for loop (Outsource: 5-15 – 5-17)

A loop that executes a specific number of times is a **count controlled loop**. You begin a for loop with the keyword for followed by a set of parentheses. Within the parentheses there are three sections separated by exactly two semicolons. The three sections are usually used for the following:

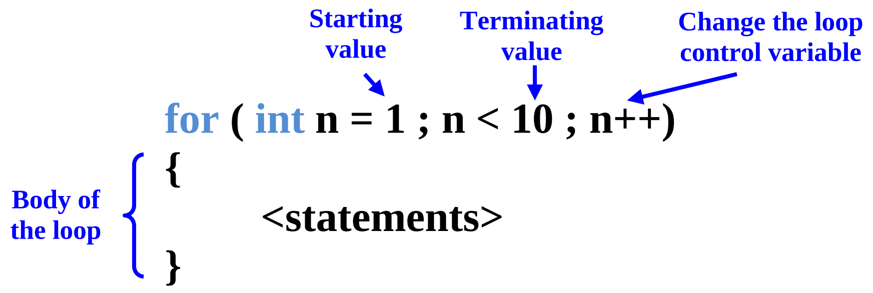
- Initializing the loop control variable
- Testing the loop control variable
- Updating the loop control variable

The body of the for statement follows the parentheses. As with an if statement, you can use a single statement as the body of a for loop, or you can use a block of statements enclosed in curly brackets.

Although the three sections of the for loop are most commonly used for initializing, testing, and incrementing, you can also perform the following tasks:

- You can initialize more than one variable by placing commas between the separate statements as in **for (g = 0, h = 1; g < 6; g++)**.
- You can perform more than one test, as in **for (g = 0; g < 3 && h > 1; g++)**.
- You can decrement or perform some other task, as in **for (g = 5; g >= 1; g--)**.
- You can even leave one or more portions of the for loop empty, although the two semicolons are still required as placeholders.

Usually you should use a for loop for its intended purpose, which is a shorthand way of programming a count controlled loop.



```
for ( int n = 1 ; n < 10 ; n++)
{
        <statements>
}
```

NOTE: Be careful about mistakenly placing a semicolon after the ending parenthesis of a for loop.

Accumulators and Counters

As you have seen in previous assignments, many loops are controlled by continuously adding a constant value to some variable, or **counting**, as in **count = count + 1** or **count++** (counting by 1) or **count = count + 2** (counting by 2). Another very useful technique in programming is **accumulating** or adding a new value to an existing value. For example, **total = total + newAmount**.

The following example demonstrates the use of counters and accumulators by counting all the number that are divisible by 7 from the start value (15) to the end value (77). The program then calculates the average of the numbers by dividing the accumulator by the counter.

```
import java.text.DecimalFormat;

public class Example
{
   private int start = 15;
   private int end = 77;
   private int count = 0;          // A counter variable
   private int sum = 0;            // An accumulator variable
   private double average = 0;

   public static void main(String args[]) {
      Example ex = new Example();
      ex.process();
      ex.output();
   }

   public void process() {
      for (int i = start; i <= end; i++)
      {
         if (i % 7 == 0)
         {
            count++;               //or count += 1;  or count = count + 1;
            sum = sum + i;
         }
      }
      average = (double) sum / count;
   }

   public void output() {
      DecimalFormat df = new DecimalFormat("#.#");

      System.out.println("There are " + count +
            " multiples of 7 between " + start + " and " + end + ".");
      System.out.println ("The sum of the multiples of 7 is " +
            sum + ".");
      System.out.println ("The average of the multiples of 7 is " +
            df.format(average) + ".");
   }
}
```

Lab 09 - ASSIGNMENT

Lab 09A - 60 points

OBJECTIVE

WAP to input an integer from the keyboard and displays a row of goblins equal to the input number. Use only one output statement that prints only one goblin.

HINT: The loop required for this program is a simple counting loop. It should count from 0 to the number entered from the keyboard. *The loop has nothing to do with the goblin other than determining how many goblins are printed.* The print statement inside the loop should print *one* goblin character.

FIELD SUMMARY

- **int goblins** – the number of goblins to be displayed.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `input` and `output`.
- **input** – declare a `Scanner` object and read an integer from the keyboard using an appropriate prompt.
- **output** – display the specified number of goblins using a `for` loop. Goblins can be defined as char 1, I.E. **`char goblin = 1;`**

SAMPLE KEYBOARD INPUT

Enter the number of Goblins to display: **6**

SAMPLE OUTPUT

There be goblins here: ☻ ☻ ☻ ☻ ☻ ☻

Lab 09B - 70 points

OBJECTIVE

WAP to output the sum and the average of all the even numbers between two values read from the keyboard, excluding multiples of 5. Include appropriate prompts for the two input integers. Your program should be able to input the two integers in any order, i.e., smaller first or larger first.

