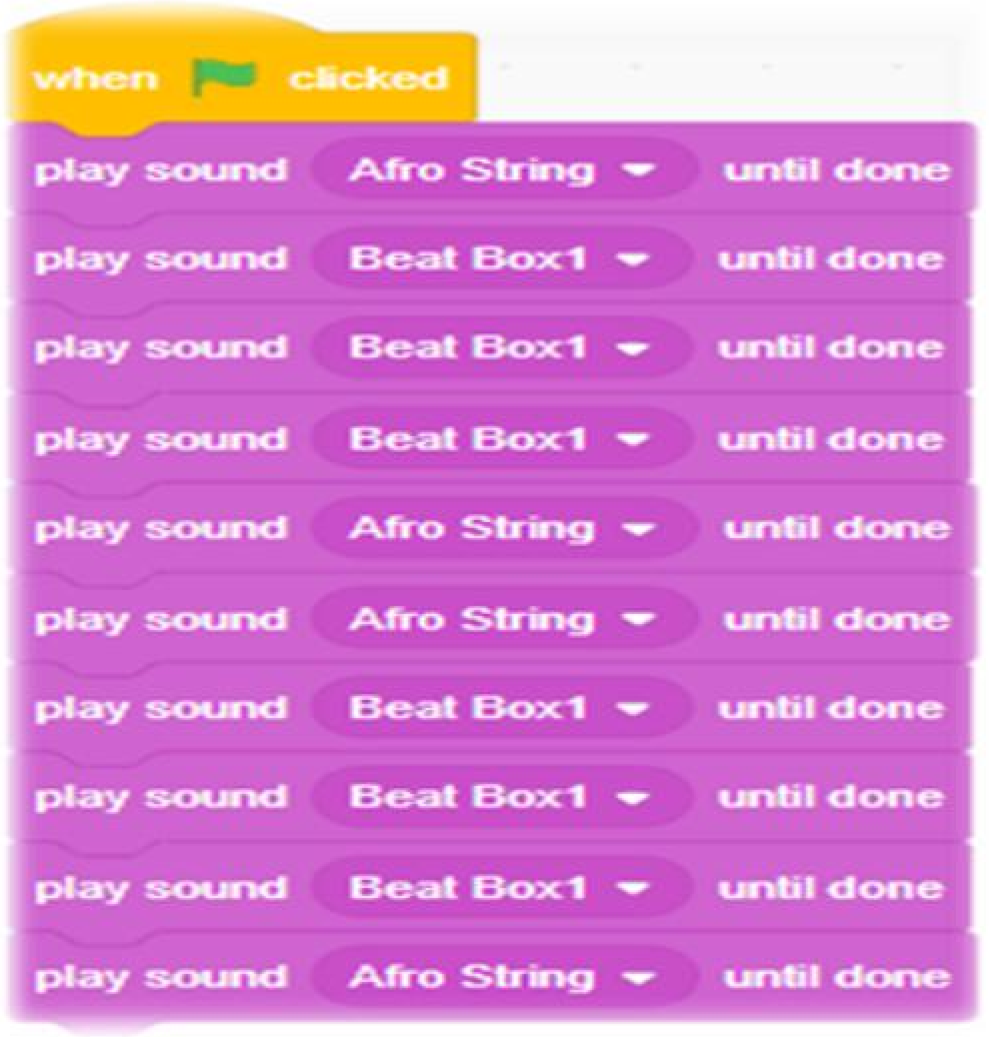
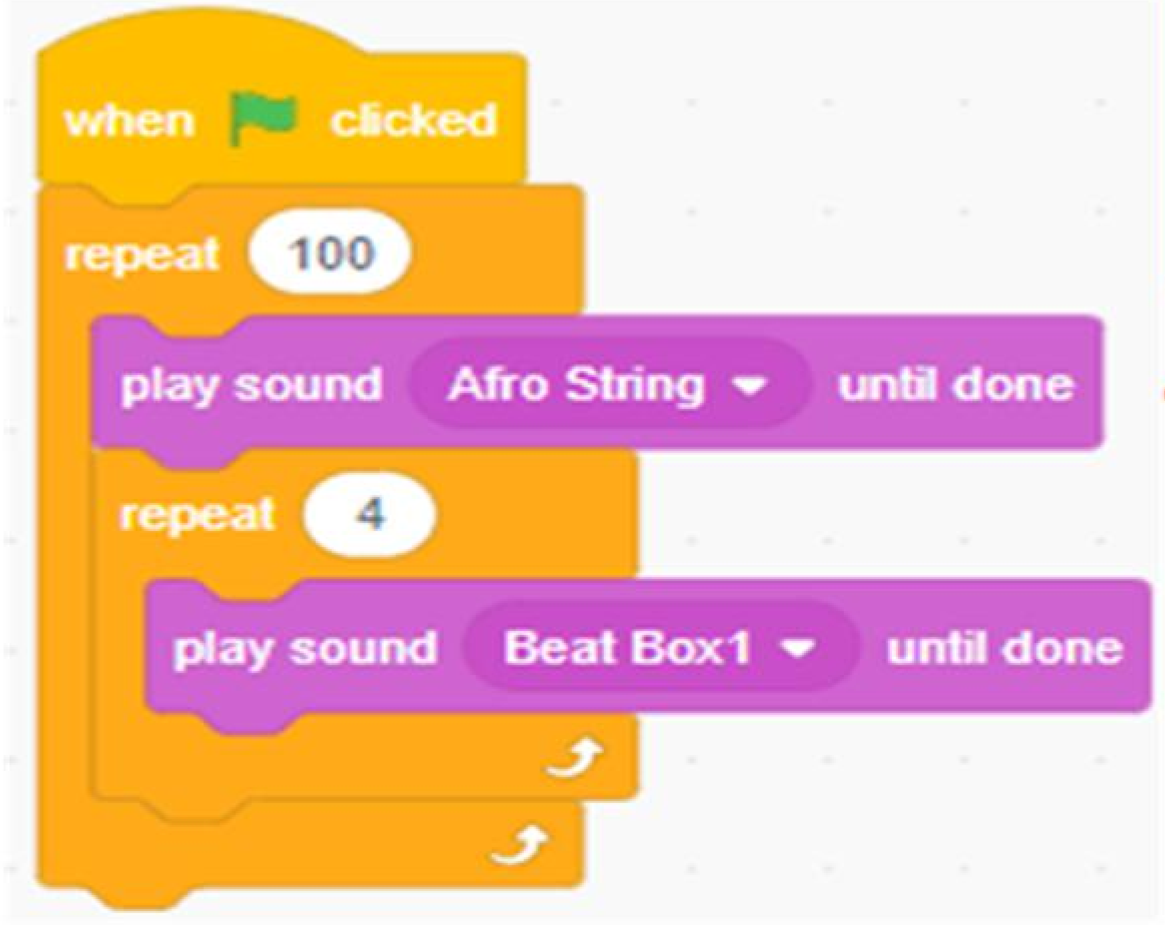
Playing Multiple Soundtracks



```
when [flag] clicked
play sound Afro String until done
play sound Beat Box1 until done
play sound Beat Box1 until done
play sound Beat Box1 until done
play sound Afro String until done
play sound Afro String until done
play sound Beat Box1 until done
play sound Beat Box1 until done
play sound Beat Box1 until done
play sound Afro String until done
```

I wish to play 2 soundtracks, 400 times in the correct sequence. How can I do this?

Playing Soundtracks



```
when [flag] clicked
repeat 100
    play sound Afro String until done
    repeat 4
        play sound Beat Box1 until done
```

How can you use Java to perform this function?

