Lesson Plan

Loops

PW SKILLS

List of Concepts Involved:

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

Topic 1: Introduction to Iterative statements/Loops

Assume 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. That’s where loops come in. They 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 condition would be till the time it is less than 10000. In Java, we have 3 types of iterative statements -

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

Topic 2: 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;
```

Example -

Print the first 10 natural numbers.

To do this-

1. We declare a variable ‘i’ which denotes the current number. We initialize it with the value 1 (the first natural number).
2. In the while loop, we put the condition that runs the loop till the value of the variable i doesn’t exceed 10.
3. Finally in the statement, first 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 this

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 3: The for loop

Unlike while loop, in for loop we have 3 parts in the for header.

Syntax

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

Init-statement: In general, this statement is used to initialize or assign a starting value to a variable which may be altered over the course of the loop. It is executed only once at the start.

Condition: Similar to the while condition, it serves as a loop control. 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.

Execution flow of for loop:

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

Output - 0 1 2 3 4

1. Init-statement is executed once at the start of the loop. In this example, index is defined and initialized to zero.
2. Next, the condition is evaluated. If the index is not equal to 5, the for body is executed. Otherwise, the loop terminates. If the condition is false on the first iteration, 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.
4. Finally, the final-expression is evaluated. In this example, the 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.

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 writing 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 loop.

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.

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 have multiple statements inside the loop.

Example-

```
for(int i = 0, j = 14; i < 10 && j > 5; i++, j-);
```

Example-

Print the first 10 natural numbers.

To do this through a for loop-

1. We declare the int variable 'i' the same as before, but this time we do it as a part of for loop.
2. Again we put the condition for i to be less than or equal to 10.
3. In the for statement 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 this

1. Print the sum of the first 10 natural numbers using a for loop.
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 4: How to choose between while loop and for loop?

Deciding which loop to use is a judgemental call. 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.
2. Use a for loop when you know that loop needs to run 'n' number of times.

Whereas,

1. Use a while loop when increment type is nonstandard like i = i * 2.
2. 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.

Topic 5: The do-while loop

Unlike while and for loop, do-while loop tests for 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. Other than that everything is the same as in the while loop.

Syntax

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

Example -

Code

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

Output- 15

Explanation- In the above example, when we first enter the loop, there will be no condition check. Consequently, we get 15 as an output. But for entering the loop the next time, we will go through the condition check. This time it will fail and the loop execution will end.

Code

```
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 6: Break keyword

It is a special keyword used to terminate the execution of the nearest executing loop. It can be used in cases where we want the immediate termination of a loop based upon certain conditions.

Example -

Code (without break)

```
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 (without 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- Without the break statement, the inner loop is executed 3 times for each iteration of i. However, after the break statement is added on the condition that 'i' equals 'j', the inner loop is terminated whenever the condition is fulfilled.

Try this

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 7: Continue keyword

It is a special keyword used to skip to the next iteration of the loop. It can be used in cases where we want the remaining block of code to get executed in the loop for the specific iteration.

Example -

Code (without break)

```
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 printing 'i'.

Try this

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 8: Using labels with continue and break keywords

In this we label a specific loop, just like we name a variable, and then use the continue or break statement to apply continue/break on that specific loop.

```
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- As soon as we reach the continue statement, unlike normal scenarios, we move on to the next iteration of the outer loop that we 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- As soon as we reach the break statement, unlike normal scenarios, we break out of the outer loop that we labeled as "first".