Lesson:

Loops

Pre-Requisites

- Basic syntax of Java
- Variables
- Operators
- Conditionals

List of Concepts Involved

- Introduction to Iterative statements/Loops
- while loop
- for loop
- How to choose between while loop and for loop?
- do-while loop
- Break keyword
- Continue keyword
- Using labels with continue and break keyword

Topic: Introduction to Iterative statements/Loops

Are you familiar with the word loop? The simple dictionary meaning of loop is a structure, series, or process, the end of which is connected to the beginning. The point of focus here is that the beginning and end are connected.

Pictorially, you can visualize a loop like this

Here, once the loop begins, end will drive start and start will drive end until certain condition to break out of loop is encountered.

Now that we have understood the simple meaning of loop, let us now move to the computer programming terms.
In computer programming, a loop is a sequence of instructions continually repeated until a certain condition is reached.

Assume this scenario, someone comes up to you and says “I want you to give me a program that can give me all the numbers between 1 and 10000”. In such a situation, writing all the numbers from 1 to 10000 isn’t a feasible solution. Can you connect the dots now?
Yes, that’s where the concept of loops comes in. Loops help you perform a task repeatedly until a certain condition is met. In our example, the task would be to print the value of the number, and the till it is less than 10000.

We have the following types of iterative statements -

1. The while loop
2. The for loop
3. The do-while loop

Topic: The while loop

A while loop is a loop that runs through its body, known as a while statement, as long as a predetermined condition is evaluated as true.

Syntax

```
while (condition)
 statement;
```

Let us understand the working of while loop with the help of this simple example -

Q- Print the first 10 natural numbers.

To accomplish this-

1. We declare a variable ‘i’ which denotes the current number. We initialize it with the value 1 (as you already know, the first natural number).
2. In the while loop, we put a condition that makes the loop run till the value of the variable i doesn’t exceed 10.
3. Finally, we print the value of the variable ‘i’ and then increment it by 1.

Code

```
int i = 1;
while (i ≤ 10) {
    System.out.print(i + “ “);
     i = i + 1;
}
```

Output - `1 2 3 4 5 6 7 8 9 10`

Try these :

1. Print the sum of the first ‘n’ natural numbers using a for loop, where n is the input.
2. Write a short program that prints each number from 1 to 100 on a new line.
   For each multiple of 3, print "Fizz" instead of the number.
   For each multiple of 5, print "Buzz" instead of the number.
   For numbers which are multiples of both 3 and 5, print "FizzBuzz" instead of the number.

Topic: For loop

This is the most widely used loop amongst all.
The advantage of a for loop is we know exactly how many times the loop will execute even before the actual loop starts executing.
Unlike *while loop*, in for loop we have 3 parts in the for header.

Syntax:

```
for (init-statement; condition; final-expression) {
     statement
}
```

Init-statement: This statement is used to initialize or assign a starting value to a variable which may be altered over the course of the loop (we will see this while solving examples). It is used/referred only once at the start of the loop.

Condition: This condition serves as a loop control statement. The loop block is executed until the condition evaluates to true.

Final expression: It is evaluated after each iteration of the loop. It is generally to update the values of the loop variables.

Now that we know the use of each of these, let's understand it with an actual loop.

Execution flow of for loop:

```
for (int index = 0; index =/= 5; index++) {
      System.out.print(index + “ “);
}
```

Output - `0 1 2 3 4`
Here,

1. Init-statement is executed only once at the start of the loop. In this example, index variable is defined and initialized to 0.
2. Next, the condition is evaluated. If index is not equal to 5, the for body is executed. Otherwise, the loop terminates. If the condition is false in the first iteration itself, then the for body is not executed at all.
3. If the condition is true, the for body executes. In this case, the for body prints the value of index (the print statement).
4. Finally, the final-expression is evaluated. In this example, index is incremented by 1.

These four steps represent the first iteration of the for loop. Step 1 is executed only once on entry to the loop. Steps 2, 3, and 4 are repeated until the condition evaluates as false i.e. index becomes equal to 5.

Acceptable alternatives in for loop:

Omitting parts of for loop

In a 'for' loop, we can omit any (or all) of init-statement, condition and final-expression.

1. We can omit the init-statement when an initialization is unnecessary. This may be the case when the variable may have already been declared.
   Example-
   ```
   int index = 0;
   for(; index < 5; index++)
         System.out.println(index);
   ```
   Note that the semicolon is necessary to indicate the absence of init-statement—more precisely, the semicolon represents a null init-statement.

2. Omitting the condition is equivalent to setting it as true. Because of this, the for loop must have an exit statement inside the loop. Otherwise, it may lead to a never-ending or infinite loop which is a nightmare for any processing system.
   Example-
   ```
   for(int index = 0; ; index++)
   {
      statement + code inside the loop must stop the iteration!
   }
   ```

3. We can also omit final expression. In such loops, either the condition or the body must do something to advance the iteration, otherwise the loop will not run any further owing to lack of information.
   Example-
   ```
   for(int index = 0; index =≠ 5; ) {
      System.out.println(index);
      index = index + 1;
   }
   ```

Note -

1. The above statements can all be omitted together too.
2. We can also have multiple statements inside the loop.
   Let us look at this example to know what we are talking about

   ```
   for(int i = 0, j = 14; i < 10 && j > 5; i++, j--);
   ```
   Here, you may see that we are handling two variables, conditions wrt both and final expression, all in one for loop.

   This is why it is the most popular loop of all.

Let us take the same previous example -

Print the first 10 natural numbers.

To accomplish this with a for loop-

1. We declare and define the int variable ‘i’ (int i=1), but this time we do it as a part of for loop.
2. We put the condition as i to be less than or equal to 10.
3. Inside the for loop body, we print the value of ‘i’.
4. Finally in the final-expression, we increment the value of the variable ‘i’.