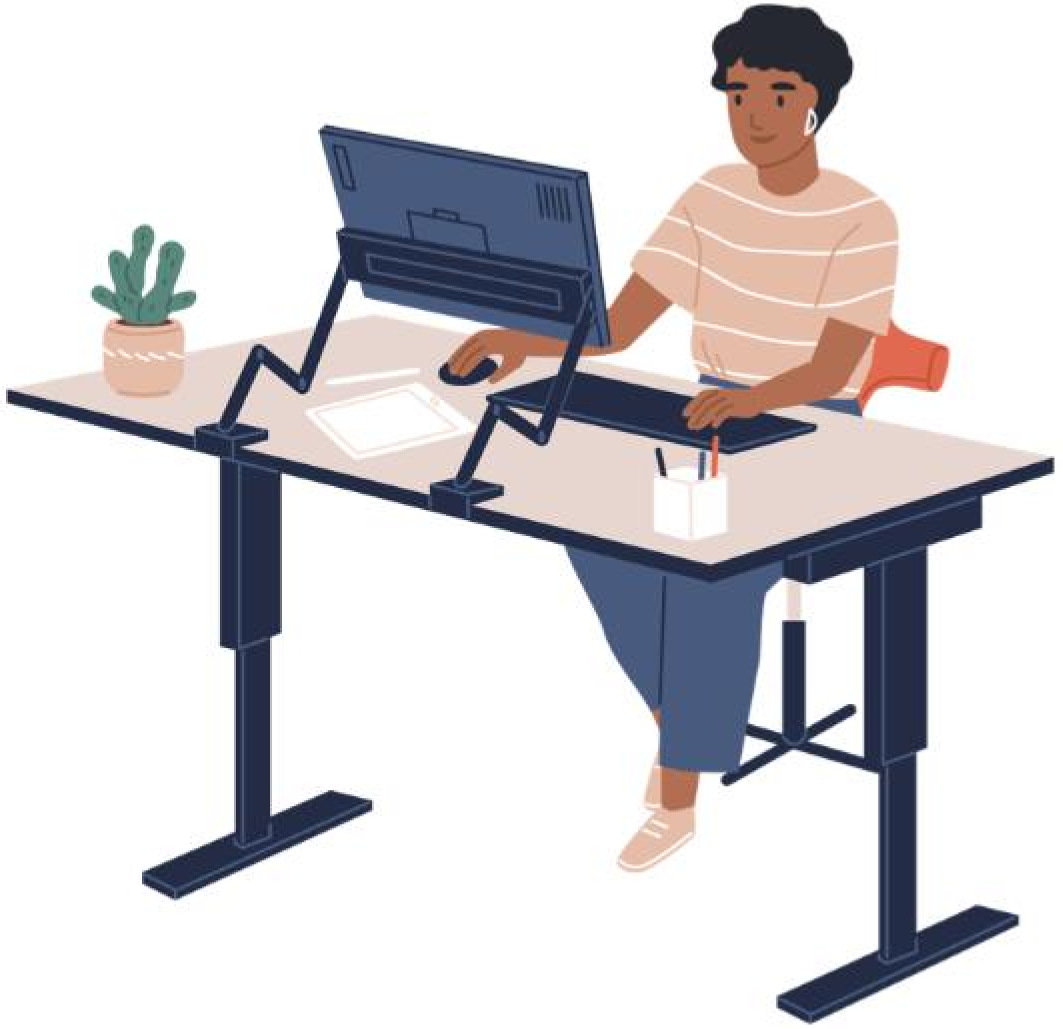
Loops in Java

Entry Point

How can similar problems be solved by using Java?
Are you aware of few ways to do so?

Exit Point

Looping Constructs



Learning Objective

- Define loops in Java
- Implement for loop
- Implement while loop
- Create do…while loop

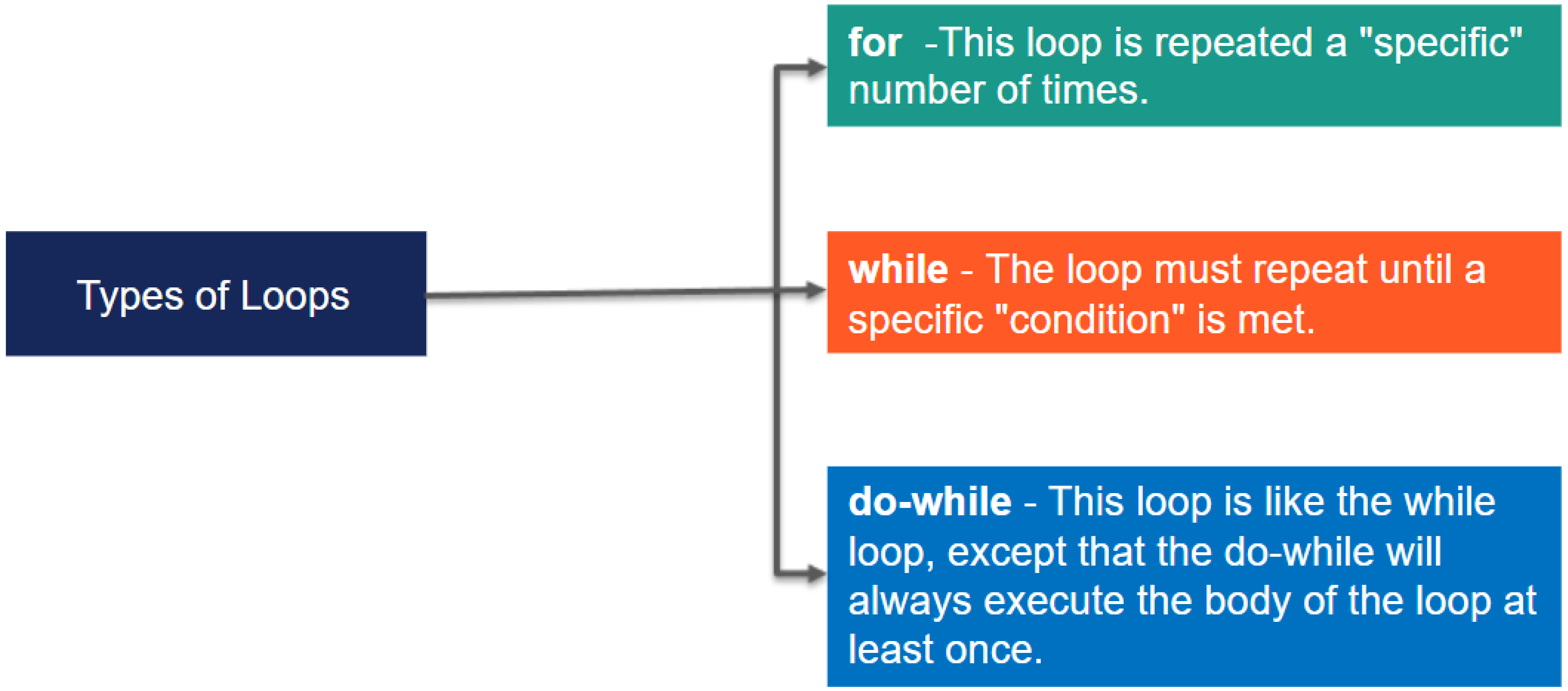
Think and Tell

- John wants to find an even and odd number between 1 and 500.
  - How many lines of code does John need to write?
  - Should he create 500 variables and store the values, and then check the even and odd numbers?

Loops in Java

Loops

- Loops enable you to execute the same block of code several times.
- A set of repetitive instructions can be reduced to a single instruction by using a loop.



Types of Loops

for -This loop is repeated a "specific" number of times.

while - The loop must repeat until a specific "condition" is met.

do-while - This loop is like the while loop, except that the do-while will always execute the body of the loop at least once.

