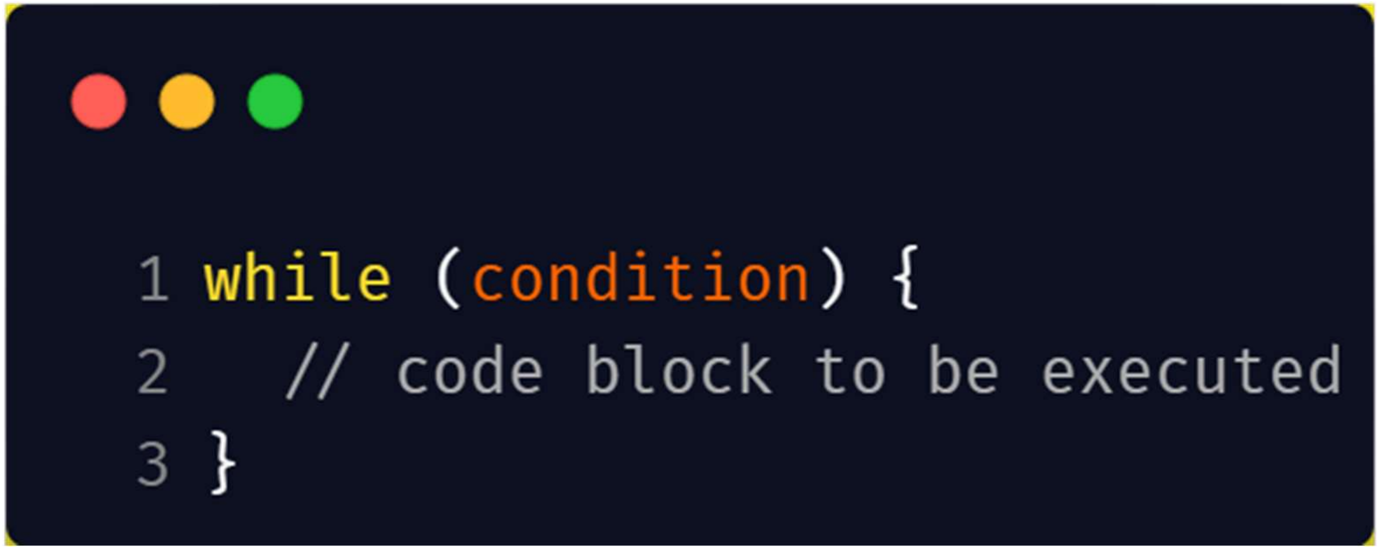
Java Programming
Chapter 4

Loop

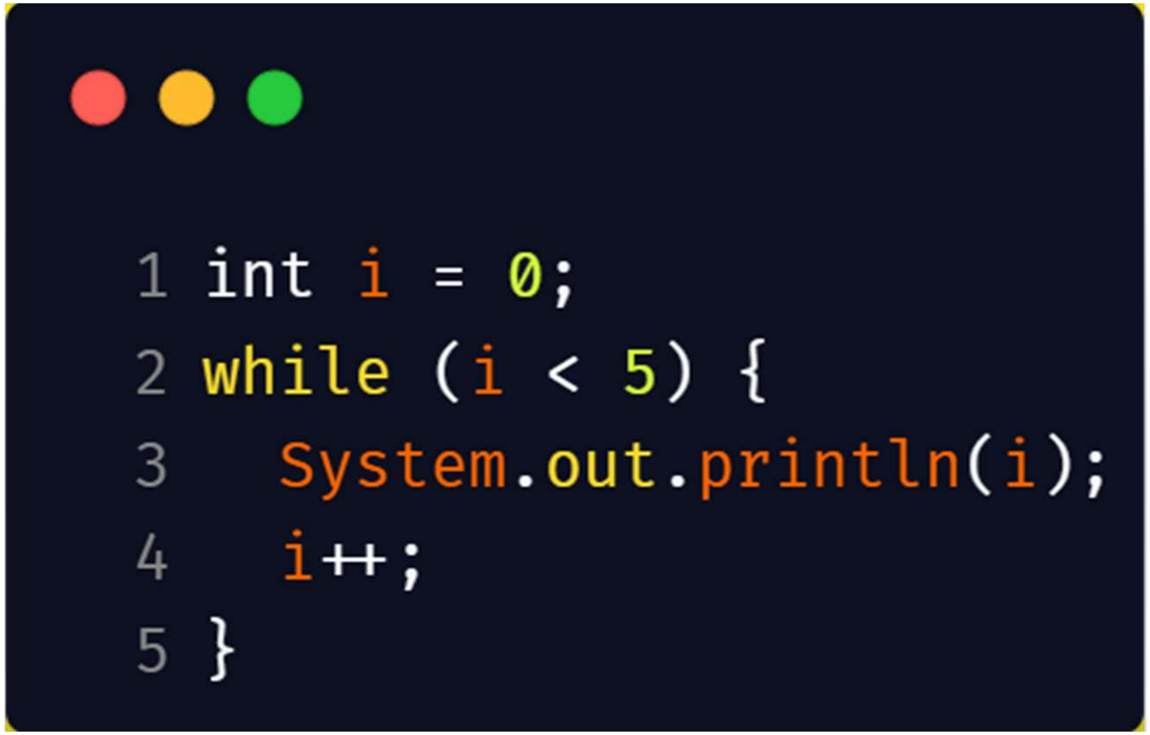
Loops

- Loops can execute a block of code as long as a specified condition is true.
- Loops are handy because they save time, reduce errors, and they make code more readable.



