UNIT VIII : ITERATIVE CONSTRUCTS IN JAVA

Loop- A construct in which a block of statements gets executed repeatedly unless the required task has been completed, such repetitive construct is known as an iterative construct or loop.

Iteration- Iteration means repeated execution of a set of statements.

Based on flow of control, there are two types of loop :-

- Entry Control Loop (For loop and while loop)
- Exit control loop (Do while loop)

Loop Type	Syntax	Description
for	**for (initialization; condition; update)** **{ /* Code to be executed */ }**	A loop that repeats a specific number of times, based on a counter or iterator. The **initialization** statement sets the initial value of the counter, the **condition** is checked before each iteration of the loop, and the **update** statement is executed at the end of each iteration to update the counter.
while	**while (condition)** **{ /* Code to be executed */ }**	A loop that repeats as long as a specific condition is true. The **condition** is checked before each iteration of the loop, and if it is true, the loop continues.
do-while	**do** **{ /* Code to be**	A loop that is guaranteed to execute at least once, because the **condition** is checked at the end of each iteration. If the **condition** is true, the loop continues.

Loop Type	Syntax	Description

	executed */ }
	while (condition);	

Here are some key differences between these loop types:

Loop Type	Key Differences
for	- The number of iterations is known at the start of the loop - The **initialization**, **condition**, and **update** statements are all included in the loop header
while	- The loop condition is checked at the start of each iteration - The loop body may never execute if the condition is false at the start
do-while	- The loop body is guaranteed to execute at least once - The loop condition is checked at the end of each iteration - The loop body will always execute at least once, even if the condition is false at the start

Break and Continue statements used in loop

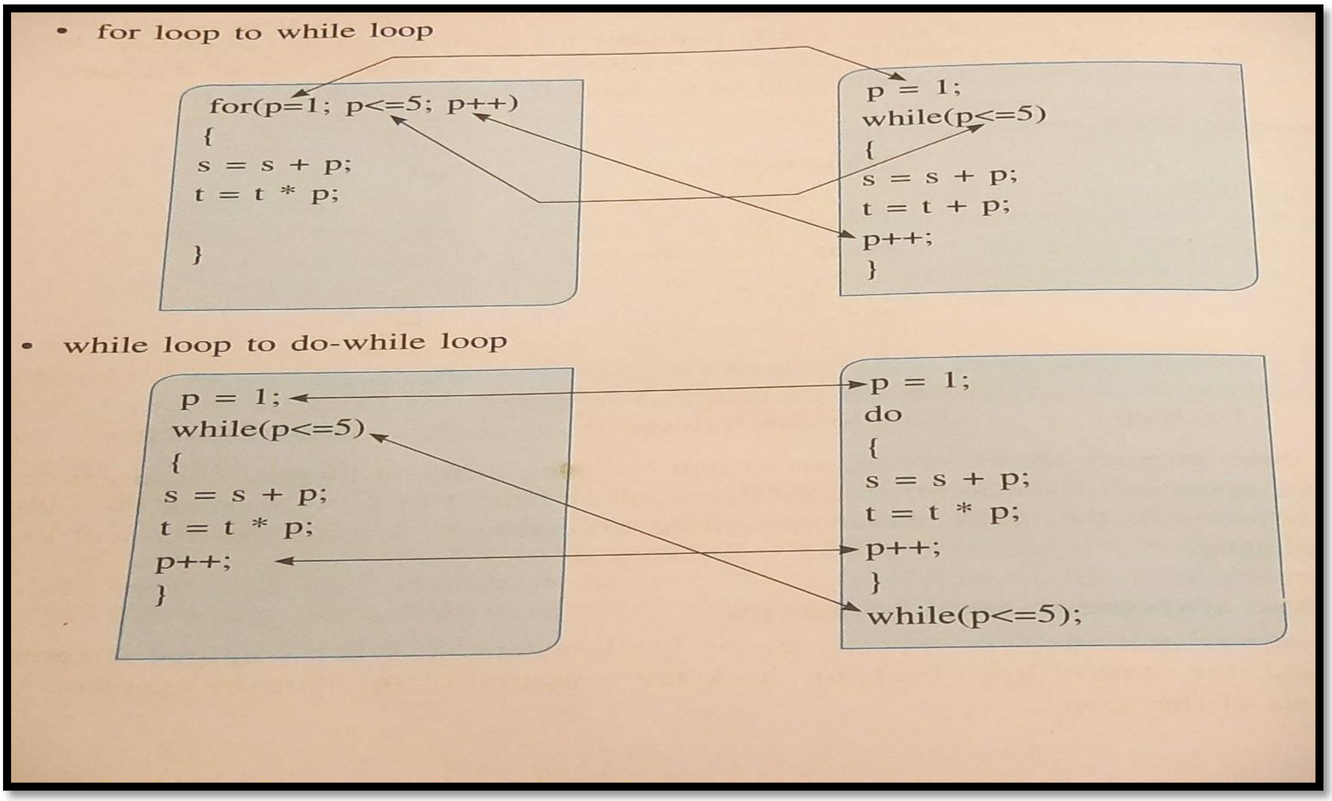
Name	Description	Use	Syntax	Example
break	Used to exit a loop prematurely	When a certain condition is met and the loop should no longer continue	break;	for (int i = 0; i < 10; i++) { if (i == 5)

