Java Course

Table of Content

Java Expression

- Java expression consists of variables, operators, literals and method calls
- Expressions evaluates to a **single** value

```java
// score = 90 is an expression that returns an int
// note that semicolon is not part of the expression
int score;
score = 90;

// 4 + 3 is an expression
double result;
result = 4 + 3;

// number1 == number2 is an expression
if (number1 == number2) {
    System.out.println("Number 1 is equal to Number 2");
}
```

Java Statement

- Statements form a complete unit of execution
- Expressions are part of statements.
- In the example, we have a statement.
- Complete execution of this statement involves multiplying 9 and 5 and then assigning the result to the variable.

```
// statement
int score = 9 * 5;
```

Expression Statements

- We can convert an expression into a statement by terminating the expression with a semicolon -> this results in an Expression-Statement

```
int number;
// expression
number = 10

// statement
number = 10;
```

Declaration Statement

```
int number = 10;
```

Java Blocks

- A block is a group of statements (zero or more) that is enclosed in curly braces

```java
class Main {
    public static void main(String[] args) {

        int myAge = 23;

        if (myAge >= 18) {  // start of block
            System.out.println("Wohoo!");
            System.out.println("I can drink!");
        } // end of block
    }
}
```

- Let's look at the if { .. } block
- Inside the block we have 2 statements:
  - System.out.println("Wohoo!");
  - System.out.println("I can drink!");
- …but, we can have 0 statements inside the block as well

```java
class Main {
    public static void main(String[] args) {

        int myAge = 23;

        if (myAge >= 18) {  // start of block
            System.out.println("Wohoo!");
            System.out.println("I can drink!");
        } // end of block
    }
}
```

```java
class Main {
    public static void main(String[] args) {    // start of block

    }   // end of block
}
```

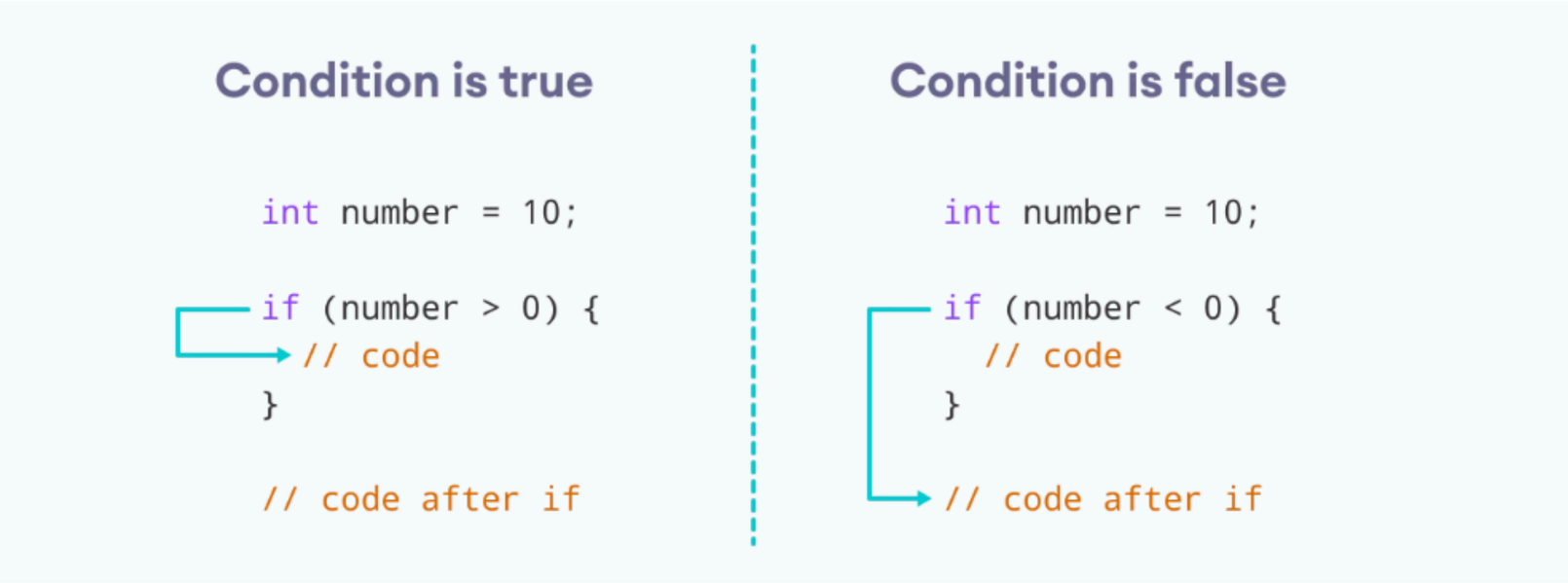
```java
class Main {
    public static void main(String[] args) {
        if (10 > 5) {       // start of block

        }                   // end of block
    }
}
```

