Looping

Introduction

Looping (also called Repetition or Iteration) is when a program executes a block of code repeatedly, stopping only when a Test Condition changes to true (or to false, depending on the type of Loop).

Humans perform looping all the time. Suppose you prefer milk in your coffee. You might start by pouring a cup of black coffee, and then perform the following loop:

(Test Condition) 1. Is the coffee the right color? If yes, we're done. Stop adding milk. If not, go to Step 2.

(Loop Body)   2. Add a little milk, and then stir.

3. Start over, go back to Step 1.

Test Condition

A computer always evaluates the Test Condition as True or False.

The Test Condition can be a Boolean variable, which itself can only have a value of True or False.

Or, the Test Condition can be a simple expression, like i <= 5

```
while (i<=5)
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
```

Or the Test Condition can be **a compound expression**, like ((x > 10) && (y > 10))

```
// x and y must both be greater than 10 for this compound expression to be True

while ((x > 10) && (y > 10))

{
   document.write("x and y are each greater than 10");

    x = x +1;

    y= y +1;
}
```

Just remember, the Test Condition is always either True or False.

Java Looping Examples

Java has 3 looping structures.

- while Loop
- do...while Loop
- for Loop

while Loop

The syntax of a while loop is:

```
while(Boolean_expression)
{
   //Statements in Loop Body
}
```

- If *boolean_expression* result is TRUE, the actions inside the loop will be executed. This will continue as long as the expression result is true.
- Note: the *while* loop might not ever run. When the expression is first tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

Example:

```
public class Test {
   public static void main(String args[]){
      int x= 10;

      while( x < 20 ){
         System.out.print("value of x : " + x );
         x++;
         System.out.print("\n");
      }
   }
}
```

A lot of times, the programmer will have to “prime the loop”. If you want to remember it, think about trying to get water out of an aquarium with a syphon. It will work but you have to attract the water down. This is called priming a pump.

You need to set the control variable before testing it.

For example, if you want to echo people’s name until someone enter “q”.

```
String name;
Output (“enter the name”);
```

```
Input name;
While (name <>”Q”)
      Output (“welcome to the class, “ + name);
      Output (“enter the name”);
      Input name;
endWhile.
```

The part bolded and blue is called “priming the loop”.

do…while Loop

The syntax of a do...while loop is:

```
do
{
   //Statements
}while(Boolean_expression);
```

Notice that the Boolean expression appears at the end of the loop, so the statements in the loop execute at least once before the Boolean is tested.

If the Boolean expression is true, the flow of control jumps back up to do, and the statements in the Loop Body execute again. This process repeats until the Boolean expression is false.

Example:

```
public class Test {
   public static void main(String args[]){
      int x= 10;

      do{
         System.out.print("value of x : " + x );
         x++;
         System.out.print("\n");
      }while( x < 20 );
   }
}
```

Do note that do.. while execute the loop body at least once before testing. Most of the time this is the wrong structure for the problem. Apprentice programmers think that they can avoid priming a pretest (while..) loop by using a posttest (do.. while) loop but they can’t.

Let’s review a pretest loop which needs to be primed. For example: add all the numbers entered until the user enters a 10.

This should translate as:

```
Num my_num;
Num total = 0;
output (“enter a number”)
input my_num;
while (my_num <> 10)
      total = total + my_num;
      output (“enter a number”)
      input my_num;

Endwhile
```

Note that the part bolded and blue is an exact repetition.

Now if we try to do that with a post test loop, we get:

```
Num my_num
total = total + my_num;
do
     output ("enter a number")
     input my_num;
     if (my_num <> 10) total = total + my_num
while (my_num <> 10)
```

By trying to remove the loop priming, we end up testing for the condition twice.

for Loop

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

A for loop is useful when you know how many times a task is to be repeated.

The syntax of a for loop is:

```
for(initialization; Boolean_expression; update)
{
   //Statements
}
```

Here is the flow of control in a for loop:

1. The initialization step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
2. Next, the Boolean expression is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement past the for loop.
3. After the body of the for loop executes, the flow of control jumps back up to the update statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the Boolean expression.
4. The Boolean expression is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then update step,then Boolean expression). After the Boolean expression is false, the for loop terminates.

Example:

```
public class Test {
   public static void main(String args[]){

      for(int x = 10; x < 20; x = x+1){
         System.out.print("value of x : " + x );
         System.out.print("\n");
      }
   }
}
```

Looping Tips

1. Avoid Infinite Loops

If the Test Condition in a while loop always evaluates to True, then the statements in the Loop Body will execute over and over without stop. It is the programmer's responsibility to make sure that a statement in the Loop Body changes the value(s) of the variables tested in the Test Condition.

In the example below, the value of count never changes:

```
class WhileDemo
{
     public static void main(String[] args)
        {
          int count = 1;
          while (count < 11)
               {
                      System.out.println("Count is: " + count);

               }
        }
}
```

After the println statement, the programmer should add count++; to increment (add one to) the value of the count variable each time the Loop Body is executed.

2. Difference Between Pre-Test and Post-Test Loops

If the Test Condition appears above the Loop Body, it is a Pre-Test Loop. If the Test Condition appears below Loop Body, at the end, it is a Post-Test Loop.

In Java, the while Loop is a Pre-Test Loop, and the do…while Loop is a Post-Test Loop. The only difference between them is this: if the Test Condition is False the first time through the loop, the Loop Body in the while Loop will never be executed but the Loop Body in the do…while Loop will be executed one time before the Test Condition is tested.