Name	Description	Use	Syntax	Example

				{ break; // exit the loop when i equals 5 }
				System.out.println(i); }

continue	Used to skip the current iteration of a loop and move on to the next iteration	When a certain condition is met and the loop should skip over that particular iteration	continue;	for (int i = 0; i < 10; i++)
				{ if (i % 2 == 0)
				{ continue; // skip even numbers }
				System.out.println(i); }

Few examples of interconversion of Loops:



- for loop to while loop

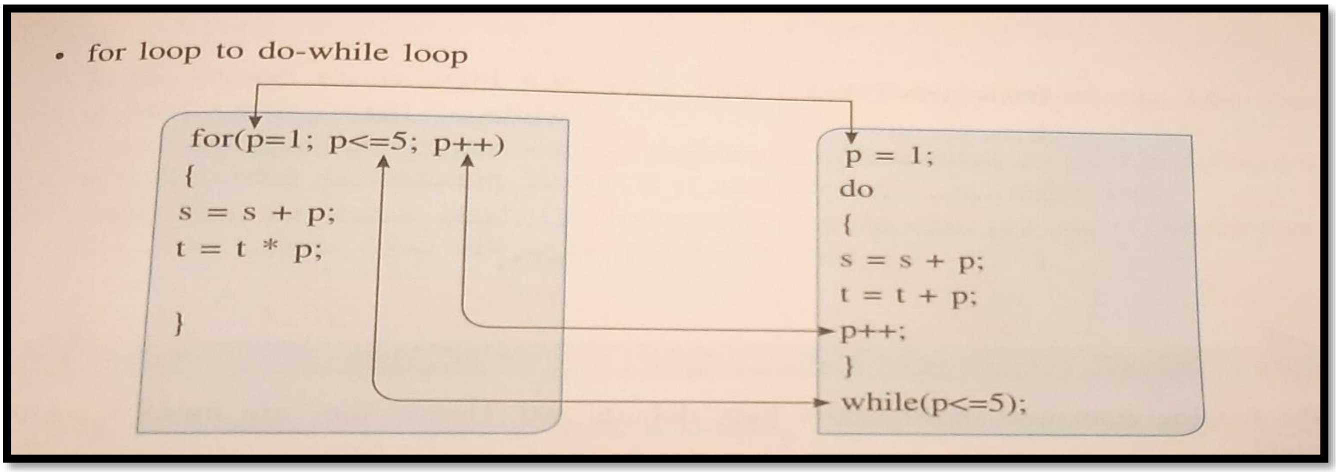
```
for(p=1; p<=5; p++)
{
s = s + p;
t = t * p;

}
```

```
p = 1;
while(p<=5)
{
s = s + p;
t = t + p;
p++;
}
```

- while loop to do-while loop

```
p = 1;
while(p<=5)
{
s = s + p;
t = t * p;
p++;
}
```

```
p = 1;
do
{
s = s + p;
t = t * p;
p++;
}
while(p<=5);
```



- for loop to do-while loop

```
for(p=1; p<=5; p++)
{
s = s + p;
t = t * p;

}
```

```
p = 1;
do
{
s = s + p;
t = t + p;
p++;
}
while(p<=5);
```

<u>Some important terms used with loop</u>

Here are some key differences between these loop types:Finite
1.**Finite loop**: A loop that runs a fixed number of times based on a condition.
2.**Infinite loop**: A loop that runs indefinitely unless manually terminated or a special control statement is used to exit.
3.**Empty or null loop**: A loop that does not contain any statements and is typically used to create a delay or wait for an event to occur.