`if` statement

if-then

```
if (condition) {
    // statements
}
```

- In programming, `If` statements are used to run a block of code among more than one alternatives.
- Simple if-then in Java



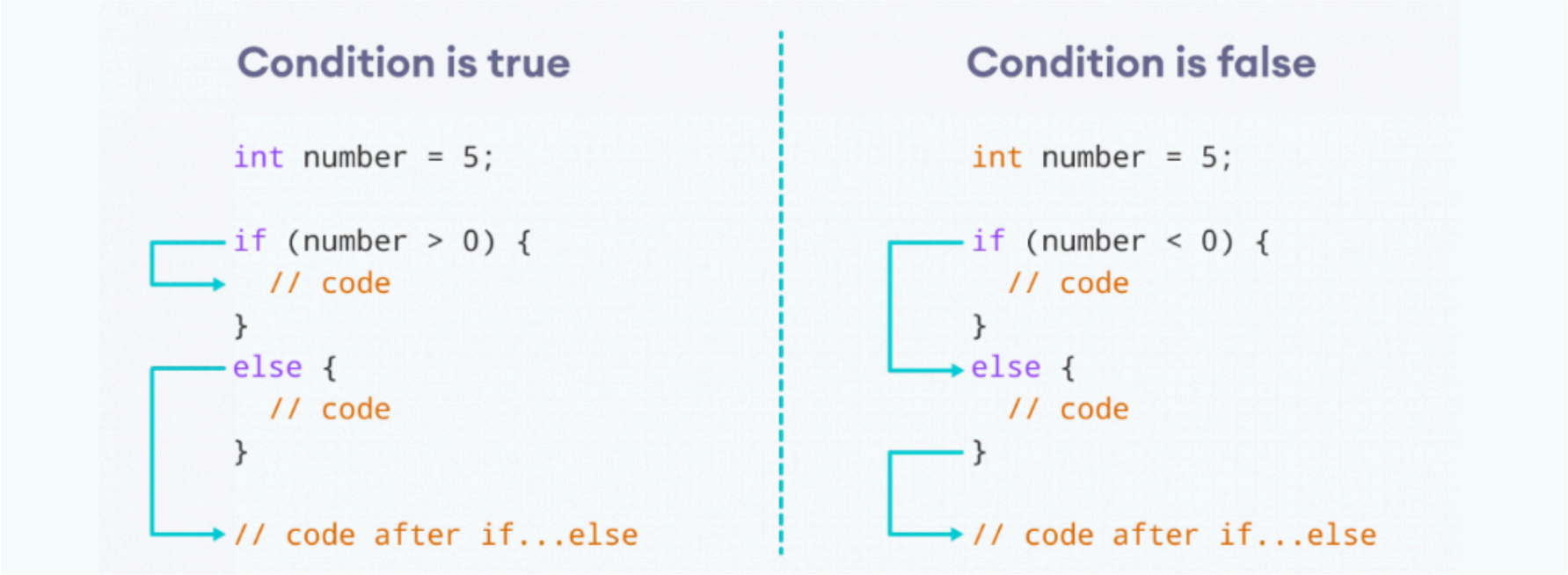
```java
class Main {
    public static void main(String[] args) {
        int number = 10;

        // checks if number is less than 0
        if (number < 0) {
          System.out.println("The number is negative.");
        }

        System.out.println("Statement outside if block");
    }
}
```

`if...else` statement

if-then-else

```
if (condition) {
    // code in if block
} else {
    // code in else block
}
```

- if-then flow:



```java
class Main {
    public static void main(String[] args) {
        int number = -10;

        // checks if number is positive
        if (number > 0) {
            // if condition is satisfied
            System.out.println("The number is positive.");
        } else {
            // if condition is not satisfied
            System.out.println("The number is NOT positive.");
        }

        System.out.println("Statement outside if..else block");
    }
}
```

if…else if…else

```
if (condition1) {
    // code if condition1 is satisfied
} else if (condition2) {
    // code if condition2 is satisfied
} else if (condition3) {
    // code if condition3 is satisfied
}
    .
    .
    .
else {
    // code if no condition is satisfied
}
```

1st Condition is true

```
int number = 2;
if (number > 0) {
  // code
}
else if (number == 0){
  // code
}
else {
  //code
}

//code after if
```