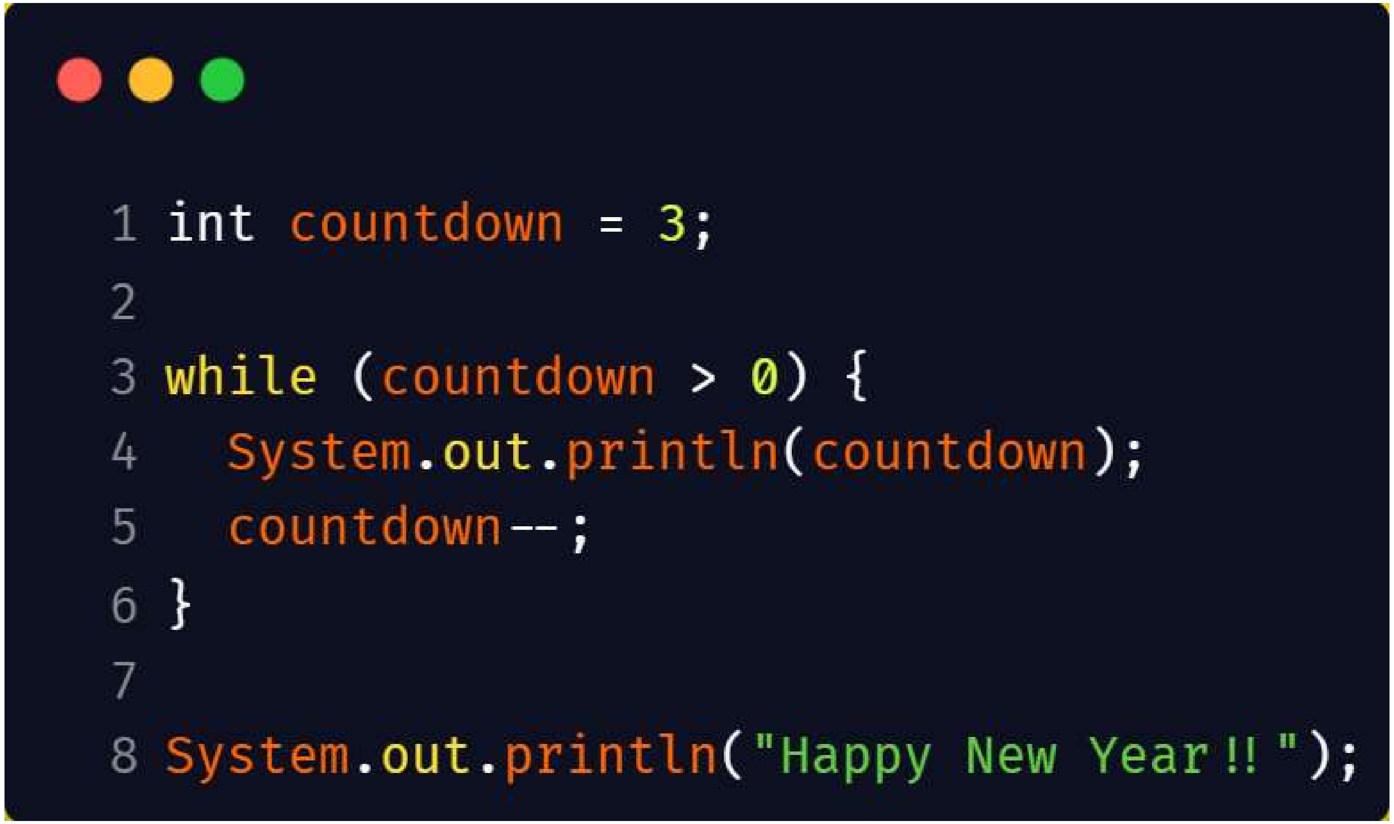
```
while (condition) {
  // code block to be executed
}
```

