جو لوگ ایمان لائے اور نیک عمل کرتے اور نماز پڑھتے اور زکوۃ دیتے رہے ان کو ان کے کاموں کا صلہ خدا کے ہاں ملے گا اور (قیامت کے دن) ان کو نہ کچھ خوف ہوا اور نہ وہ غمناک ہوں گے ۔ البقرۃ آیت ۲۷۷

Happiness depends upon ourselves. Aristotle

Lecture 13

Nested Loop and Intro to Methods/ Function Creation

In this lecture we will extend our discussion on nested loop and at the end we will discuss how to create user defined function/ methods.

Nested Loop

Once again we are starting discussion on nested loops. By definition nested loop means loop inside loop. There are certain problems for which creating logic is bit simpler using nested loop instead of single loop. In last lecture we have seen some examples here we will discuss more examples. We want to create a rectangular box of user defined size. Previously we have handled same problem with single loop using string of characters, spaces etc. Like:

```
-  input rows, columns
-  String stars="************************************"
-  String spaces="                                  "
-  print stars.substring(0, columns)
-  for a=2 to rows-1
-       print "*", spaces.substring(1, columns-1), "*"
-  next
-  print stars.substring(0, columns)
```

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

In this code we have used two strings name stars and spaces. We have print first line of stars outside loop using substring function on string stars and last line also outside loop in same way. Inside loop we are print *, followed by spaces again using substring function on string spaces, followed by * again. We used rows & columns variables to produce box of user defined size. Two rows are printed outside loop whereas rows-2 lines are printed using loop, that's why you can see loop is from 2 to rows-1. Students should implement this code in Java for practice. As I discussed before it is not always a choice to use Strings for such codes, however, nested loop is the choice always available. See code before discussion:

```
-  int rows, columns, i, j;
-  rows=in.nextInt();
-  columns=in.nextInt();
-  for (j=0;j<columns;j++)
-       ? ("*");
-  ?ln();// move to next line
-  for (i=1;i<rows;i++){
-       ? ("*");
-       for (j=2; ;)
-            ? (" ");
-       ?ln("*");//move to next line
-  }
-  for (j=0;j<columns;j++)
-       ? ("*");
```

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

Here is a generic code to draw box of any size, with any character. There are 4 loops in this program, two of them are independent and one is pair of nested loop. Two independent loops are drawing top most and lower most row; whereas; nested loop is drawing remaining rows. In pair of nested loop outer loop is printing starting, ending star and moving to next line; whereas inner loop printing spaces.

Here is another example of nested loop is to find similar letters from two strings. The idea is to pick characters one by one from string 1 and compare it with characters of string 2. If character of string 1 match by any character of string 2 increment count by 1. Here is the code:

```
String s1="Something", s2="Various";
char c1, c2;
int i,j,similarCount=0;
for (i=0;i<s1.length();i++){
  c1=s1.charAt(i);
  for (j=0;j<s2.length();j++){
    c2=s2.charAt(j);
    if (c1==c2)
      similarCount++;
  }
}
System.out.println("String has "+similarCount+" similar letters");
```

Output:

```
String has 2 similar letters
```

Student should implement code to find characters of string 1 not existing in string 2. Student should not modify above code by replacing statement `(c1==c2)` with `(c1!=c2)`, rather they have to use concept of flag. They should set flag if c1 equal to c2 and outside loop increment count if flag is not set by inner loop. Similarly previously we have created triangles of characters using substring but just consider if we want to create a pattern like:

```
1
12
123
1234
12345
123456
```

To create this code we can use an outer counting loop varying from 1 to 6. In inner loop terminating condition can be the variable of inner loop. See pseudo code:

```
-   for i=1 to 6
-      for j=1 to i
-         print j
-      next
-      move to next line
-   next
```

Here for loop is counting loop where ***i*** has an increment of 1 by default. Inner loop j is starting from 1 and ending on ***i*** where ***i*** is varying in outer loop, therefore indirectly inner loop is from j=1 to 1, j=1 to 2 and so on j=1 to 6. Therefore, output is single 1 in first line, 1, 2 in next line and so on. Student should implement this pseudo code in java. Last example is to create a pattern like:

```
    1
   212
  32123
 4321234
543212345
```

