اور خدا تمہارا مددگار ہے تو تم پر کوئی غالب نہیں آسکتا۔ اور اگر وہ تمہیں چھوڑ دے تو پھر کون ہے کہ تمہاری مدد کرے

اور مومنوں کو چاہیئے کہ خدا ہی پر بھروسا رکھیں ۔آلعمران آیت 160

A friend to all is a friend to none. Aristotle

Lecture 10

Practice Counting Loop and Intro to Conditional Loop

We are passing through the most important phase of learning in programming. How rapidly variables change their values inside loop and effect of different statements within loop is not simple to understand. Also this is the phase where students can learn debugging as well. Therefore, if you are keeping on pending things as you use to do in your annual system, trust me, it's last time to leave this attitude in this course otherwise you will not be able to catch this course in this semester.

Practice Counting Loop

Write a program to get sum of first ten positive integers. Apparently it is very simple to write a program having following single statement:

```
System.out.println (1+2+3+4+5+6+7+8+9+10);
```

The problem is in this program is it is difficult to extend this program, if we want to get sum of first 200 positive integers or first 1000 positive integers and impossible to write a program which take input from user and get sum 1 to n; where n is input from user. So we will write all these now:

```
int count=1, sum=0;
while (count <= 10){
      sum = sum + count;
      count = count + 1;
}
? ("Sum of first 10 numbers is:"+sum);
```

In this program we are using 2 variables **count**, **sum**. First variable is used for controlling loop and second is to get sum. It is important point to note that whenever we have to do sum or count we have to initialize variable by zero. Inside loop we are adding value stored in variable **count** in previous value of **sum**, thus updating **sum** each time. At start **sum** is 0 and **count is 1**, inside loop count is added into sum, thus as **count** goes on increasing 1, 2, 3, ..., 10 as well **sum** goes on 1, 3, 6, 10, …, 55. Finally at the end output will be:

Sum of first 10 numbers is: 55

Now first of all it is very simple to update this program for any positive count by just replacing 10 by that other number like 500. Student must try modifying this program by changing 10 in condition and print statement. To get output like:

Sum of first 50 numbers is: 1275

Sum of first 300 numbers is: 45150

Sum of first 1500 numbers is: 1125750

Let us see another way to write same program by using another variable instead of writing constants:

```
int count=1, sum=0;
final int N=10;
while (count <= N){
      sum = sum + count;
      count = count + 1;
}
? ("Sum of first "+ N + " numbers is:"+sum);
```

In this program we have used N another variable and we have written a new keyword final with N to make it constant. By convention final/ constant variables are written in CAPS. A variable made constant can't be changed once a value is assigned. We have also change print statement having N instead of constants. The advantage of using this extra variable is that now we have to made change at only one place and remaining program will adjust automatically. Like if I assign 100 to N instead of 10, the program will give sum of first 100 positive numbers.

Lastly we can further generalize this program by taking **N** as input from user. In this case we have to add only three lines in the code:

```
Scanner in = new Scanner (System.in);
System.out.print ("Enter count to get sum: ");
N = in.nextInt();
```

Now our program can run for any count given by user and that's the point where loop is indispensable. We can't create such a program without loop. If you are not clear about last statement, don't advance until you clear this statement. Finally we are giving a complete program for those students who might join here, as I already discuss that this is the last point to indulge in this course, if you miss these lectures, sorry to say that you don't stand anywhere in this course:

```
import java.util.*;

class SumCount{
   public static void main(String[]args){
      Scanner in=new Scanner (System.in);
      int count=1, sum=0;
      final int N;
      System.out.print("Enter count to get sum:");
      N = in.nextInt();
      while (count<=N){
         sum=sum+count;
         count=count+1;
      }
      System.out.println("Sum of first "+ N + " positive numbers:" + sum);
   }
}
```

A sample run of above program is:

Enter count to get sum: 20

Sum of first 20 positive numbers: 210

Consider another example print n, square of n and cube of n, where n is any positive number, here I am writing a hard code but student must generalize this code by taking user input:

```
int n=1;
final int N=5;
while (n<=N){
   ? (n + "\t" + Math.pow(n,2) + "\t" + Math.pow(n,3));
} //Here replace ? with System.out.println
```

Output:

```
1     1     1
2     4     8
3     9     27
4     16    64
5     25    125
```

Now consider an example from String. Input user name and print all characters in separate lines. If name is "Safdar" output should be:

```
S
a
f
d
a
r
```

Once again hardcode program for the same can be written with the help of six statements like:

```
String n="Kareem";
? (n.charAt(0));
? (n.charAt(1));
...
? (n.charAt(5));
```

This program is fine for "Kareem" or any name of length 6, what if length varies? Either there will be a run-time error or incomplete output in result. Once again loop is indispensable to make this program generalize. If we critically examine the code we will see first print statement in **charAt** method argument is 0, whereas, in second statement argument is 1, and so on it goes to 5; whereas; length of string is 6. Therefore, we can write a loop with loop controlling variable moving from 0 to length of string and using same as argument of **charAt** function. The code is:

```
String n=in.next();//input from user
int i=0;
while (i<n.length()){
   ? (n.charAt(i));
   i=i+1;
}//once again replace ? by System.out.println
```

This code can work for any name or any other String. The program discussed in start related to sum of positive numbers, the same analogy can be used to draw pattern like:

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

The idea is instead of sum take a string variable **stars**, initialized **stars** by single star. Inside loop print **stars** and add one start to variables **stars** using concatenation. The code is:

```
String stars="*";
```

```
int i=1;
while (i<5){
   ? (stars);
   stars = stars + "*";
   i = i + 1;
}//once again replace ? by System.out.println
```

Conditional Loop

A conditional loop is not necessarily required all 3 elements required in counting loop, rather it is based on happening of some particular event. For such a loop only condition part is essential and programmer has to take care how to enter in the loop and how to exit. Like print two random numbers. Second number must be different from first. Therefore, loop can run for any number of times until a number different from first is generated. A similar example is that you want to generate a random number not divisible by 5. See example code:

```
int n=5;
while (n%5==0){//condition is to check if n is divisible by 5
      n = (int) (Math.random()*1000);
}
? (n);
```

This program will generate any random number between 0 and 1000 not divisible by 5 like 357, 672 etc. It is important to note that we require a number for which n%5 ≠ 0 but we have written n%5 == 0 because we want to remain inside loop till condition is true means number is divisible by 5. When a number will be generated not divisible by 5 loop will terminate and that is our requirement.

It is important to note that if we will not initialized n by a multiple of 5 loop will not execute because while loop is entry restricted. It is possible that loop will not execute even once; whereas; this program and many other programs require execution of loop at least once. For this another loop structure is do-while. In do-while entry is open and exit is restricted. That is every time control must enter inside loop and after each execution of loop condition is evaluated for exit. Therefore a while loop may execute 0 or more times; whereas; do-while loop may execute 1 or more times. To understand do-while syntax see the previous program with do-while:

```
int n;
do{
      n = (int) (Math.random()*1000);
} while (n%5==0); //condition is to check if n is divisible by 5
? (n);
```

For further practice see homework 5 coming soon on website InshaAllah.