- In the example below, the code in the loop will run again and again, as long as a variable (i) is less than 5:



```java
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

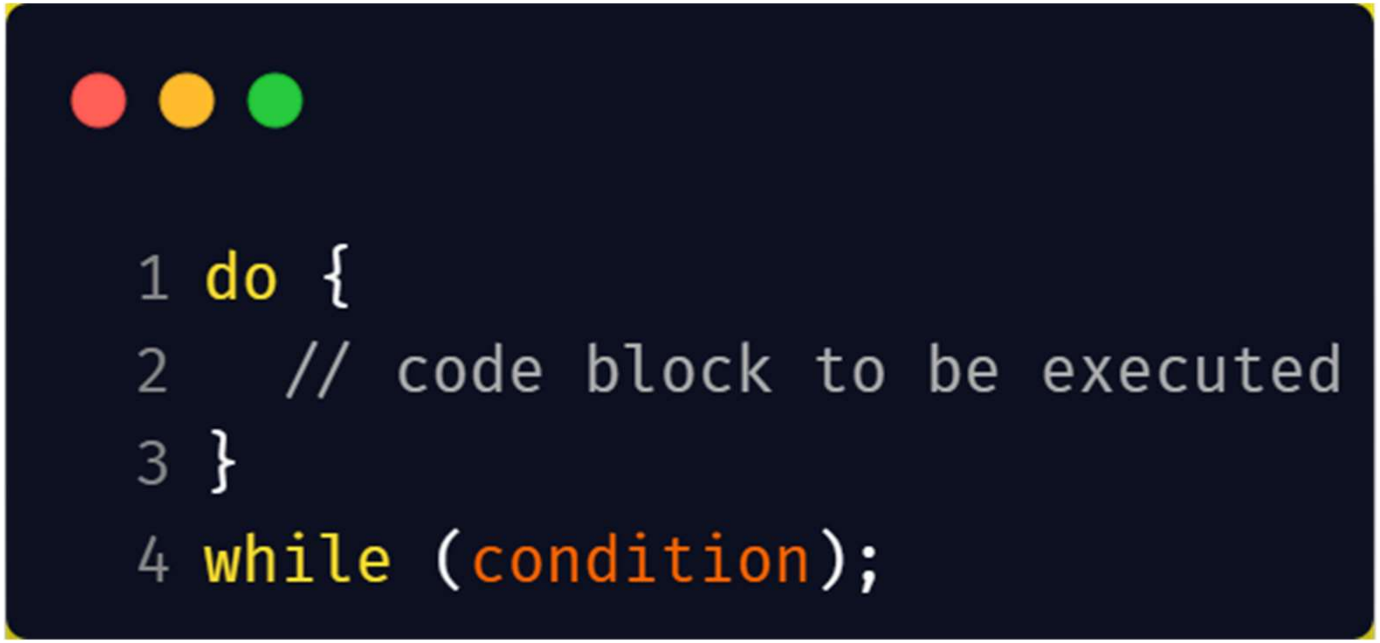
- Countdown Example
  - You can also use a while loop to count down.
  - This example counts from 3 to 1, and then prints "Happy New Year!!" at the end:



```java
int countdown = 3;

while (countdown > 0) {
  System.out.println(countdown);
  countdown--;
}

System.out.println("Happy New Year!!");
```

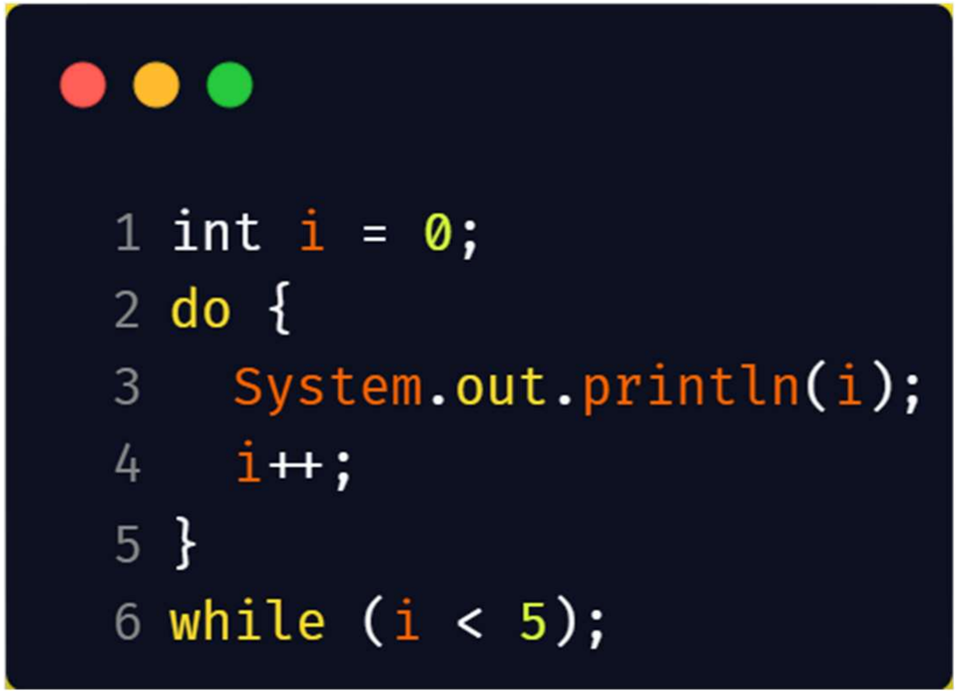
The Do/While Loop

- The do/while loop is a variant of the while loop.
- This loop will execute the code block once, before checking if the condition is true. Then it will repeat the loop as long as the condition is true.
- Syntax



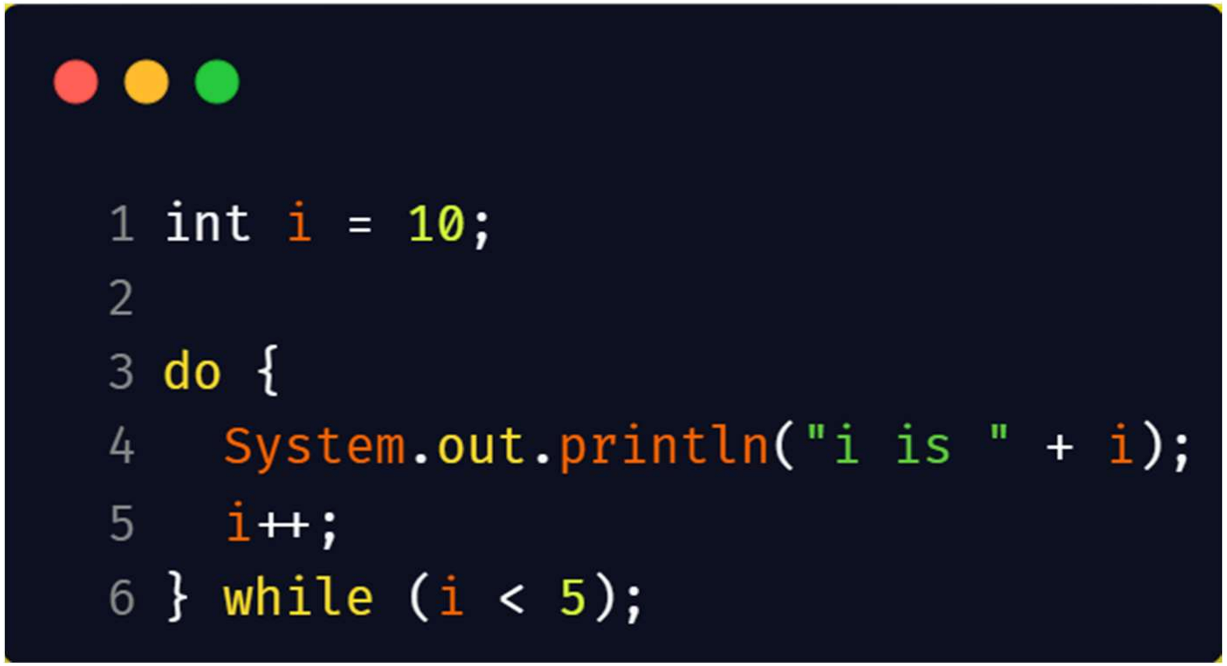
```
do {
  // code block to be executed
}
while (condition);
```

- The example below uses a do/while loop.
- The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:



```java
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

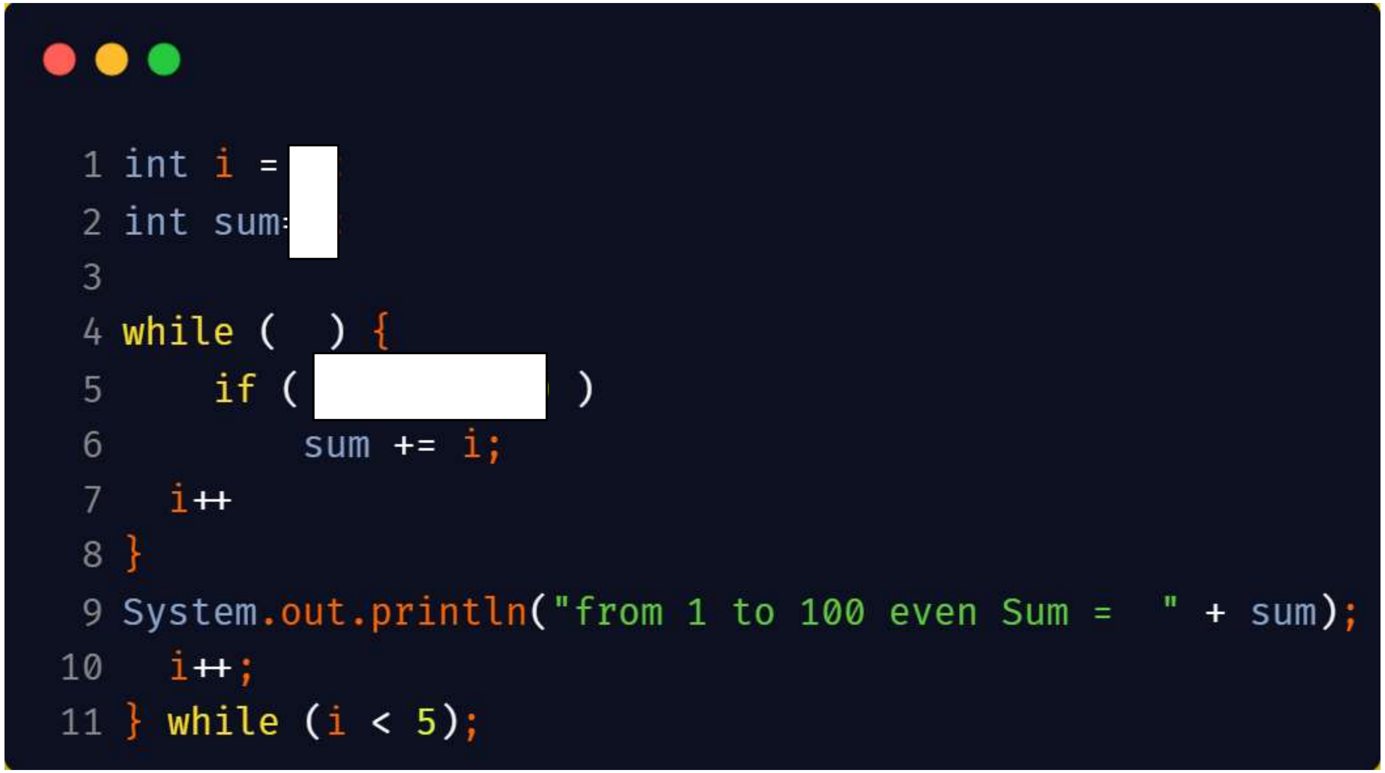
- Condition is False from the Start
  - In the while loop , we saw that if the condition is false at the beginning, the loop never runs at all.
  - The do/while loop is different: it will always run the code block at least once, even if the condition is false from the start.



```java
int i = 10;

do {
  System.out.println("i is " + i);
  i++;
} while (i < 5);
```