Once again I will discuss and give pseudo code and leave implementation for students. I may put them in next lab and homework as well. Anyhow see the pattern. We have 5 rows, where each row has 1 less space from next row. If we count spaces in first row, we can put a condition something count – row, so as row increase count-row decrease. Coming to numbers we can print them with two loops one is starting from row and decreasing towards 1 and other is starting from 2 and increasing towards row. Finally put them in an outer loop where row is starting from 1 and increasing towards 5:

```
- for rowNo=1 to 5
-     for spaces=1 to 5-rowNo
-         print "  "
-     next
-     for j=rowNo to 1
-         print j
-     next
-     for j=2 to rowNo
-         print j
-     next
-     move to next line
- next
```

Method/ Function Creation

Java like every programming language has enriched library full of defined functions but there is always need for user defined functions. User may develop their libraries by writing a java class having related functions. User may name them MyMath, MyString etc. for extension of existing libraries and Student, Account, Cards etc. for new libraries. User may develop their libraries for long term but for current problem they may write separate class for new functions or may write function in the same class where they are writing main function. Later we will learn how to decompose a problem into functions. Here we will just learn how to write a new function.

In order to create a new function we have to think what should be the **name** of the function. What should be the **parameters** of functions and finally **return value** of function. A function may have any name having first character non-digit, means first digit can be any alphabet or underscore but not a digit. By convention a function name should be verb representing the functionality of the function having first character small as well other characters like **find**, **search**, **sort**, **get**, **put**, **rewind** etc. In case a function name consists of two words second word should have first letter capital like **isEven**, **getValue**, **getIndex**, **nextValue** etc.

Now once again back to question why user needs to create functions. The answer is to remember software engineering principle **"DRY" Don’t Repeat Yourself**, means if you have to write same or similar code multiple times you can save repetition by creating a function and calling it as many times as required. For example you want to show a welcome message at start of every program. You may create a function ***sayWelcome*** in class **Messages**. Code is:

```
class Messages{
  public static void sayWelcome(){
    ?ln("-----------------------------------------------------------");
    ?ln("|  *               *    **********  *           *           |");
    ?ln("|  *               *    *           *           *           |");
    ?ln("|   *             *     *           *           *           |");
    ?ln("|   *             *     *           *           *           |");
    ?ln("|    *     *     *      ****        *           *           |");
    ?ln("|    *    * *    *      *           *           *          |");
    ?ln("|     *   * *   *       *           *           *          |");
    ?ln("|      * *   * *        *           *           *          |");
    ?ln("|       *     *         *********   ********    *********  |");
    ?ln("------------------------------------------------------------");
  }
  public static void main(String []args){
    sayWelcome();
  }
}
```

Here this code has main and sayWelcome functions in same program but one may call this function in any program by writing:

```
Messages.sayWelcome();
```

The only requirement is that both programs should exist in same folder. Similarly one may write a function to print some character any number of times. See code:

```
public static void print(char character, int times){
  int i;
  for (i=1;i<=times;i++)
    ? (character);
}
```

This code can be called anywhere to print any characters. Like we may write anywhere print ('*', 10) or print ('-', 20) in same class and in other class write class name followed by period followed by function call. Similarly we have written a code to check whether given number is prime number or not. Here we are creating a class MyMath and just check function; whereas; it may be called everywhere:

```
class MyMath{
  public static void main(String []args){
    int i,n;
    for (i=1;i<=10;i++){
      n=(int)(Math.random()*98+2);
      System.out.print(n+" is ");
      if (isPrime(n))
        System.out.println("prime");
      else
        System.out.println("not prime");
    }
  }
  public static boolean isPrime(int n){
    int i;
    boolean isPrime=true;
    for (i=2;i<=n/2 && isPrime;i++)
      if (n%i==0)
        isPrime=false;
    return isPrime;
  }
}
```

Test Output:

```
72 is not prime
86 is not prime
14 is not prime
96 is not prime
41 is prime
14 is not prime
18 is not prime
15 is not prime
61 is prime
24 is not prime
```

In main program we are generating 10 random numbers between 2 and 100 and using isPrime method we are saying whether generated number is prime number or not. One should note that we have not written code to check prime number in main rather we are calling it from main. Therefore this method can be called from any program by writing:

`MyMath.isPrime(n);` // where n is any integer

For further practice see lab 7 &homework 7 coming soon on website InshaAllah.