2nd Condition is true

```
int number = 0;
if (number > 0) {
  // code
}
else if (number == 0){
  // code
}
else {
  //code
}

//code after if
```

All Conditions are false

```
int number = -2;
if (number > 0) {
  // code
}
else if (number == 0){
  // code
}
else {
  //code
}

//code after if
```

```java
class Main {
    public static void main(String[] args) {
        int number = 0;

        // checks if number is greater than 0
        if (number > 0) {
          System.out.println("The number is positive.");
        }

        // checks if number is less than 0
        else if (number < 0) {
          System.out.println("The number is negative.");
        }

        // if both condition is false
        else {
          System.out.println("The number is 0.");
        }
    }
}
```

`switch-case` statement

- The expression is evaluated once and compared with the values of each `case`.
- If expression matches with any value, it will execute the code of that `case`.
- If there is no match - code in the `default` case will be executed

```
switch(expression) {
    case value1:
        // code
    case value2:
        // code
      .
      .
      .
    default:
        // code
}
```

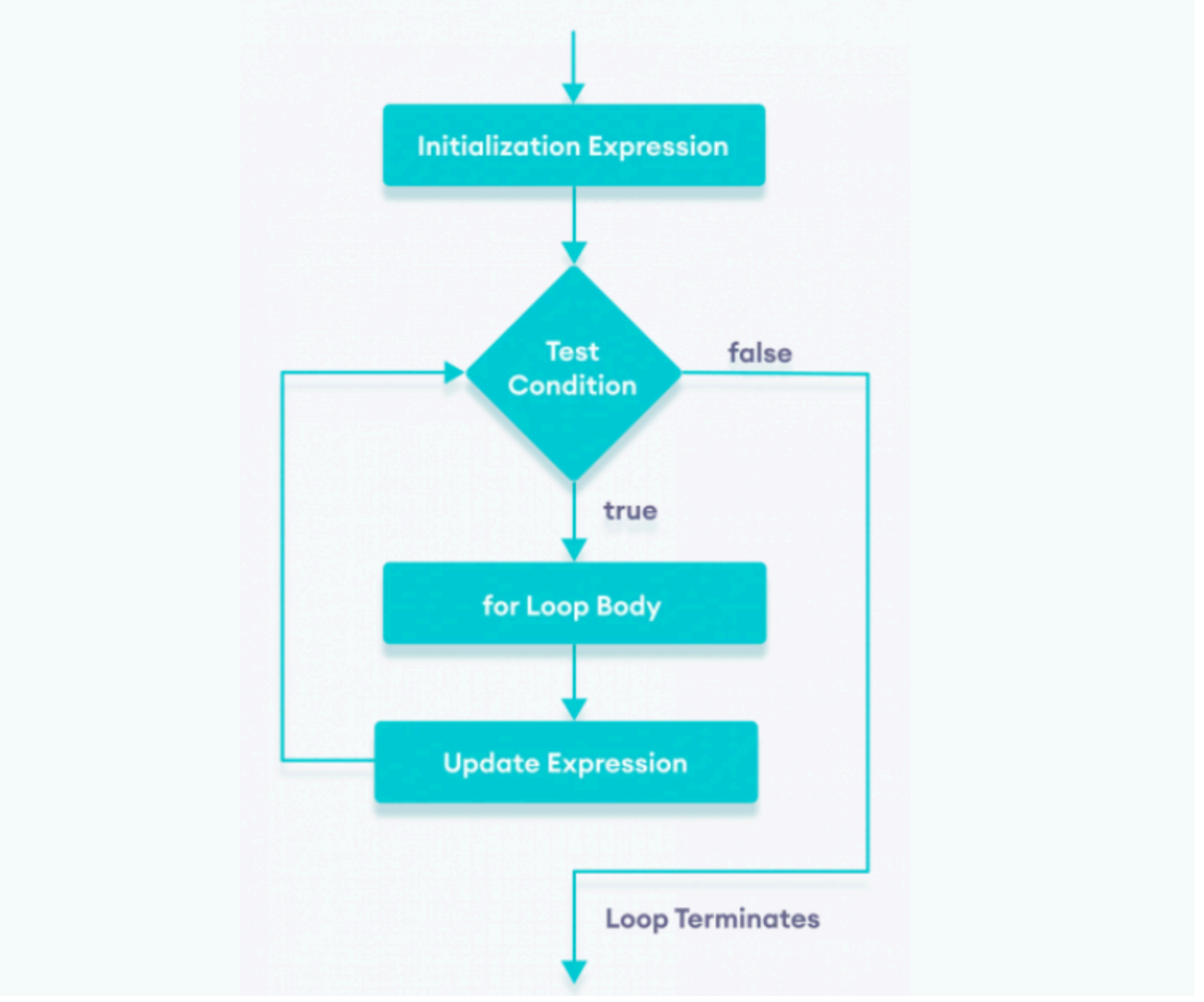
```java
class Main {
    public static void main(String[] args) {
        int number = 44;
        String size;

        // switch statement to check size
        switch (number) {

          case 29:
            size = "Small";
            break;

          case 42:
            size = "Medium";
            break;

          // value matched
          case 44:
            size = "Large";
            break;

          case 48:
            size = "Extra Large";
            break;

          default:
            size = "Unknown";
            break;

        }
        System.out.println("Size: " + size);
    }
}
```