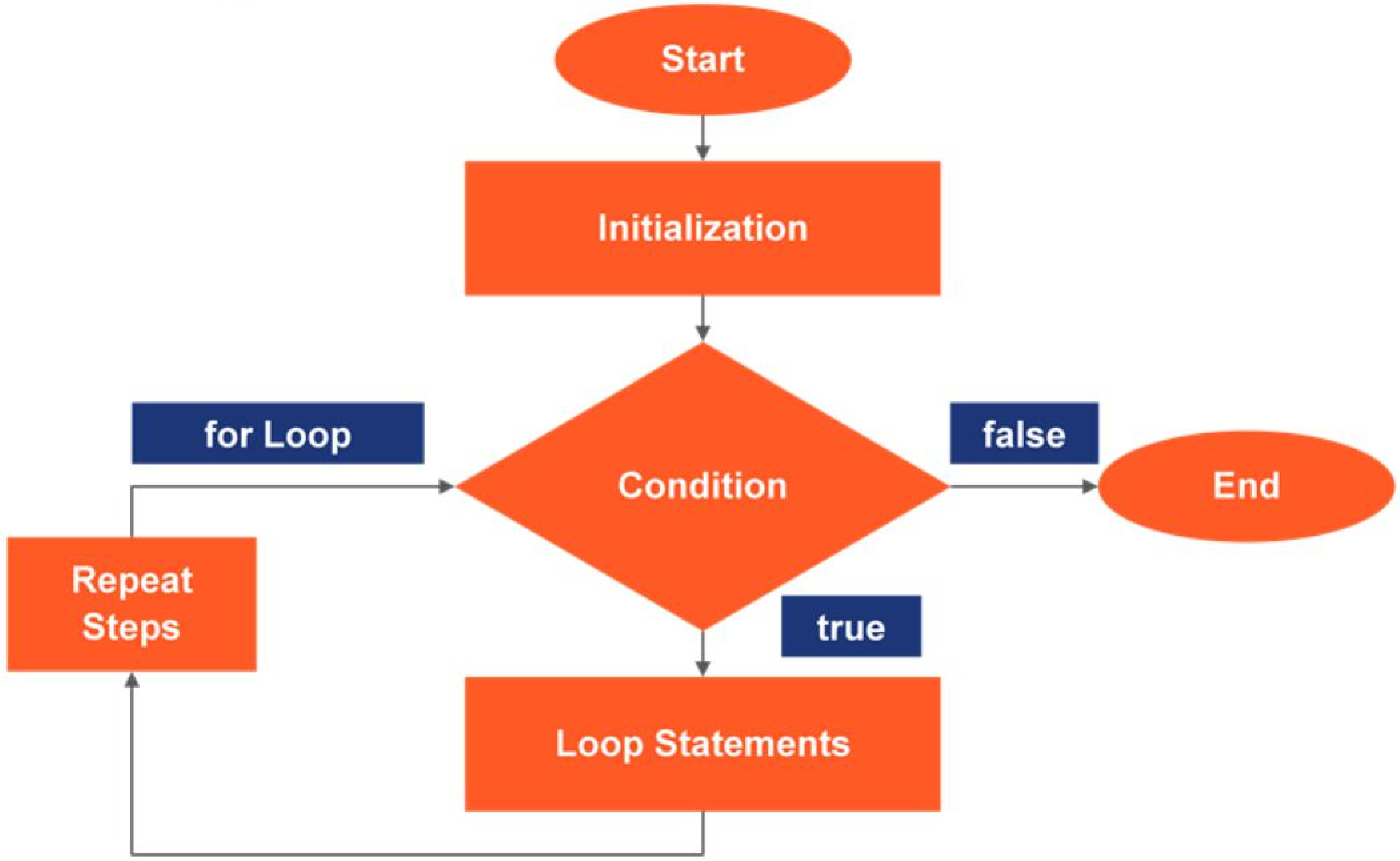
for-Loop

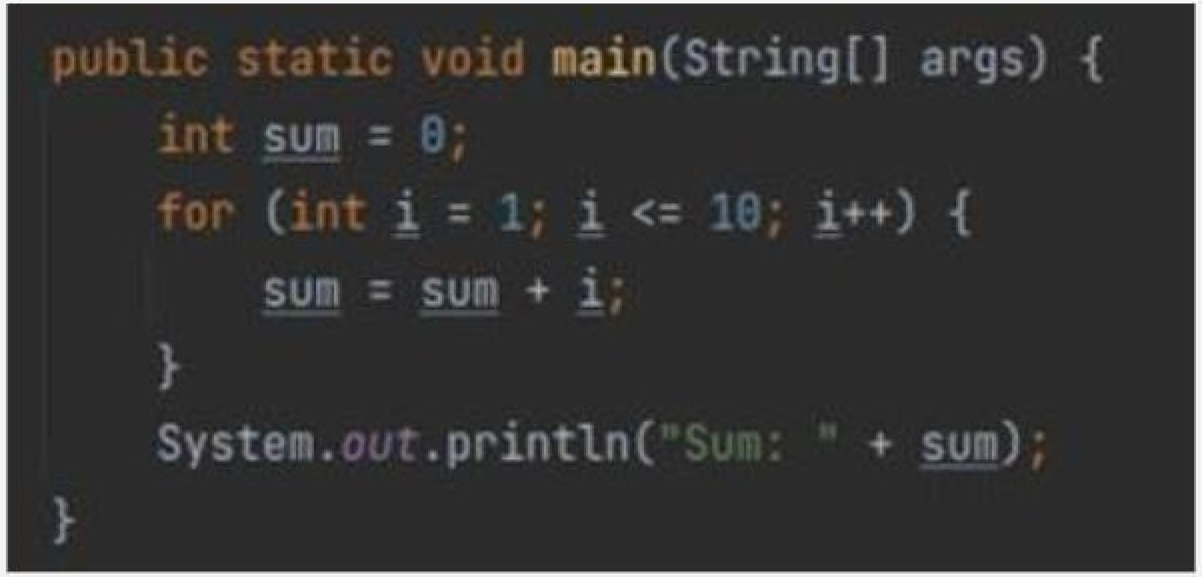
Java Syntax – for-Loop

```
for(initialization; condition; updation)
{
  statement_1
  statement_2
}
```

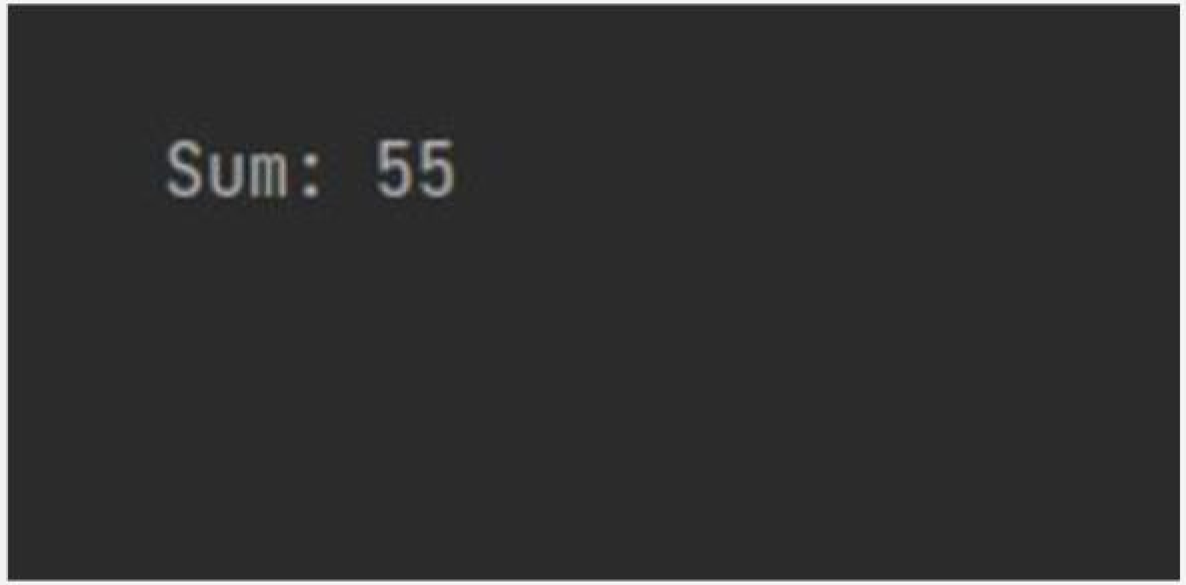
- The **initialization** step is executed first, and only once. Here, you can initialize the variable.
- Next, the **condition expression** is evaluated.
  - If it is true, the body of the loop is executed.
  - If it is false, the body of the loop will not be executed.
- **Updation** means incrementing or decrementing the variable value.

for-Loop Flow Diagram





```
public static void main(String[] args) {
    int sum = 0;
    for (int i = 1; i <= 10; i++) {
        sum = sum + i;
    }
    System.out.println("Sum: " + sum);
}
```



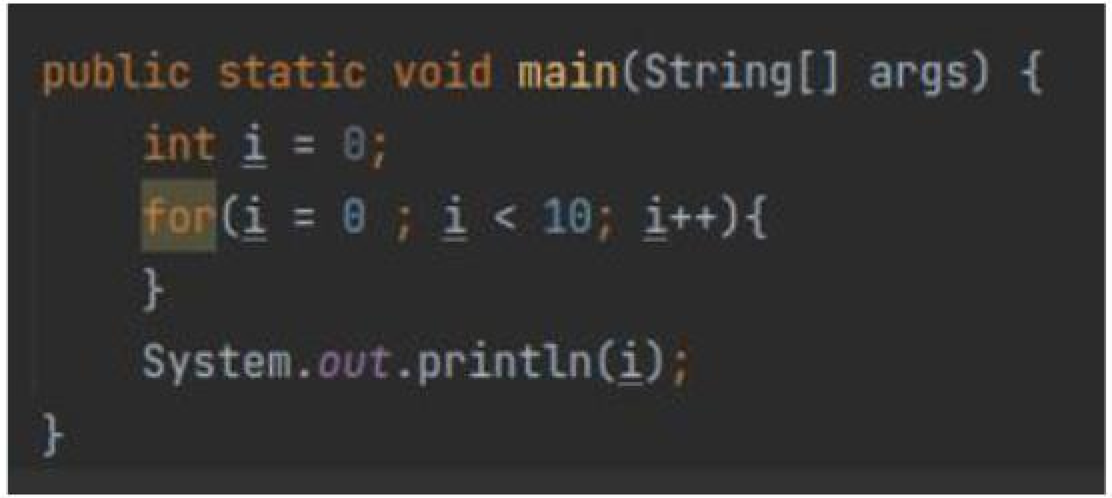
```
Sum: 55
```

for-Loop Demo

- This loop will print a sum of numbers starting from 1 to 10.
- The value of `i` is initialized with 1.
- The value of `i` is checked to determine whether it's less than or equal to 10 `(i<=10)`.
    - If this condition is true, then the block will be executed and the value of `i` is added to the sum variable.
- Thereafter, the value of `i` is incremented by 1.
- The loop will iterate until the condition is true.
- Once the condition becomes false, it will come out of the loop and print the value of the sum variable.

