

JAVAGURU INTRODUCTION TO JAVA

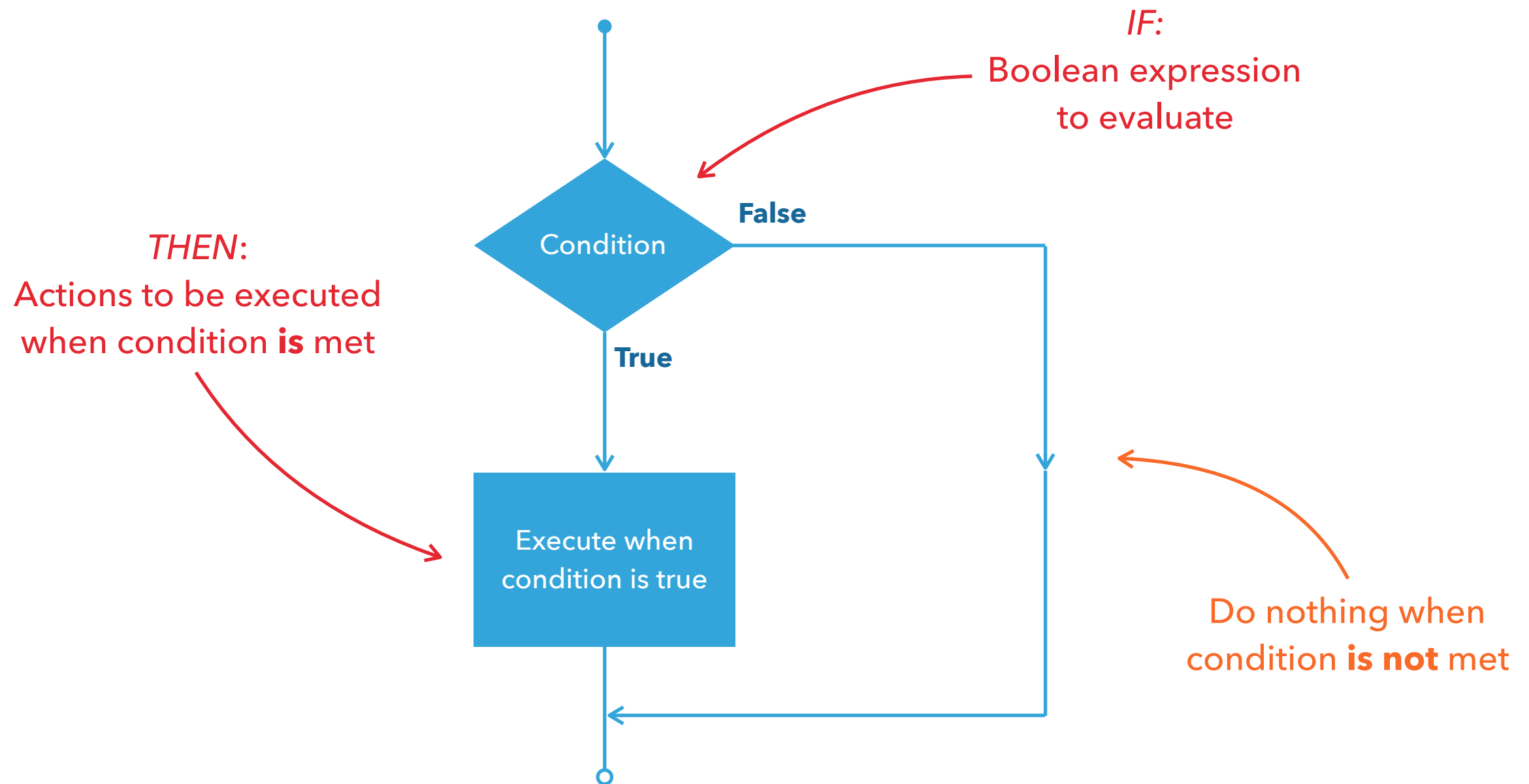
LESSON 4

CONDITIONAL FLOW CONTROL

CONDITIONAL STATEMENTS

- ▶ **Control** code execution by specifying **certain conditions**
 - ▶ When conditional statement is **met** (equals to '**true**')
 - ▶ When conditional statement is **not met** (equals to '**false**')
 - ▶ There are **two** main conditional statements:
 - ▶ **If** statement
 - ▶ **Switch** statement

