i = 1; i < 20; i++)
    {   String s = pad(i, 10) + pad(i*i, 10) + pad(i*i*i, 10);
        IBIO.output(s);
    }
}

static String pad(int n, int tab)
{   String st = "" + n;                 // make a string of the number
    while (st.length() < tab)       // st.length()=how many characters are in st
    {   st = " " + st;                  // add spaces in front of the number
    }
    return st;
}
```

The method is to turn the number into a string and add spaces to the left side of the number.

Casting

To change a decimal into an integer we precede it with (int). This rounds down.

```
    int xx = 9.63 * 3.73;                 // error
    int xx = (int)9.63 * 3.73;        // error only 9.63 changed
    int xx = (int)(9.63 * 3.73);      // is correct = 35
    int xx = (int)9.63 * (int)3.73;   // also is correct = 27
```

Pr 7.1 Consider the program below. It computes powers of the number 3.732 and prints them out. Change the program so that all the answers are printed out to 2 decimal places only. Do this by multiplying by 100, change to integer and then dividing by 100;

```
    public static void main(String[] args)
    {   double  xx = 1;
        for (int i = 0; i < 10; i++)
        {   xx = xx * 3.732;
            IBIO.output(xx);
        }
    }
```

Pr 7.2 Change the program above so that decimal places line up. To do this you must change the decimal answer xx into a string (*String yy = "" + xx;*). Then use the command *yy.indexOf('.')* to find the position the decimal place is in the string. eg if *String yy = "47.29";* then *yy.indexOf('.')* will be 2. Remember that counting starts from 0. Then add enough spaces at the start to line up the number.