`for` Loop

- In programming, loops are used to repeat a block of code
- DRY (**D**on’t **R**epeat **Y**ourself)
- `for` loop is used to run a block of code for a certain number of times

```
for (initialExpression; testExpression; updateExpression) {
    // body of the loop
}
```

1. The `initialExpression` initialises and/or declares variables and executes only once
2. The `condition` is **evaluated**
3. If the `condition` is **`true`**, the body of the `for` loop is executed
4. The `updateExpression` updates the value of `initialExpression`
5. The `condition` is evaluated again
6. The process continues until the `condition` is **`false`**


Initialization Expression
Test Condition
false
true
for Loop Body
Update Expression
Loop Terminates

```java
class Main {
    public static void main(String[] args) {

        int n = 5;
        // for loop
        for (int i = 1; i <= n; i++) {
            System.out.println(i);
        }

    }
}
```

Output

```
1
2
3
4
5
```

Iteration	Variable	Condition: i <= n	Action
1st	`i = 1` `n = 5`	`true`	`1` is printed. `i` is increased to **2**.
2nd	`i = 2` `n = 5`	`true`	`2` is printed. `i` is increased to **3**.
3rd	`i = 3` `n = 5`	`true`	`3` is printed. `i` is increased to **4**.
4th	`i = 4` `n = 5`	`true`	`4` is printed. `i` is increased to **5**.
5th	`i = 5` `n = 5`	`true`	`5` is printed. `i` is increased to **6**.
6th	`i = 6` `n = 5`	`false`	The loop is terminated.

for-each Loop

- This type of loop is used for iterating over arrays or collections

```java
class Main {
    public static void main(String[] args) {

        // create an array
        int[] numbers = {3, 7, 5, -5};

        // iterating through the array
        for (int number : numbers) {
            System.out.println(number);
        }

    }
}
```