Quick Check

Predict the output for the following code:

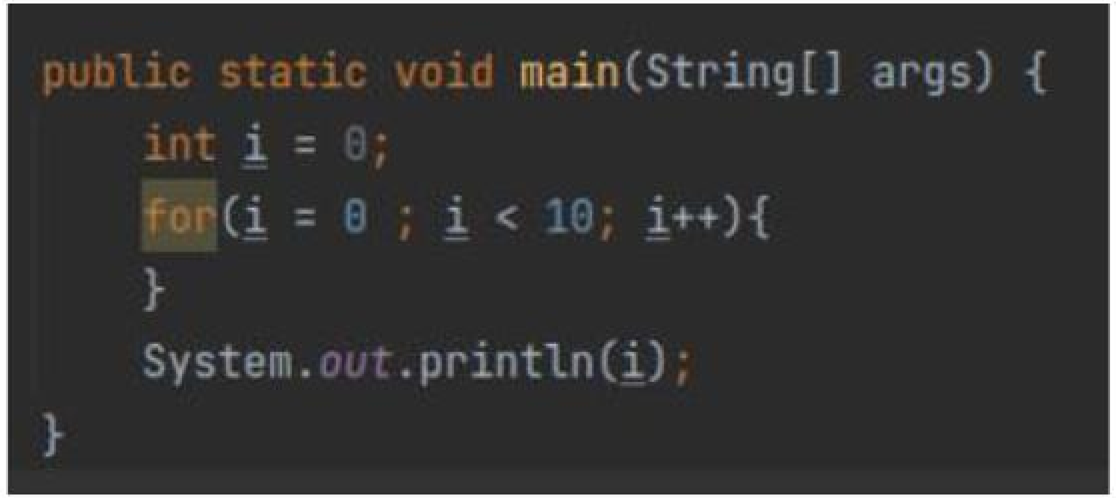


```
public static void main(String[] args) {
    int i = 0;
    for(i = 0 ; i < 10; i++){
    }
    System.out.println(i);
}
```

1. 10
2. Syntax Error
3. No output
4. 1 2 3 4 5 6 7 8 9 10

Quick Check: Solution

Predict the output for the following code:

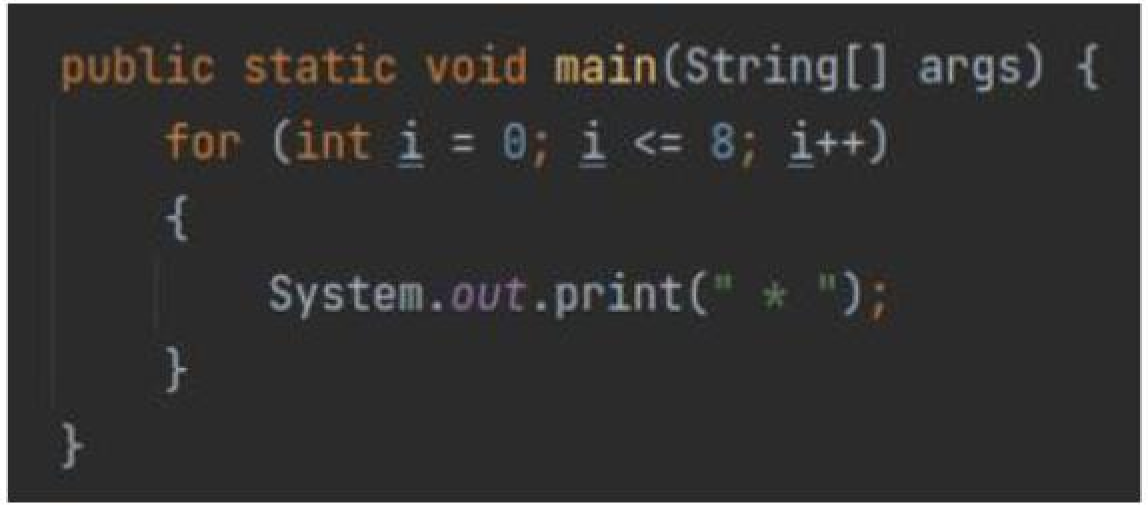


```
public static void main(String[] args) {
    int i = 0;
    for(i = 0 ; i < 10; i++){
    }
    System.out.println(i);
}
```

1. **10**
2. Syntax Error
3. No output
4. 1 2 3 4 5 6 7 8 9 10

Quick Check

How many times the loop will print the "*"

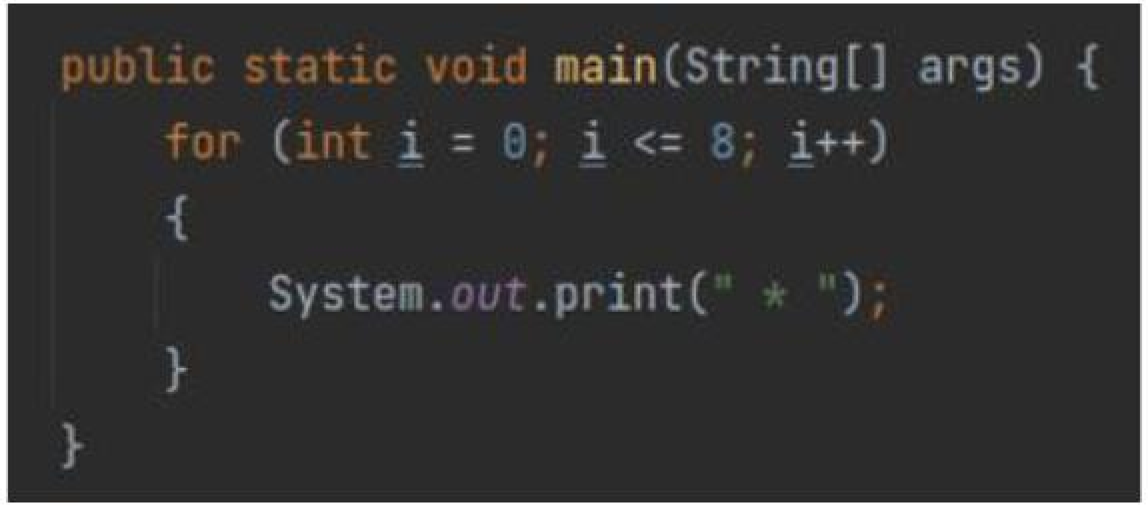


```
public static void main(String[] args) {
    for (int i = 0; i <= 8; i++)
    {
        System.out.print(" * ");
    }
}
```

1. 7
2. 8
3. 9
4. Syntax Error

Quick Check: Solution

How many times the loop will print the "*"



```
public static void main(String[] args) {
    for (int i = 0; i <= 8; i++)
    {
        System.out.print(" * ");
    }
}
```

1. 7
2. 8
3. **9**
4. Syntax Error

Even Numbers

Write a program that displays all even numbers between 1 and 50.