Code

```
for (int i = 1; i ⩽ 10; i++) {
 System.out.println(i + “ “);
}
```

Output - `1 2 3 4 5 6 7 8 9 10`

Try these using for loop

1. Print the sum of the first 10 natural numbers.

2. Write a short program that prints each number from 1 to 100 on a new line.
For each multiple of 3, print "Fizz" instead of the number.
For each multiple of 5, print "Buzz" instead of the number.
For numbers which are multiples of both 3 and 5, print "FizzBuzz" instead of the number.

Topic: How to choose between while loop and for loop?

We have a fair knowledge of both the loops viz while and for. The common confusion at this point is, how do we decide which one to use ? Are both exactly the same or do they have any difference?

The simple answer to this is your own judgement and choice. Each person has different preferences. However, generally a while loop is used whenever the total number of iterations to be made is unknown. For example,

1. Use a for loop when you are traversing a data structure like an array(we will learn about arrays in the forth-coming class).
2. Use a for loop when you know that loop needs to run ‘n’ number of times.

Whereas,

1. Use a while loop when the termination condition is taking too much time to be checked or the termination condition is not known beforehand.
2. Use a while loop when increment condition is non standard like i=i*2.
3. Use a while loop when you are unsure till when the loop will continue, like while finding the first number divisible by both 5 and 7.

Let us move to the next looping statement which is seldom used and is a variation of while loop.

Topic: The do-while loop

Unlike while and for loop, do-while loop tests the condition at the end of each execution for the next iteration. In other words, the loop is executed at least once before the condition is checked. Rest everything is the same as while loop.

Syntax

```
do
{
 statement;
} while (condition);
```

Let us understand it with the help of the following example :

```
int idx = 15;
do
{
 System.out.print(idx + " ");
} while (idx < 5);
```

Output- 15

Explanation- In the above example, when we enter the loop, there will be no condition check. Therefore, we get 15 printed because of the system.out.println() statement inside the loop. However, in the next iteration, the program goes through the condition check mentioned in the while part. This time it will fail and the loop execution will end.

Let us now look at this example, can you guess the output?

```
int idx = 15;
do {
System.out.print(idx + " ");
  idx = idx + 1;
} while (idx ⩽ 16);
```

Output- 15 16

Try this:
Print the sum of the first 10 natural numbers using do while loop.

Topic: Break keyword

The break command **allows you to terminate and exit a loop (do while / for / while) or switch commands from any point other than the logical end**. It can be used in cases where we want the immediate termination of a loop based upon certain conditions.

Let us look at these examples for clarity -

```
for(int i = 1; i <= 3; i++) {
      for(int j = 1; j <= 3; j++) {
        System.out.print(j + " ");
      }
}
```

Output-
```
1 2 3
1 2 3
1 2 3
```

Code (with break)

```
for(int i = 1; i ⩽ 3; i++) {
 for(int j = 1; j ⩽ 3; j++) {
  System.out.print(j + " ");
  if(i == j) break;
 }
}
```

Output-
```
1
1 2
1 2 3
```

Explanation- Did you notice the difference in output ?Without the break statement, the inner loop is executed 3 times for each iteration of i. However, after the break statement is added to the condition that 'i' equals 'j', the inner loop is terminated whenever this condition is true.

Try these :

1. Print the first multiple of 5 which is also a multiple of 7.
2. Tell if the number in the input is prime or not.

Topic: Continue keyword

The continue keyword is used to end the current iteration in a for loop (or a while loop), and continues to the next iteration.It can be used in cases where we want the remaining block of code to get executed in the loop for the specific iteration.

Let us look at the following example for clarity -

Code

```
for(int i = 1; i ⩽ 5; i++) {
 if(i == 3) continue;
 System.out.print(i + " ");
}
```

Output- `1 2 4 5`

Explanation- When the value of i becomes equal to 3, the continue statement makes the loop jump onto the next iteration, skipping the remainder of the code, which in this case is skipping the printing statement of printing 3.

Try these

1. Print all values between 1 and 100, except if it's a multiple of 3.
2. Print all factors of the number in the input.

Topic: Using labels with continue and break keywords

We can also label a specific loop, just as we name a variable, and then use the continue or break statement to apply continue/break on the specific loop.

Let us look at the example below for better understanding

```
first:
 for(int i = 0; i < 3; i++) {
  for(int j = 0; j < 3; j++) {
   if(i == 1 && j == 1)
    continue first;
   System.out. println(i + " " + j);
  }
 }
```

Output

```
0 0
0 1
0 2
1 0
2 0
2 1
2 2
```

Explanation

Here, as soon as we reach the continue statement, unlike normal scenarios, the control moves to the next iteration of the outer loop that is labeled as “first”.

```
second:
 for(int i = 0; i < 3; i++) {
  for(int j = 0; j < 3; j++) {
   if(i == 1 && j == 1)
    break second;
   System.out. println(i + “ “ + j);
  }
 }
```

Output

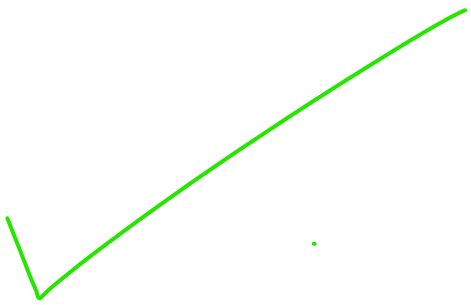
```
0 0
0 1
0 2
1 0
```

Explanation

Here, as soon as we reach the break statement, unlike normal scenarios, the control breaks out of the outer loop that we labeled as “second”.

Upcoming Class Teasers

- Problem on loops



18/08/2023