Output

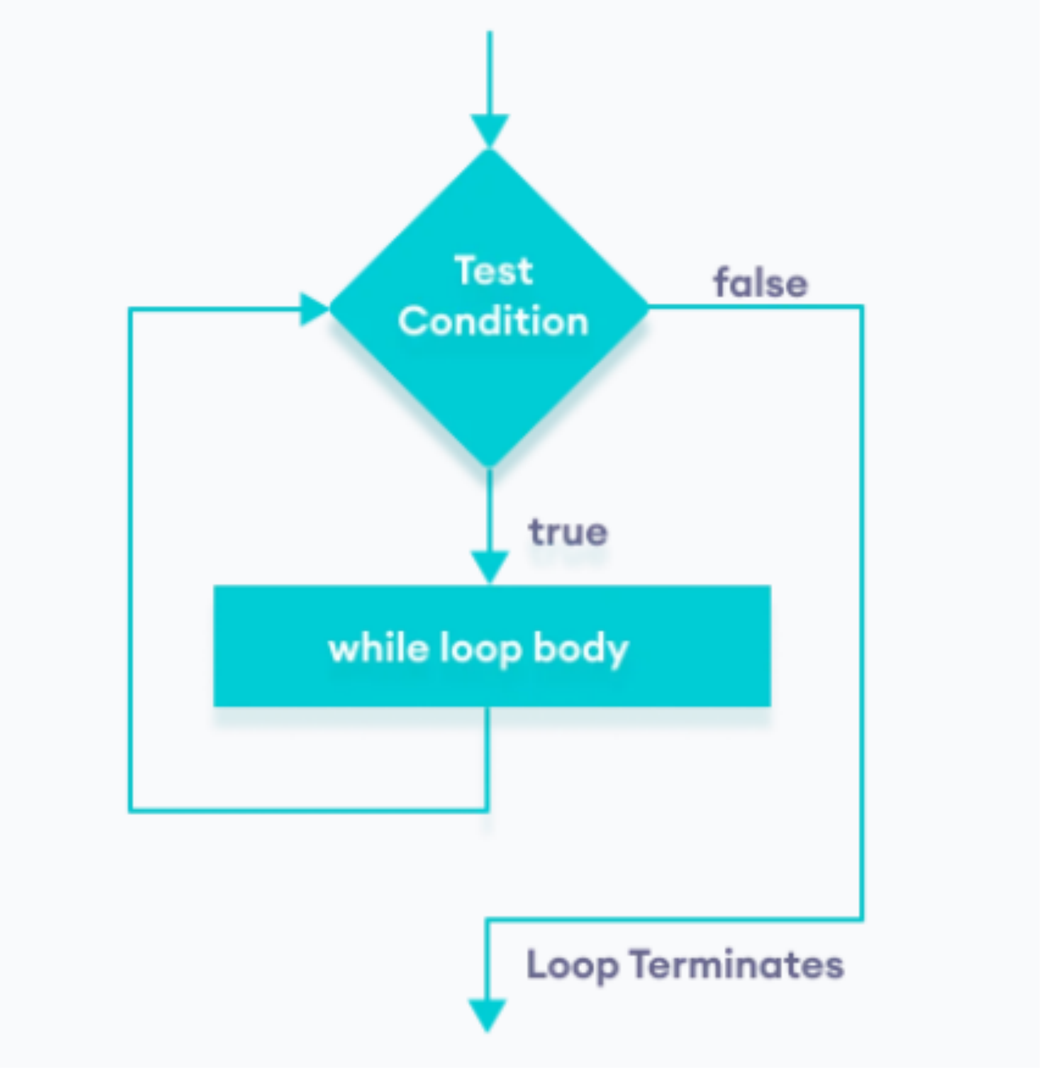
```
3
7
5
-5
```

`while` Loop

- Used to run a block of code until a certain condition is met

```
while (testExpression) {
    // body of loop
}
```

1. Evaluate testExpression
2. If `testExpression` evaluates to **`true`**, the code inside a body of loop is executed
3. The `testExpression` is evaluated again
4. This process continues until the `testExpression` evaluates to **`false`**
5. When the `testExpression` evaluates to **`false`**, the loop stops


Test
Condition
false
true
while loop body
Loop Terminates

```java
class Main {
    public static void main(String[] args) {

        // declare variables
        int i = 1, n = 5;

        // while loop from 1 to 5
        while(i <= n) {
            System.out.println(i);
            i++;
        }

    }
}
```

```
1
2
3
4
5
```

Iteration	Variable	Condition: i <= n	Action
1st	`i = 1` `n = 5`	`true`	`1` is printed. `i` is increased to **2**.
2nd	`i = 2` `n = 5`	`true`	`2` is printed. `i` is increased to **3**.
3rd	`i = 3` `n = 5`	`true`	`3` is printed. `i` is increased to **4**.
4th	`i = 4` `n = 5`	`true`	`4` is printed. `i` is increased to **5**.
5th	`i = 5` `n = 5`	`true`	`5` is printed. `i` is increased to **6**.
6th	`i = 6` `n = 5`	`false`	The loop is terminated

Infinite Loops

- We have to be careful when writing loop conditions
- Infinite loop happens when condition always evaluates to **`true`**

```java
// infinite for loop
for (int i = 10; i < 5; i++) {
    System.out.println("Hello");
}
```

```java
while (true) {
    // body of loop
}
```

`break` statement

- `break` statement is used for terminating a loop
- In nested loops, it breaks only it's inner loop

```java
class Main {
    public static void main(String[] args) {

        for (int i = 0; i < 10; i++) {
            System.out.println("i: " + i);
            if (i == 5) {
                break;
            }
        }
    }
}
```

```java
class Main {
    public static void main(String[] args) {

        for (int i = 0; i < 10; i++) {
            System.out.println("i: " + i);
            for (int j = 0; j < 10; j++) {
                System.out.println("j: " + j);
                if (j == 5) {
                    break;
                }
            }
        }
    }
}
```

`continue` statement

- `continue` statement is used to jump to the next iteration of the loop

```
class Main {
    public static void main(String[] args) {

        for (int i = 0; i < 10; i++) {
            if (i == 5) {
                continue;
            }
            // this doesn't get executed when i == 5
            System.out.println("i: " + i);
        }
    }
}
```

Exercise

- https://www.hackerrank.com/challenges/java-if-else/problem
- https://www.hackerrank.com/challenges/java-loops-i/problem