Excercise

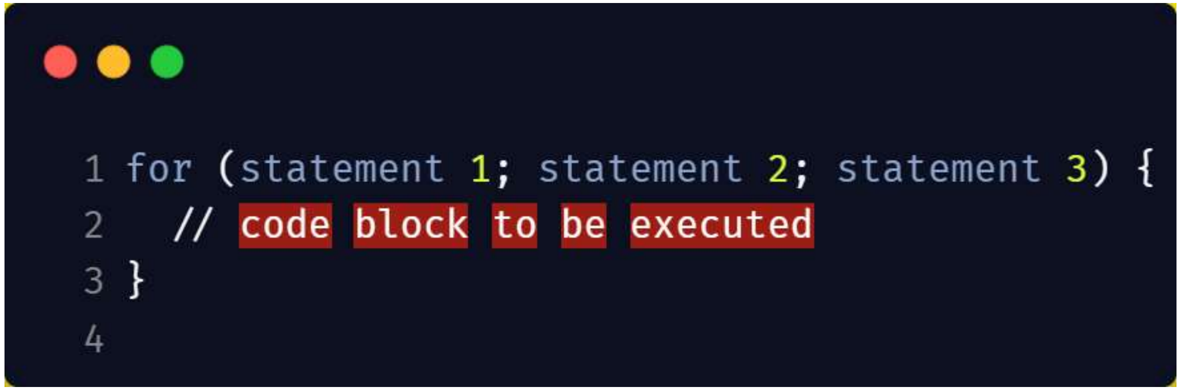
- Let's find the sum of even numbers from 1 to 100.
- fill the white blank !!



```
int i = 
int sum=

while (   ) {
    if (             )
        sum += i;
  i++
}
System.out.println("from 1 to 100 even Sum =  " + sum);
  i++;
} while (i < 5);
```

For loop

- When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:
- Syntax

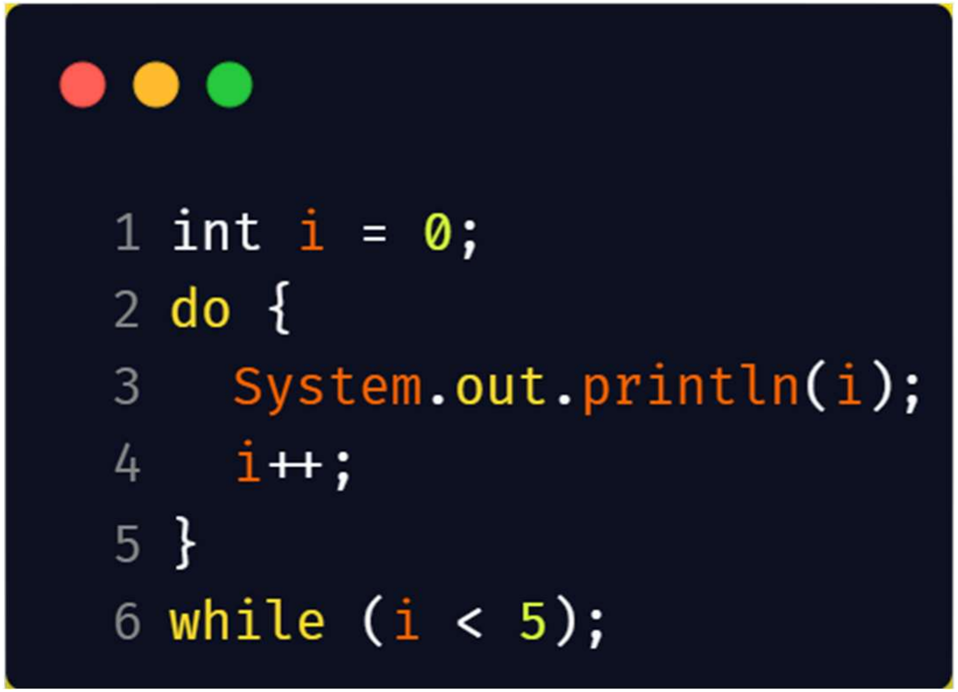


```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}