DEMO

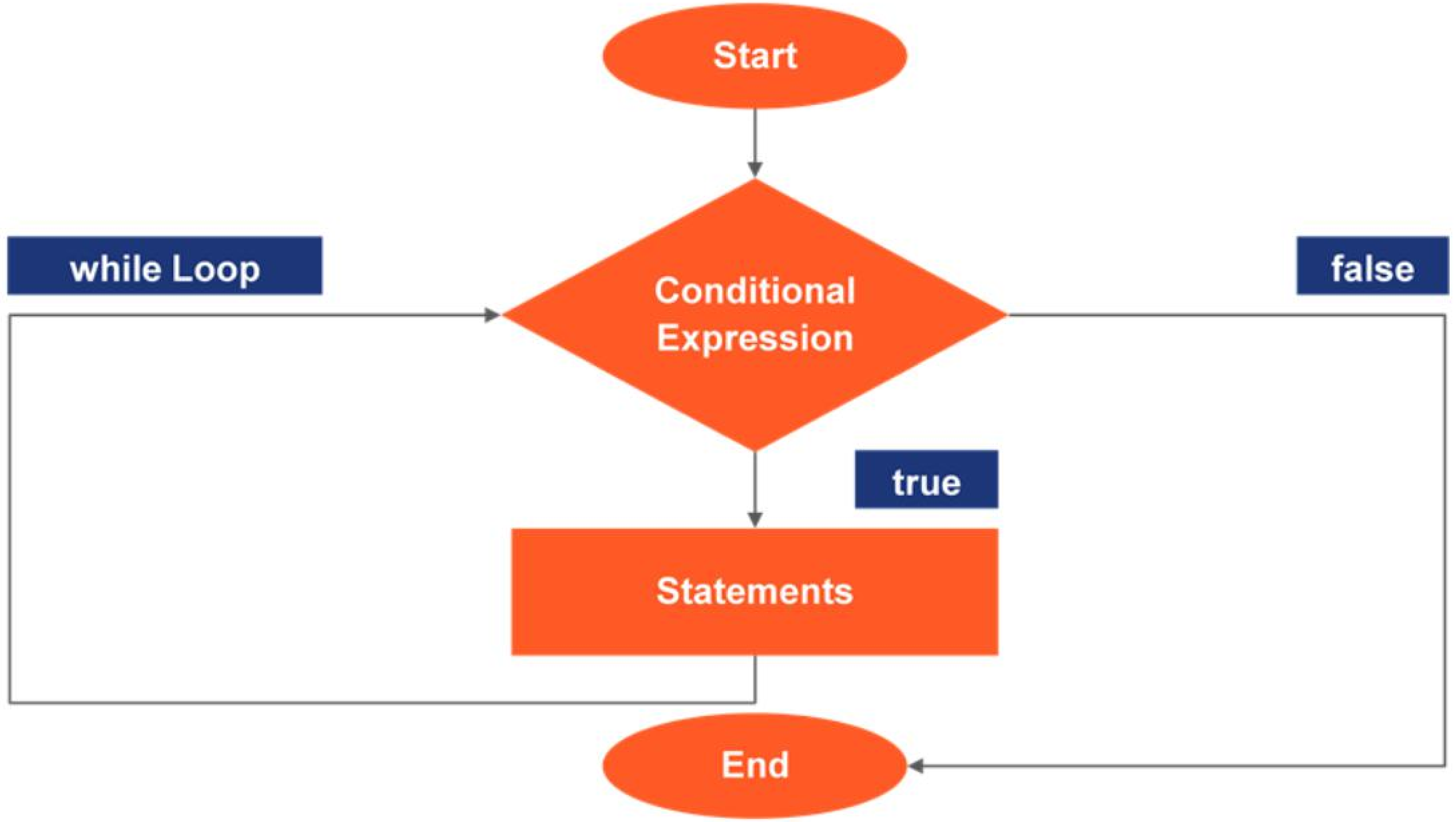
while-Loop

Java Syntax: while-Loop

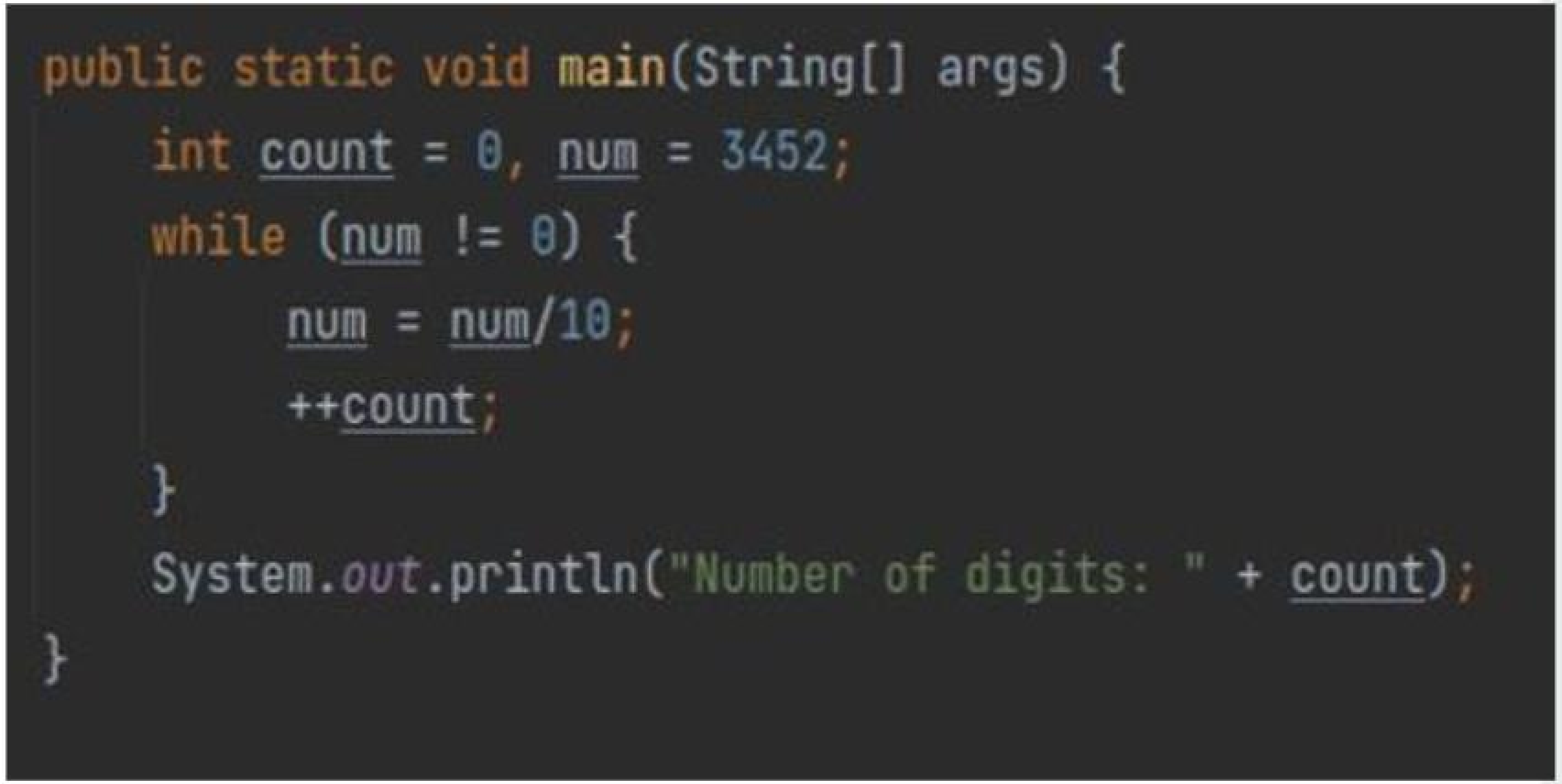
```
while (test_expression)
{
   // statements

}
```

- In a while loop, the first expression is to test the condition. If the condition is true, then the body of the loop will be executed.
- If the condition is false, then it will exit from the while loop.

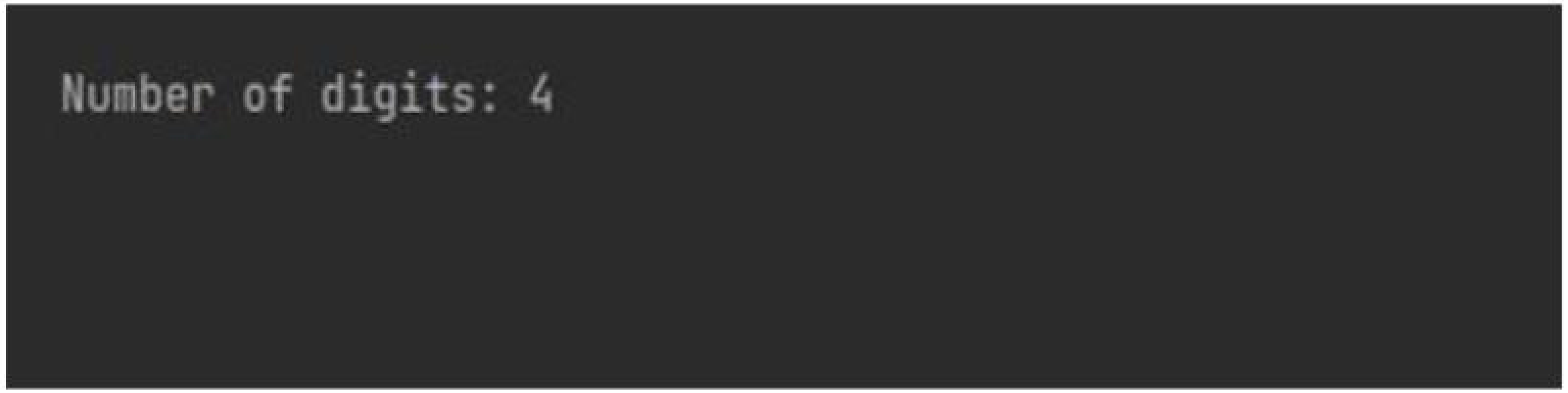
while-Loop


Start
while Loop
Conditional Expression
false
true
Statements
End

while-Loop Demo



```
public static void main(String[] args) {
    int count = 0, num = 3452;
    while (num != 0) {
        num = num/10;
        ++count;
    }
    System.out.println("Number of digits: " + count);
}
```

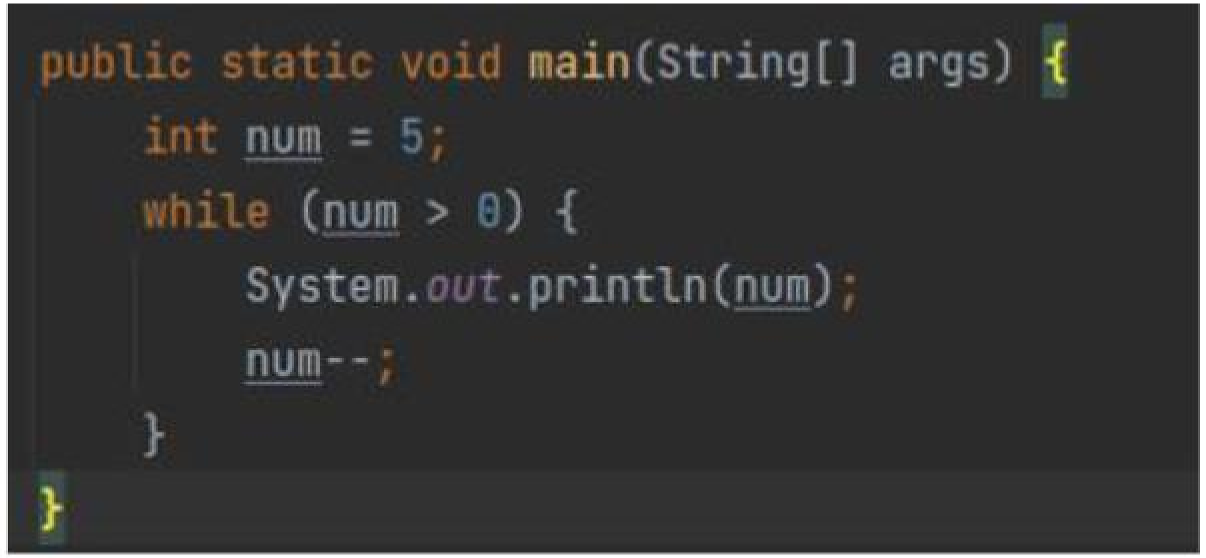


```
Number of digits: 4
```

- The loop prints the number of digits stored in the variable num.
- Inside the while loop condition is checked to determine whether it is not equal to zero by using (!=) operator.
- If the condition becomes true, then num is divided by ten (num/10) and the remainder is stored in num.
- If the condition becomes true, the count is incremented.
- Once the condition becomes false, the final count is printed.

