Loops
in Java

Introduction

In a loop, a part of a program is repeated over and over until a specific goal is reached. Loops will continue to execute statements, called the body of the loop, as long as the condition holds true.
In Java, there are three types of loops:

- **while** loops
- **for** loops
- **do** loops

Important Note:
When you declare a variable inside the loop body, the variable is created for each iteration and removed after the end of each iteration.

while and for Loops

Syntax for **while** loops:

```
while (condition)
{
    statements
)
```

Generally, the variable for the condition is declared outside of the loop and updated in the loop.

In **for** loops, the variable for the condition is initialized in the for statement.
Syntax for **for** loops:

```
for (initialization; condition; update)
{
    statements
}
```


Example

Jack earns $12 a day babysitting and wants to know how long it'll take him to earn at least $500.

while loop:

```
int m = 0; //m is how much he's earned
int days = 0;
while (m < 500)
{
    m = m + 12;
    days = days + 1;
}
System.out.println(days);
```

for loop:

```
int days = 0;
for (int m=0; m<500; m=m+12)
{
    days = days + 1;
}
System.out.println(days);
```


do Loops

In a **do** loop (or **do-while** loop), the body of the loop is executed first, and then the condition is tested.

Syntax:

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

A typical example for a do loop is input validation. You want to ask a user for an input before checking whether it is acceptable, and you keep asking for an input if it is unacceptable.

Operators in Loops

In loops, variables are constantly being incremented and decremented by certain values. There are various operators that we can use to do this.

For a variable x and any value y...

Operator	Usage	Meaning
++	x++	x = x+1
--	x--	x = x-1
+=	x+=y	x = x+y
-=	x-=y	x = x-y

Common Errors

One common mistake in loops is the infinite loop. This occurs when the condition is never met, and the loop will continue to run forever. This can occur if you forget to update the variable used in the condition or if you increment instead of decrement the variable (or vice versa).

Another common mistake is the off-by-one error. It is easy to be off by one iteration of the loop. This can be avoided by manually tracing the variables as the loop runs.

Practice

1. What does the following loop print?

```
int n = 1;
while (n < 100)
{
    n = 2*n;
    System.out.print(n + " ");
}
```

2. What type of error do you see in the following code?

```
int n = 1;
while (n != 50)
{
    System.out.println(n);
    n = n+10;
}
```