```

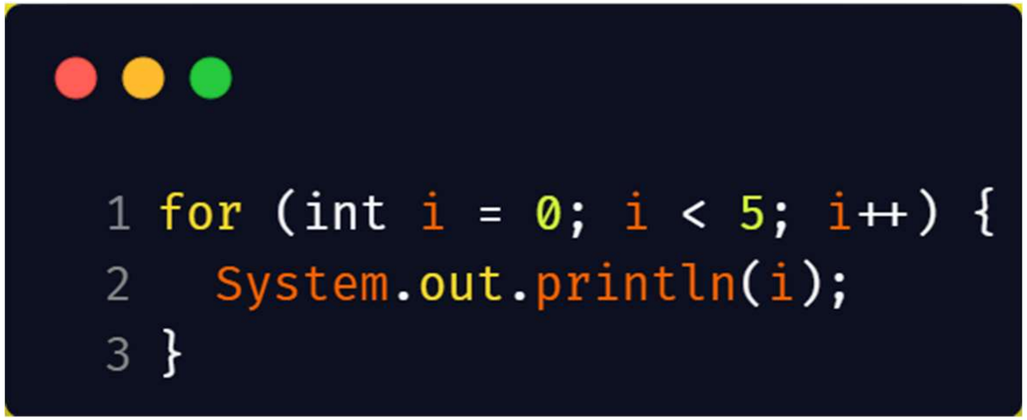
  - Statement 1 is executed (one time) before the execution of the code block.
  - Statement 2 defines the condition for executing the code block.
  - Statement 3 is executed (every time) after the code block has been executed.

For loop

- Print Numbers
  - The example below will print the numbers 0 to 4:



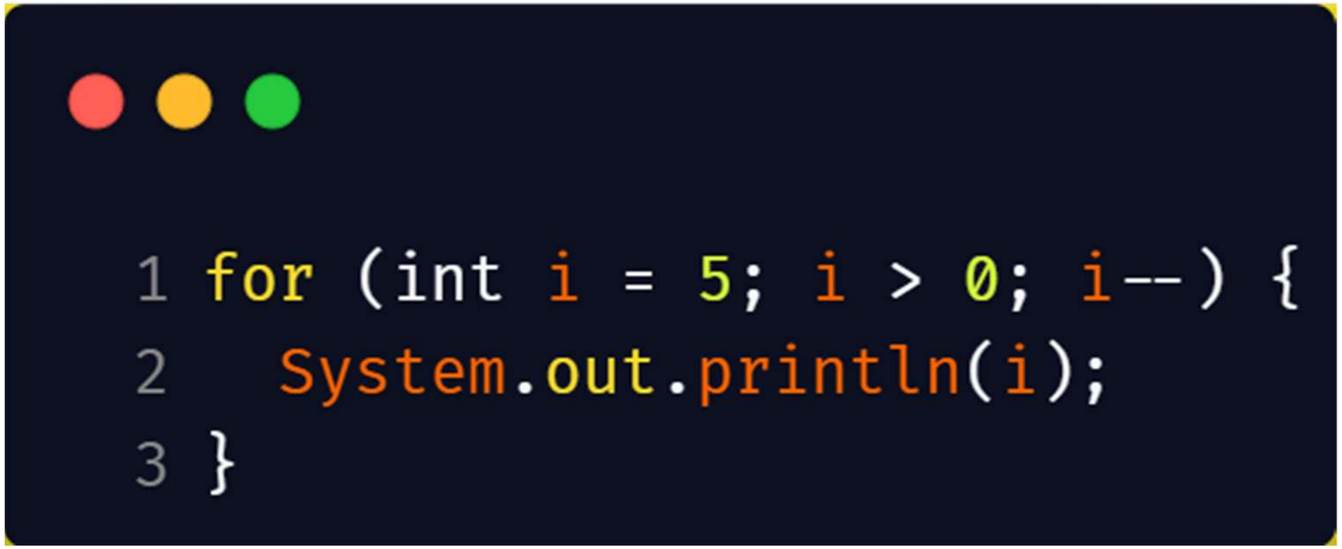
```java
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```



```java
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}
```

For loop

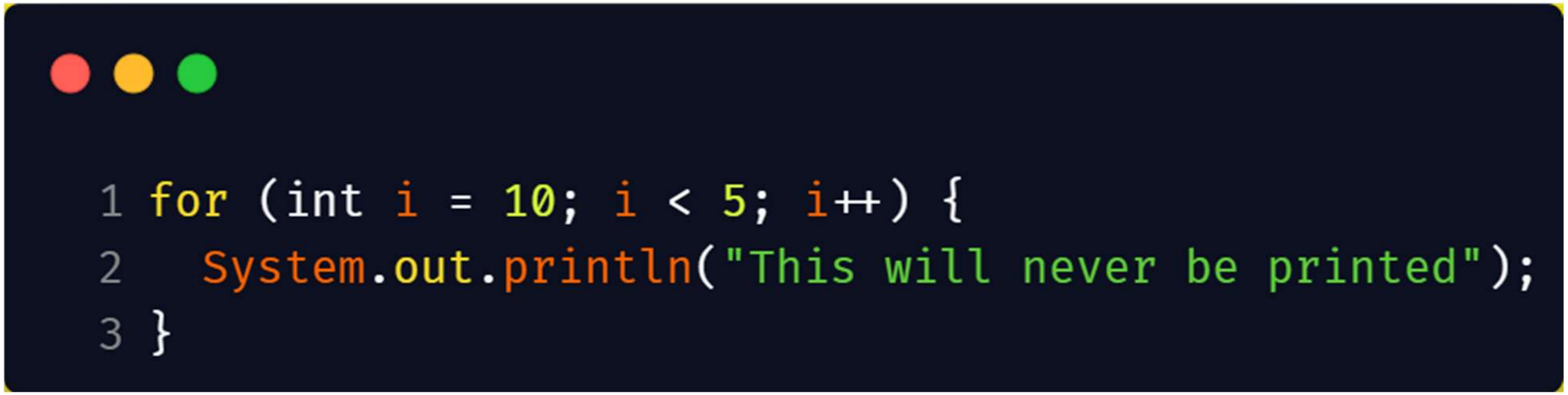
- Countdown



