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.

Chapter Eight: Thinking Procedurally and Concurrently

User defined methods

A *method* is a subroutine that may or may not return a value. When they do return a value, they work like a *function*; when they do not, they work like simple *procedures*. In Java there is no distinction between them. Procedures are declared using `void` to indicate no return value.

```
public static void main(String args[])
{
   for (int i = 1; i < 100; i++)
   {  for (int j = 1; j < 100; j++)
      {  for (int k = 1; k < 100; k++)
         {  if ( good(i,j) && good(j,k) && good(i,k) )
               IBIO.output( I + "   " + j + "   " + k );
         }
      }
   }
}

static boolean good(int a, int b)
{  int     x = a * b + 1;
   int     y = (int) (Math.sqrt(x)+.5);
   return ( y * y == x );
}
```

Diophantine

Equations that have whole numbers as solutions are called diophantine. The above program attempts to find all numbers that are less than 100 and have the property that when they are multiplied together they are one less than a perfect square. The simplest example is 1, 3, 8 because $1*3 = 2^2-1$, $3*8 = 5^2-1$, $1*8 = 3^2-1$. The program above uses a subroutine to test each pair of numbers to see if they meet such condition.

Pr 8.1 Change the program so that no duplicates will be printed out.

Pr 8.2 Change the program so that it will find 4 numbers with the above property – that any two of them multiply together to make a number one less than a perfect square. (need to loop to 200 to find one answer)

Pr 8.3 Write a program that will find all the numbers less than 100 that have the property that $a^2 + b^2 = c^2$. (100 possible answers)

Pr 8.4 Write a function gcd which calculates the greatest common divisor of two numbers a and b. It does this by subtracting the smaller number from the larger and continues to do this until the numbers are the same. e.g. start {36, 27} then next stage {9, 27} then {9, 18} then {9, 9} now stops because both numbers are the same. 9 is the gcd of 36 and 27.

```
static int gcd(int a, int b)
{
   return x;
}
```

Pr 8.5 Change the program in 8.3 so that all the duplicates are removed (now 50 answers) and also remove all multiples. 3, 4, 5 is one of the answers and we do not want 6, 8, 10 to also be an answer (16 answers). Use the function `gcd()` created in Pr 8.4

Chapter Nine: Thinking Abstractly (Arrays)

Random Numbers

In Java random numbers are decimal numbers between 0 and 1. These are very useful for simulation experiments on a computer. If we want a random whole number like throwing a dice we multiply this number by 6 add 1 then convert to an integer. In the next example we use the computer to calculate 20 random numbers simulating the throw of a dice.

```
for (int i = 0; i < 100; i++)
{  double xx = Math.random() * 6;     //Math.random - decimal
   int    yy = (int)(xx + 1);         //change to number 1 to 6
   IBIO.output(yy);
}
```

Pr 9.1 Write a program that will generate 100 random numbers from 1 to 6 as in the above program and find the average of them.

Arrays

If we wanted to investigate the 100 numbers that we created above then we must have a way of saving them. The way of doing this is by using an array. This is a list of memories that use the same name. If we decide that the name of the array was "num" then the memories would be labelled "num[0], num[1], num[2], etc ". Notice that the first one is num[0] and not as expected num[1]. Like all variables an array must be declared before it is used. With an array the size that you want it to be must also be stated. This size is called its "dimension". The first line of the following program is the line that creates the array.

```
   int[ ] array = new int[size];
```

```
int[] num = new int[100];    // create the array

for (int i = 0; i < 100; i++)
{  double xx = Math.random() * 6;
   num[i] = (int)(xx + 1);
}

for (int i = 0; i < 100; i++)
   IBIO.output(num[i]);
```

This program will create 100 numbers in an array called "num" and then print them out.

Pr 9.2 Change the program above so that it uses a function called random(int) to make the random number. The finished program will look like the one below.

```
int[] num = new int[100]; //create the array

for (int i = 0; i < 100; i++)
   num[i] = random();     // your built in function

for (int i = 0; i < 100; i++)
   IBIO.output(num[i]);
```

Pr 9.3 Write a program that will generate 100 random numbers from 1 to 6 as in the above program. Then it will print them out. First all the 1's then all the 2's etc.

```
11111111111
22222
33333333
4444444444
555555
666666666666
```

To do this you will need to have one loop that counts from 1 to 6. Inside that loop another loop counts from 0 to 99, printing out the values. To print out a number without going to the next line use the command out() instead of output().

Pr 9.4 Write a program that will generate 100 pairs of random numbers from 1 to 6. It will save the sum of these numbers in an array. So at this stage every element of the array will have a number from 2 to 12. Then to print out a bar graph showing how many 2's, 3's there are. Use the method above to print the numbers from 2 to 12 in the margin lined up. This is like throwing two dice and adding the numbers together.

```
2  XXXXXXXX
3  XXXXXX
4  XXXXXXXXXX
5  XXXXXXXXXXX
6  XXXXXXXXX
```

Use the segment of the program below to put the numbers into the array.

```
   for (i = 0; i < 100; i++)
      num[i] = random(6) + random(6);
```

To do this the numbers on the left must be lined up so you use the Pad function that was used in Sheet 7.

Final Static

Often we need to define a constant that is the same throughout the program but we may want to change later. In the above example we use 100 and we may want later to change it to 1000 but do not want to go through the program and change it everywhere in the program.

```
public class Simple
{  public final static int SIZE = 100;

   public static void main(String[] args)
   {  int[] x = new int[SIZE];
      ...
```

The above segment shows how to add a constant called SIZE that is set to 100 at the beginning of the program and retains that value throughout. Constants are always capitalised, final, and static.

Pr 9.5 Change the last program so is uses an `int` constant called `SIZE`. Then run the program for 1000.

Chapter Ten: Thinking Ahead (Strings)

String & char

We came across strings in Chapter 7. Now we will learn how to input a string.

```
String xx = IBIO.input("enter your name ");
IBIO.output(xx);
```

This is a new type of variables that we can have. Strings are not one of the primitive data types, but an actual class, so it comes with several built-in methods.

```
public static void main(String args[])
{
   String ss = IBIO.input("enter your name ");   //input your name
   char[] xx = ss.toCharArray();                 //make into an array

   for (int i = 0; i < ss.length(); i++)
      IBIO.out(xx[i]);
}
```

This program will read in a sequence of letters and print them out again. The program will save the letters first in a String and then into an array.

Remember that the size of the array is exactly the right number for the number of characters that you typed in.

Pr 10.1 Write a program that will read in a sequence of letters and print out the word and then the word reversed and then the combination of both (note that the last letter is not repeated).

```
Input a word: TRAIN
TRAIN
NIART
TRAINIART
```

Pr 10.2 Write a program that will read in a sequence of 0's and 1's as a binary number and change the input form from binary to decimal. Remember that what is being read in are characters and not numbers. Your program must test if only 0's and 1's are in the input. Otherwise output error message. To test if the digit is a '0' or a '1' you must do the following

```
char x;
if (x == '0') // is x the digit 0
if (x == '1') // is x the digit 1
```

Refer back to your notes on binary to decimal conversion. Take the number so far, multiply it by 2, and then add next digit to get the next number.