Quick Check

Predict the output for the following code:

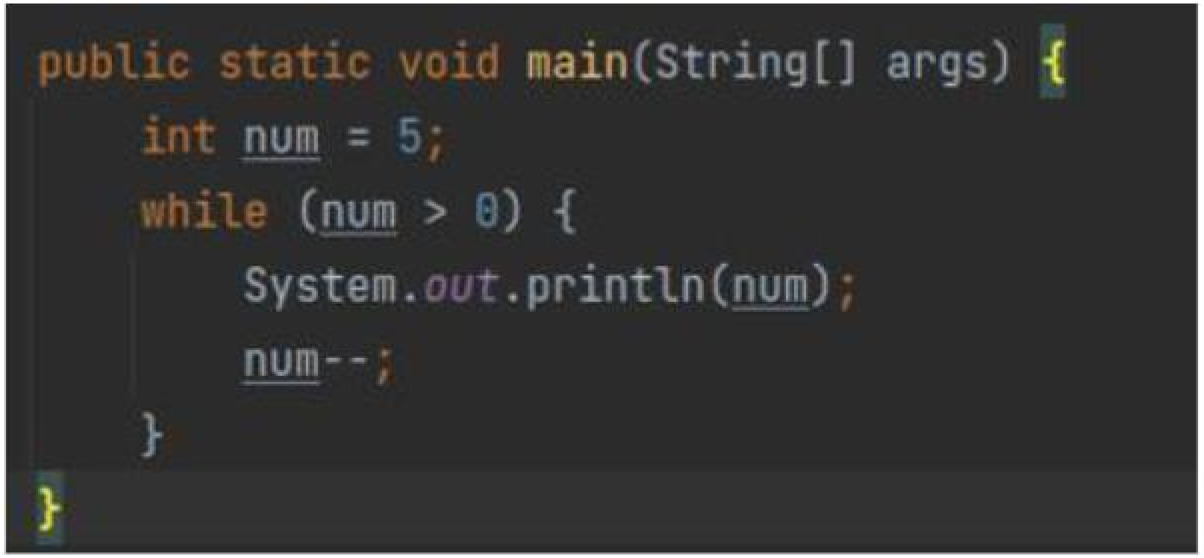


```java
public static void main(String[] args) {
    int num = 5;
    while (num > 0) {
        System.out.println(num);
        num--;
    }
}
```

1. No output
2. 5 4 3 2 1
3. 5
4. Syntax Error

Quick Check: Solution

Predict the output for the following code:



```java
public static void main(String[] args) {
    int num = 5;
    while (num > 0) {
        System.out.println(num);
        num--;
    }
}
```

1. No output
2. **5 4 3 2 1**
3. 5
4. Syntax Error

Add the Even Numbers

Write a program to add the even numbers from the given input number.

Use the scanner to take the input from the user.

DEMO

do-while

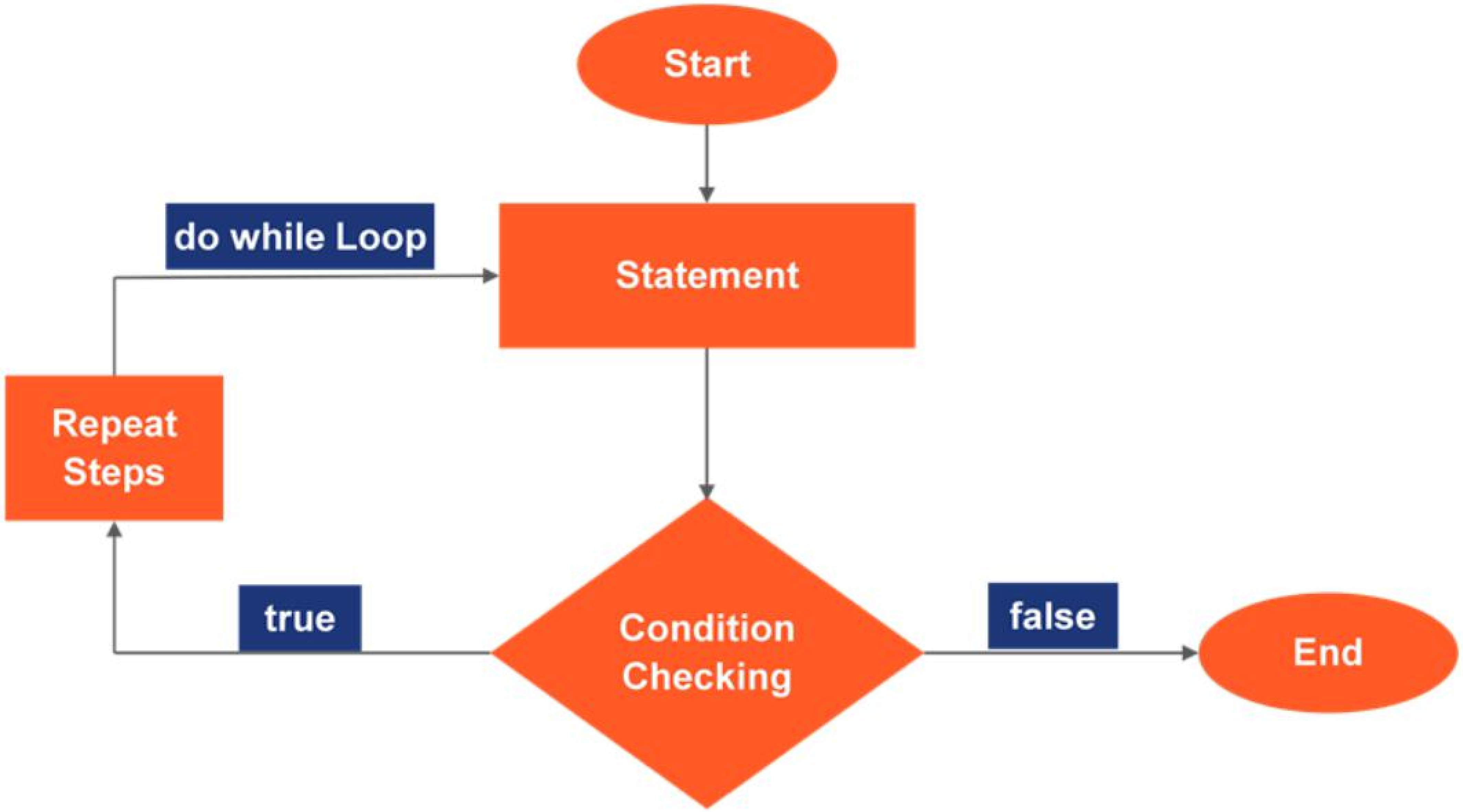
Java Syntax do-while Loop

```
do
{
     Loop Body
}

while
(test_expression);
```