```java
for (int i = 5; i > 0; i--) {
  System.out.println(i);
}
```

For loop

- Why is there no output?

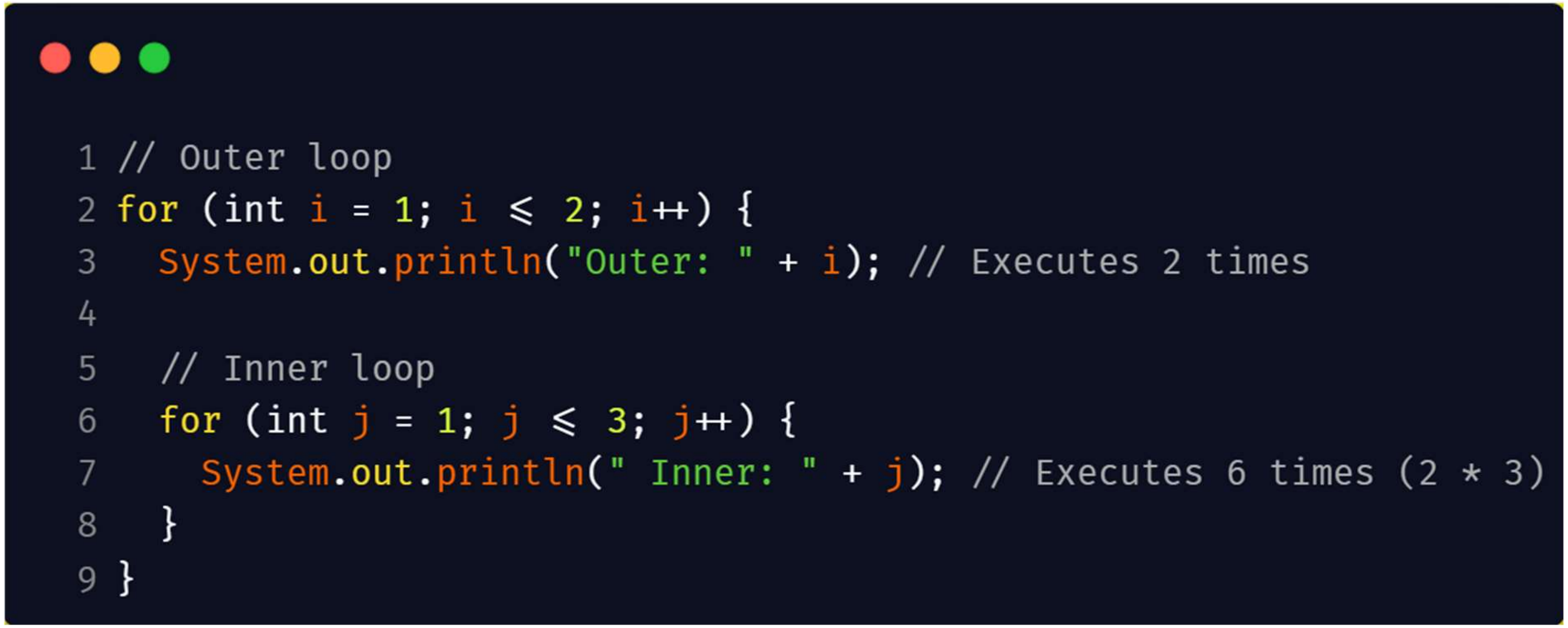