HINT: Declare a variable to use as the terminating value and assign it the *largest* value (**`Math.max`**). Assign the *smallest* of the two numbers (**`Math.min`**) to the loop control variable. Use an **`if`** statement to sum up and accumulate the *even* numbers that are *not* divisible by 5 (*two conditions*).

FIELD SUMMARY

- **int a** – the first number read from the keyboard.
- **int b** – the second number read from the keyboard.
- **int sum** – the sum of all the even numbers excluding multiples of 5.
- **int count** – the number of even numbers.
- **double average** – the average of all the even numbers excluding multiples of 5.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `input`, and `output`.
- **input** – declare a `Scanner` object and read two integers from the keyboard.
- **process** – count and accumulate the value of all the even numbers (excluding multiples of 5) from the smallest value to the largest value of the two numbers entered from the keyboard. Then calculate the average of the numbers.
- **output** – display the sum and the average in sentence format as shown below.

SAMPLE KEYBOARD INPUT

Enter the first number: 30
Enter the last number: 5

SAMPLE OUTPUT

The sum of the even numbers between 5 and 30 (excluding multiples of 5) is 174.
The average of the even numbers (excluding multiples of 5) is 17.4.

Lab 09C - 80 points

OBJECTIVE

WAP to read a sentence from the keyboard. Output the sentence in reverse.

HINT: The characters contained within a String are indexed from 0 to the Strings **length() – 1**. In the example shown below the length of the string is 21 which makes the index of the last character **length() – 1**. The loop for this program should count backwards from the last index position of the String down to (and including) 0. The print statement inside the loop should print the Strings character at the specified index position using the Strings **`charAt`** method, i.e. **sentence.charAt(<index Position>)**.

	A	**l**	**l**		**D**	**o**	**g**	**s**		**G**	**o**		**T**	**o**		**H**	**e**	**a**	**v**	**e**	**n**
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

sentence.length()
(21)

sentence.length() - 1

FIELD SUMMARY

- **String sentence** – a sentence read from the keyboard.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `input` and `output`.
- **input** – declare a scanner object and read a sentence from the keyboard
- **output** – display the input sentence backwards using a `for` loop (that counts backwards starting at the string's `length()` - 1). Use the String's `charAt` method to output individual characters in the string.

SAMPLE KEYBOARD INPUT

Enter a sentence: **All Dogs Go To Heaven**

SAMPLE OUTPUT

nevaeH oT oG sgoD llA

Lab 09D - 90 points

OBJECTIVE

WAP to output all the characters between two input values from the keyboard. Include appropriate prompts for the two input characters. Only capital letters will be used. Display all the letters from the first character to the second character (inclusive) using a for loop. Then display the number of characters that were involved. Note: If the first character is smaller than the second character the loop runs forward, otherwise it runs backward!

HINT: For loops can count using numeric values (i.e. from 1 to 10) and they can also count using characters (i.e. from 'A' to 'Z') since character data types are actually numbers (ASCII values). For example:

```
for (char c = 'A'; c < 'Z'; c++)    // Count forward by incrementing
    System.out.print(c + "  ");
```

would print all the characters from 'A' to 'Z' and

```
for (char c = 'Z'; c >= 'A'; c--)   // Count backwards by decrementing
    System.out.print(c + "  ");
```

would print all the characters from 'Z' to 'A'.

NOTE: Remember, the Scanner class does not have a method for reading characters. You must read in a string and then extract the first character, i.e. `reader.next().charAt(0)`.

SAMPLE KEYBOARD INPUT

Enter the first character: **C**
Enter the second character: **M**

SAMPLE OUTPUT

The Alphabet from C to M: C D E F G H I J K L M
There are 11 characters from C to M.

SAMPLE KEYBOARD INPUT

Enter the first character: **X**
Enter the second character: **L**

SAMPLE OUTPUT

The Alphabet from X to L: X W V U T S R Q P O N M L
There are 13 characters from X to L.

Lab 09E - 100 points

OBJECTIVE

WAP that displays a specified number of verses from the traveling song "Ninety Nine Bottles of Beer". Read the number of verses to display from the keyboard.

HINT: The loop required for this program is a simple counting loop. It should count from 0 to the number entered from the keyboard. The number of bottles of beer on the wall begins at 99. Subtracting the value of the loop control variable from 99 will result in 99, 98, 97, etc. The will be only four print statements in the body of the loop.

SAMPLE KEYBOARD INPUT

Enter the number of verses to display: **2**

SAMPLE OUTPUT

```
99 bottles of beer on the wall
99 bottles of beer
If one of those bottles should happen to fall
98 bottles of beer on the wall

98 bottles of beer on the wall
98 bottles of beer
If one of those bottles should happen to fall
97 bottles of beer on the wall
```