

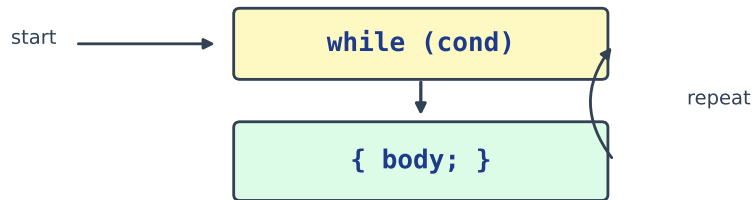
Loops

Java Reference

while loops

A **while** loop repeats while a condition is true. Change something inside, or it loops forever.

```
public class Main {  
    public static void main(String[] args) {  
        int n = 1;  
        while (n <= 3) {  
            System.out.println(n);  
            n++;  
        }  
    }  
}
```



Exam tip: condition is checked before each pass; body must eventually make it false

while checks the condition before each pass of the body

for loops

A **for** loop packs the start, the condition, and the step into one line. Best when you know the count.

```
public class Main {  
    public static void main(String[] args) {  
        for (int i = 0; i < 5; i++) {  
            System.out.print(i + " ");  
        }  
        System.out.println();    // 0 1 2 3 4  
    }  
}
```

Accumulation

The accumulator pattern: start a variable before the loop, then update it each turn.

```
public class Main {  
    public static void main(String[] args) {  
        int total = 0;  
        for (int i = 1; i <= 5; i++) {  
            total += i;  
        }  
        System.out.println(total);    // 15  
    }  
}
```

Nested loops

A loop inside a loop is a nested loop . The inner loop runs fully on each turn of the outer one.

```
public class Main {  
    public static void main(String[] args) {  
        for (int r = 0; r < 3; r++) {  
            for (int c = 0; c < 3; c++) {  
                System.out.print("*");  
            }  
            System.out.println();  
        }  
    }  
}
```

Common mistakes

- `for (int i = 0; i < n; i++)` runs `n` times (0 to `n - 1`); using `<=` runs one extra.
- Do not put a semicolon right after `for (...)` or `while (...)` — it makes an empty loop.
- Declare the counter in the `for` header so its scope ends with the loop.