```java
for (int i = 10; i < 5; i++) {
  System.out.println("This will never be printed");
}
```

- Let's try to fix it

Nested Loops

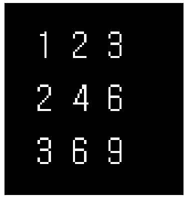
- It is also possible to place a loop inside another loop. This is called a nested loop.
- The "inner loop" will be executed one time for each iteration of the "outer loop":



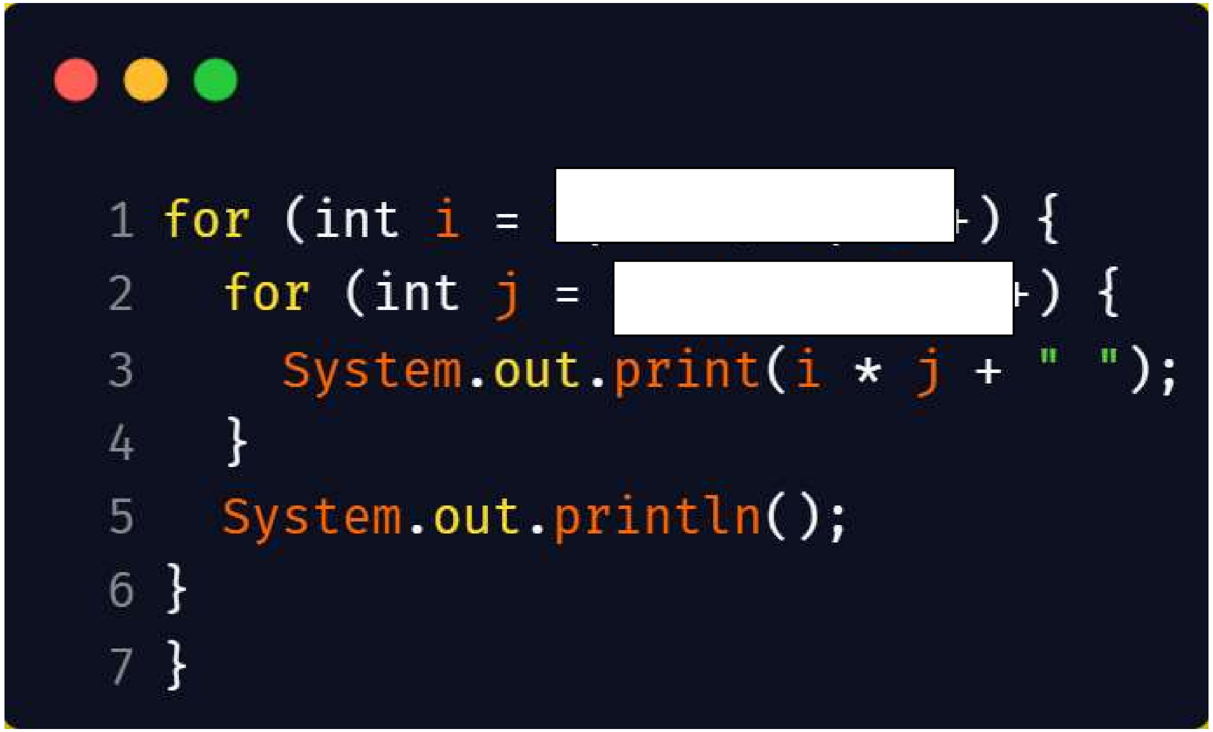
```
// Outer loop
for (int i = 1; i <= 2; i++) {
  System.out.println("Outer: " + i); // Executes 2 times

  // Inner loop
  for (int j = 1; j <= 3; j++) {
    System.out.println(" Inner: " + j); // Executes 6 times (2 * 3)
  }
}
```

Nested Loops

- Print this result
- Fill in the blank



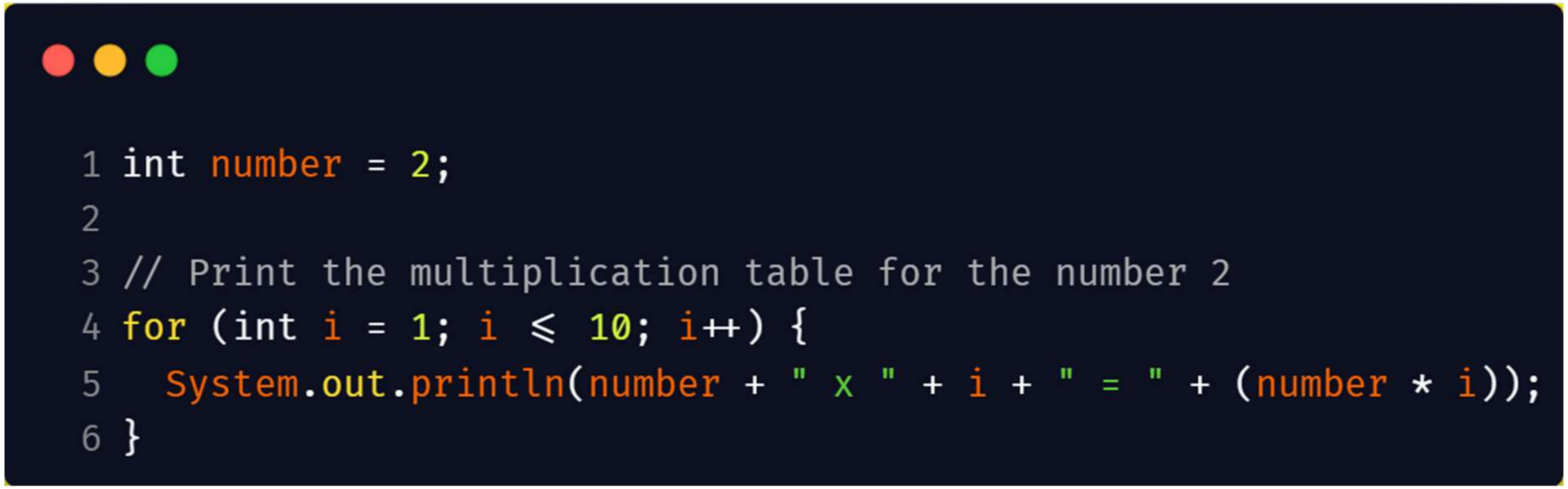
```
1 2 3
2 4 6
3 6 9
```



```java
for (int i =                  ) {
  for (int j =                  ) {
    System.out.print(i * j + " ");
  }
  System.out.println();
}
}
```

Nested Loops

- prints the multiplication table for a specified number:



```java
int number = 2;

// Print the multiplication table for the number 2
for (int i = 1; i <= 10; i++) {
  System.out.println(number + " x " + i + " = " + (number * i));
}
```