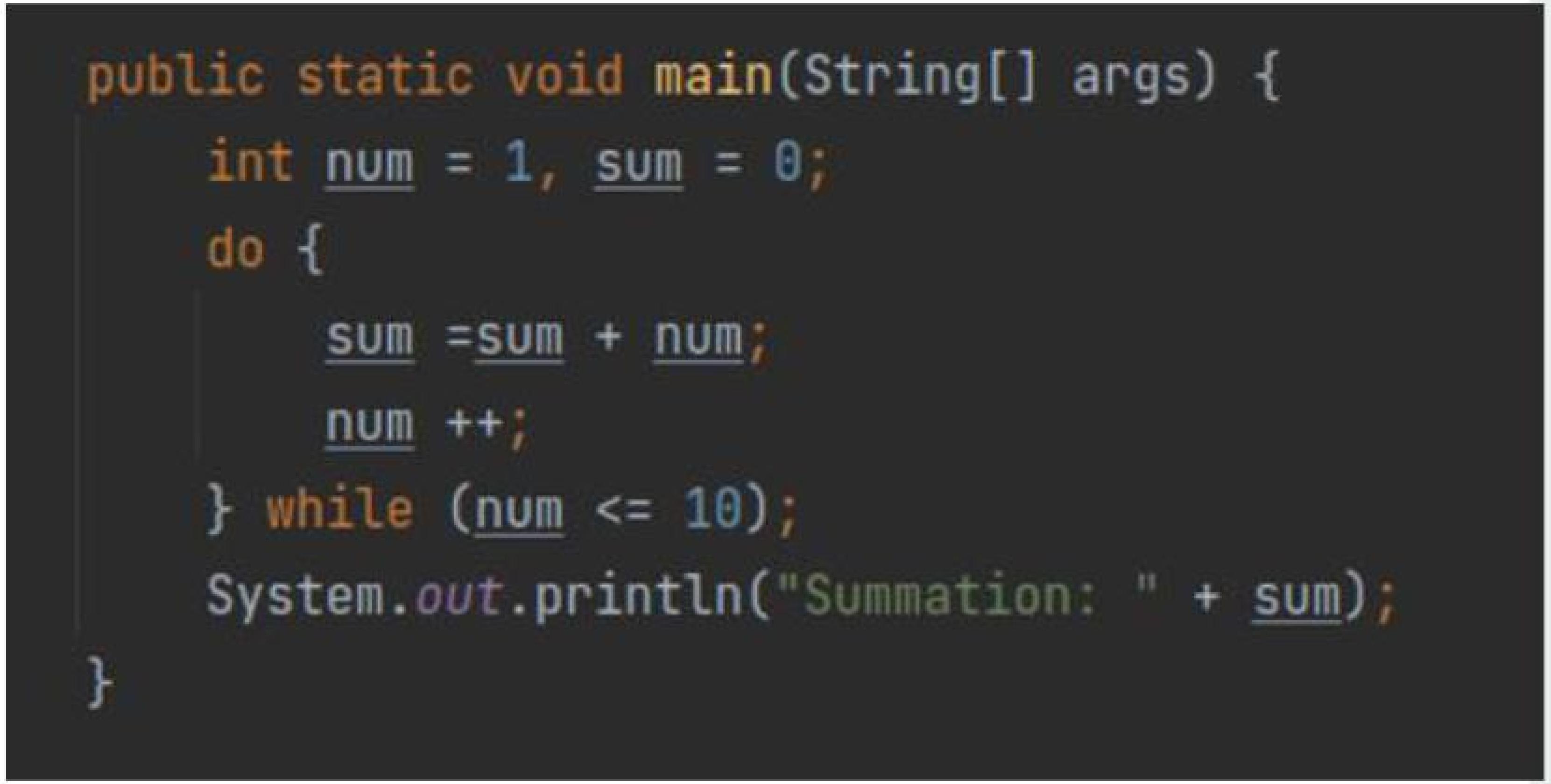
Click to add text

- The do-while loop executes the body once and before any condition is checked.
- **Test Expression:** In this expression, you have to test the condition.
  - If the condition evaluates to be true, then the body of the loop is executed.
  - If the expression is false, the loop will exit.
  - A semicolon is mandatory after the while loop.

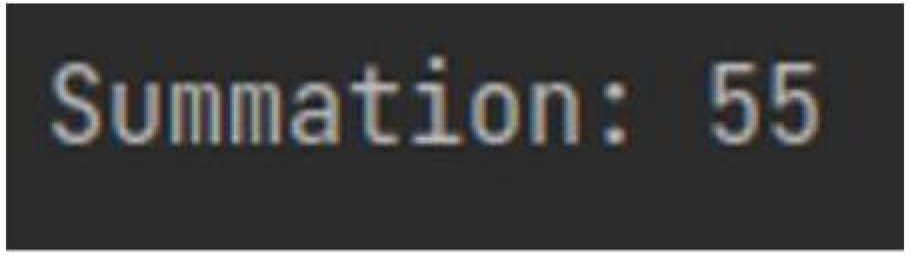
“do-while” Loop



do-while Loop



```
public static void main(String[] args) {
    int num = 1, sum = 0;
    do {
        sum =sum + num;
        num ++;
    } while (num <= 10);
    System.out.println("Summation: " + sum);
}
```

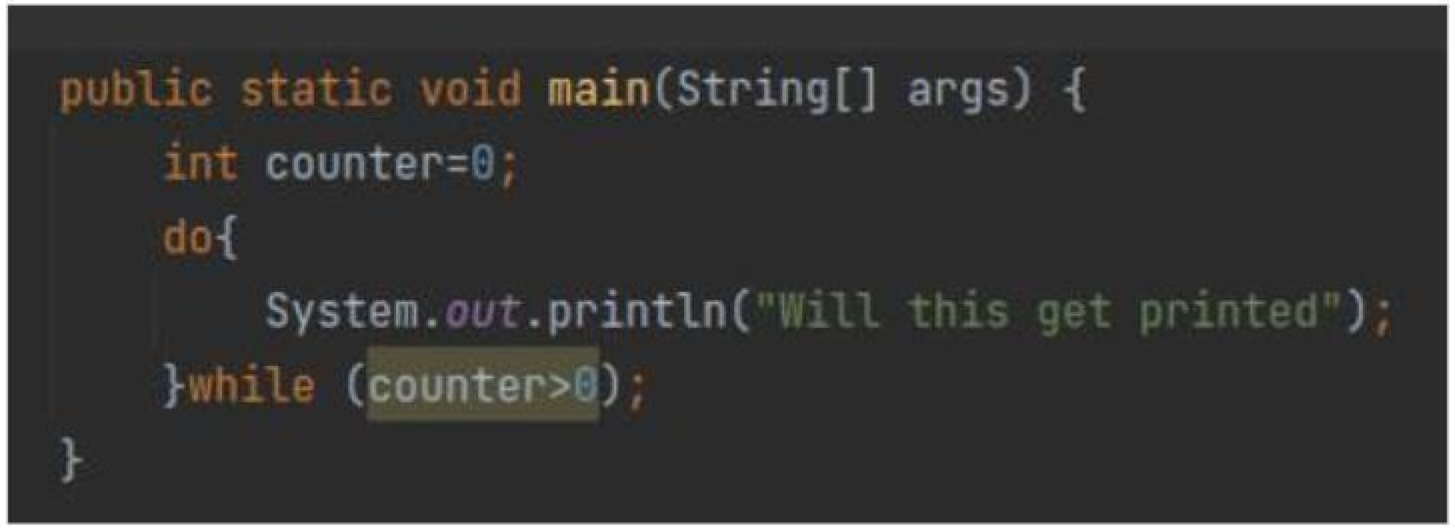


```
Summation: 55
```

- The loop prints the sum of numbers between 1 to 10.
- Inside the do block, the sum of the numbers will be calculated and the num value will be incremented by 1.
- Inside the while, it will check the condition if the num is less than or equal to 10, this means the condition is true .
- Again, it will go inside the loop
- Adding a semi colon (;) at the end of the while loop ends the line and the next lines will be executed.

Quick Check

Predict the output for the following code:

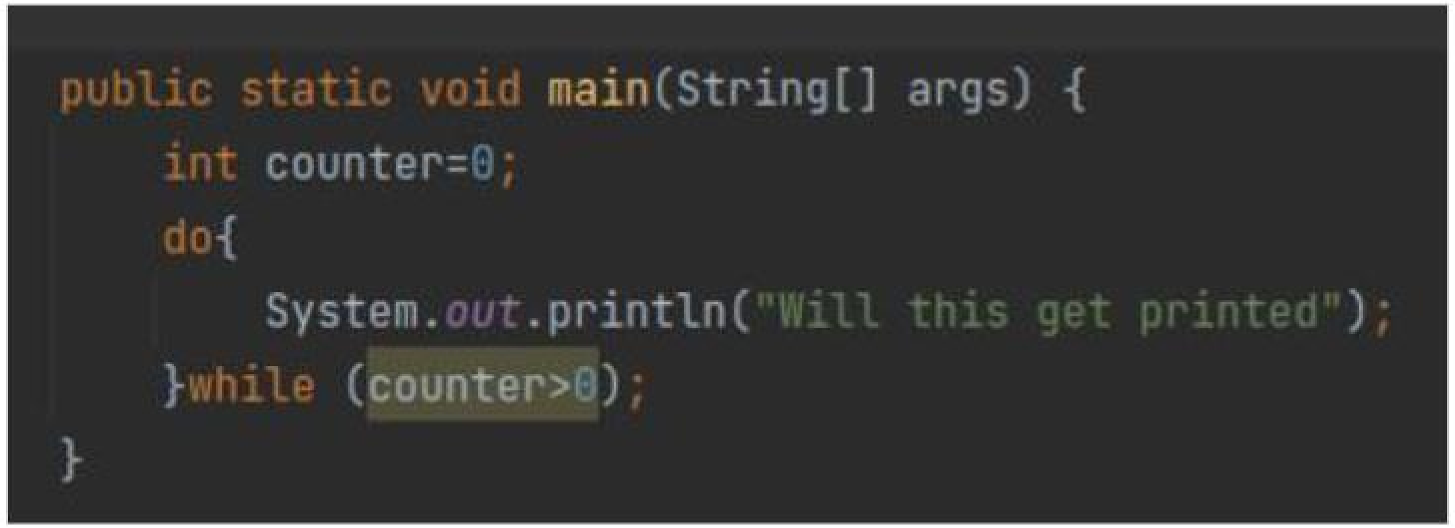


```
public static void main(String[] args) {
    int counter=0;
    do{
        System.out.println("Will this get printed");
    }while (counter>0);
}
```

1. Will this get printed?
2. Syntax Error
3. No output

Quick Check: Solution

Predict the output for the following code:



```java
public static void main(String[] args) {
    int counter=0;
    do{
        System.out.println("Will this get printed");
    }while (counter>0);
}
```

1. **Will this get printed?**
2. Syntax Error
3. No output

for-Loop vs. while-Loop vs. do-while Loop

for-loop	while-loop	do-while loop
If the number of iterations is fixed, it is recommended to use *for* loop.	If the number of iterations is not fixed, it is recommended to use *while* loop.	If the number of iterations is not fixed, and the loop needs to be executed at least once, use the *do-while* loop.
Initialization and updation are a part of the syntax.	Initialization and updation are **not** a part of the syntax.	Initialization and updation are **not** a part of the syntax.

