# **Loops**

**What types of loops are supported by java?**

Java offers three different types of loops: for, while, and do-while.

A for loop provides a way to iterate over a range of values. It’s most useful when we know in advance how many times a task is going to be repeated.

A *while* loop can execute a block of statements while a particular condition is *true*.

A *do-while* is a variation of a *while* statement in which the evaluation of the *Boolean* expression is at the bottom of the loop. This guarantees that the code will execute at least once.

**What is an enhanced loop?**

Is another syntax of the *for* statement designed to iterate through all the elements of a collection, array, enum or any object implementing the *Iterable* interface.

Syntax:

for(datatype variable : array/collection){

//code here

}

**What is the difference between a break statement and a continue statement?**

A break statement results in the termination of the statement to which it applies (switch, for, do, or while). A continue statement is used to end the current loop iteration and return control to the loop statement.

**How can you exit anticipatedly from a loop?**

Using break statement.

**What is the difference between a labelled and an unlabelled break statement?**

An unlabelled *break* statement terminates the innermost *switch*, *for*, *while* or *do-while* statement, whereas a labelled *break* ends the execution of an outer statement.

**What is the difference between an unlabelled and a labelled continue statement?**

An unlabelled *continue* statement skips to the end of the current iteration in the innermost *for*, *while*, or *do-while* loop, whereas a labelled *continue* skips to an outer loop marked with the given label.

**What is the difference between while loop and do-while loop?**

While loop doesn’t execute the code inside the loop if the conditions meet. But do-while loop executes it at least once.

Example:

while loop:

**package** package1;

**public** **class** JavaExamples {

**public** **static** **void** main(String[] args) {

**int** j = 5;

**while**(j<3) {

System.***out***.println("Value of j is "+j);

}

}

}

In the above while loop value of j is greater than 3. So it won’t print anything.

do-while loop:

**package** package1;

**public** **class** JavaExamples {

**public** **static** **void** main(String[] args) {

**int** j = 5;

**do** {

System.***out***.println("Value of j is "+j);

}**while** (j<3);

}

}

In the above do-while loop, even though the value of j is greater than 3, it still executes once as the condition is mentioned at the bottom.

**Can a for statement loop indefinitely?**

Yes, a for statement can loop indefinitely. For example, consider the following: for(;;);

**package** package1;

**public** **class** JavaExamples {

**public** **static** **void** main(String[] args) {

**for**(;;)

{

System.***out***.println("hello");

System.***out***.println("how are you?");

}

}

}

The above will continuously iterate and prints the statements.