Number Exceeds 999

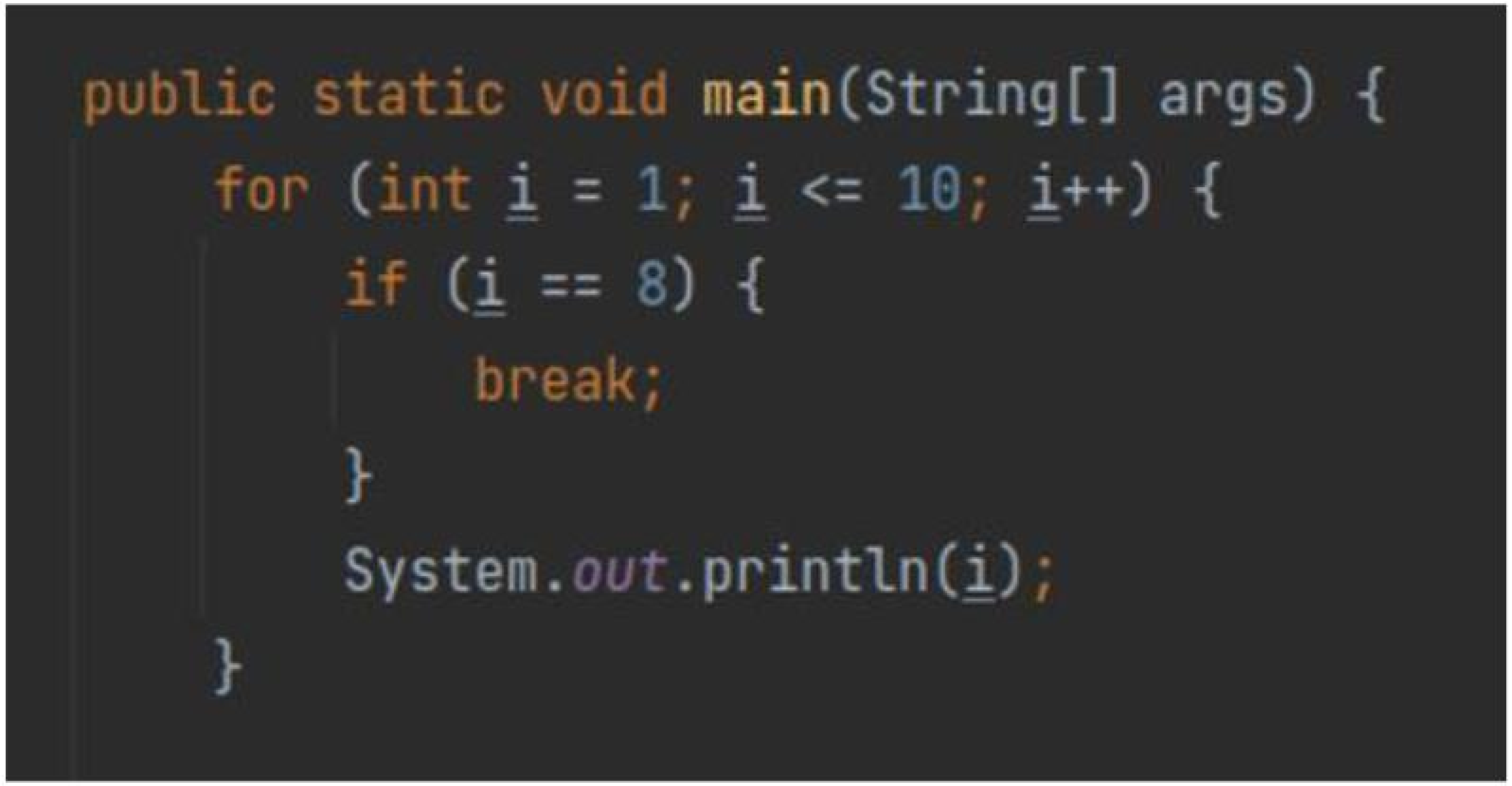
Write a program that accepts a series of numbers from a user and stops by displaying an appropriate message when the sum of the numbers exceeds 999.

DEMO

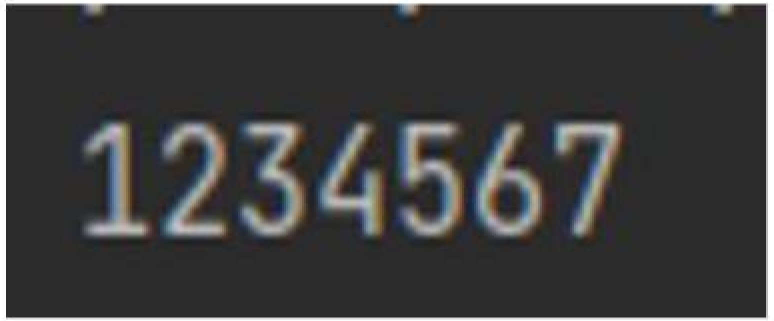
break and continue Statements

- The break and continue statements are used to manage the program flow.
- They can be used in a loop to control loop iterations.
- These statements help the control loop and switch statements by enabling them to break out of the loop or jump to the next iteration by skipping the current loop iteration.
- The break statement:
  - Is a statement used to **break current execution** flow of the program.
    - If break is used **inside a loop,** it will terminate the execution of the loop.
- The continue statement:
  - Breaks one iteration of the loop if a specified condition occurs and continues with the next iteration in the loop.

break Statement



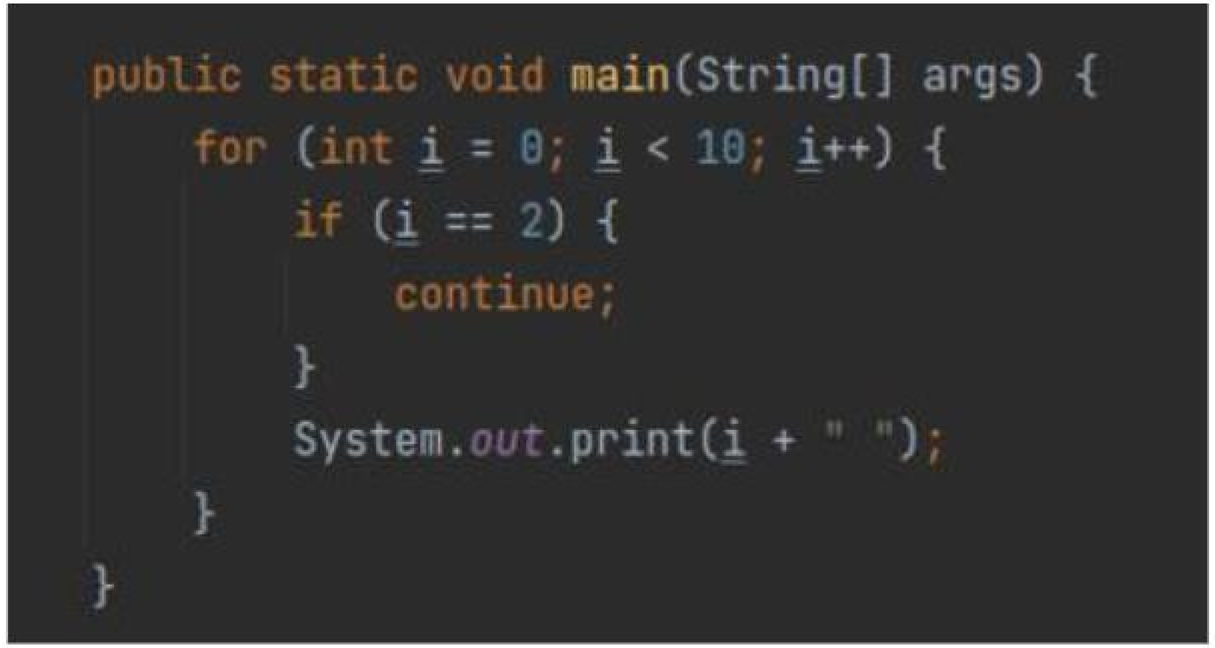
```
public static void main(String[] args) {
    for (int i = 1; i <= 10; i++) {
        if (i == 8) {
            break;
        }
        System.out.println(i);
    }
```



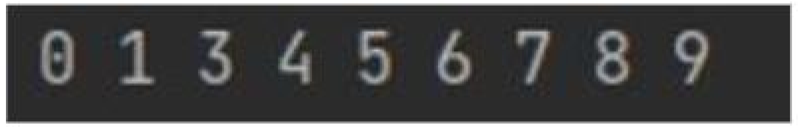
```
1234567
```

- Here the loop will print the value of I from 1 to 7.
- Once the value of I is equal to 8, the condition becomes true, and the break will be called.
- The break will terminate the execution of the loop.

continue Statement



```
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        if (i == 2) {
            continue;
        }
        System.out.print(i + " ");
    }
}
```



```
0 1 3 4 5 6 7 8 9
```

- This demo prints all the numbers between 0 and 9 except 2.
- Here the condition `(i==2)` is being checked.
  - If the condition evaluates to be true, then continue the loop.
  - If the condition is false, then the statement will be printed.
- The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Quick Check

Which of the following statement is true about a break?

1. It stops the execution of the entire program.
2. It halts the execution and forces the control out of the loop.
3. It forces control out of the loop and starts the execution of the next iteration.
4. It halts the execution of the loop for a certain time in seconds.

Quick Check: Solution

Which of the following statement is true about a break?

1. It stops the execution of the entire program.
2. **It halts the execution and forces the control out of the loop.**
3. It forces control out of the loop and starts the execution of the next iteration.
4. It halts the execution of the loop for a certain time in seconds.