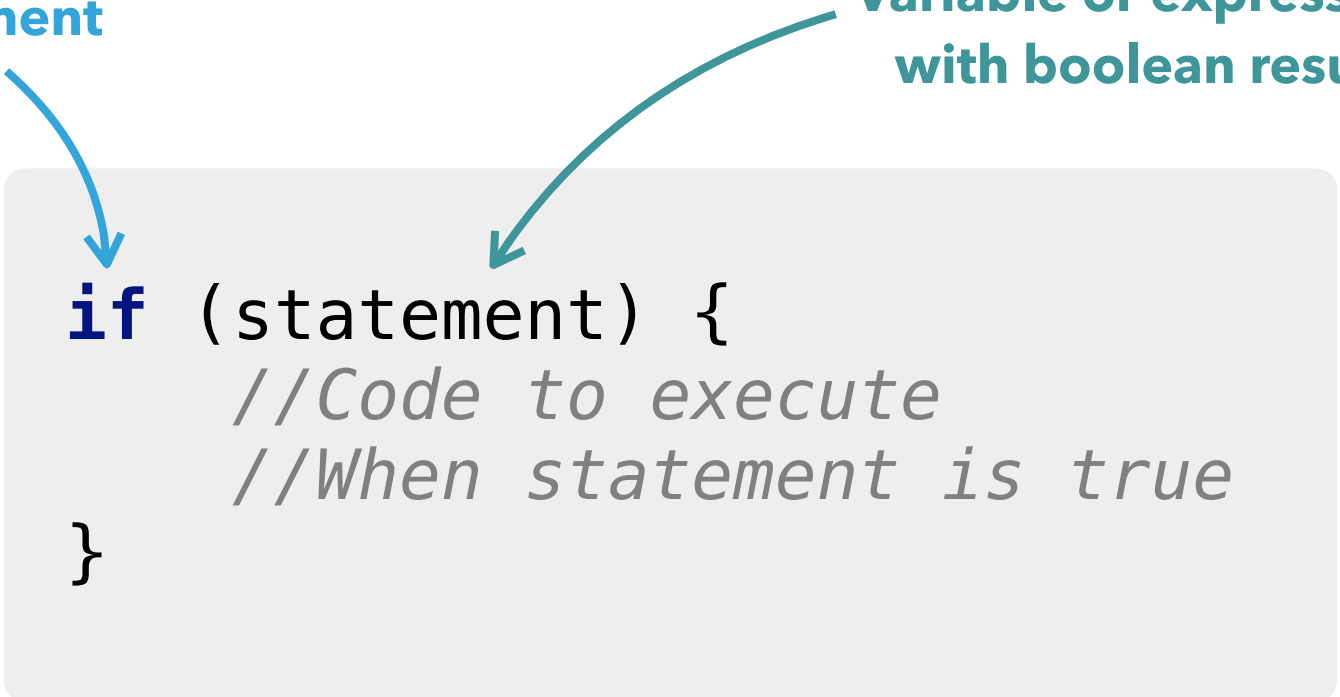
DECISION MAKING FLOWCHART: IF



IF STATEMENT: SYNTAX

Keyword specifying
conditional statement

Variable or expression
with boolean result



```
if (statement) {  
    //Code to execute  
    //When statement is true  
}
```

IF STATEMENT: EXAMPLE

Boolean variable expression

```
boolean flag = true;

if (flag) {
    System.out.print("True");
}
```

Inline expression

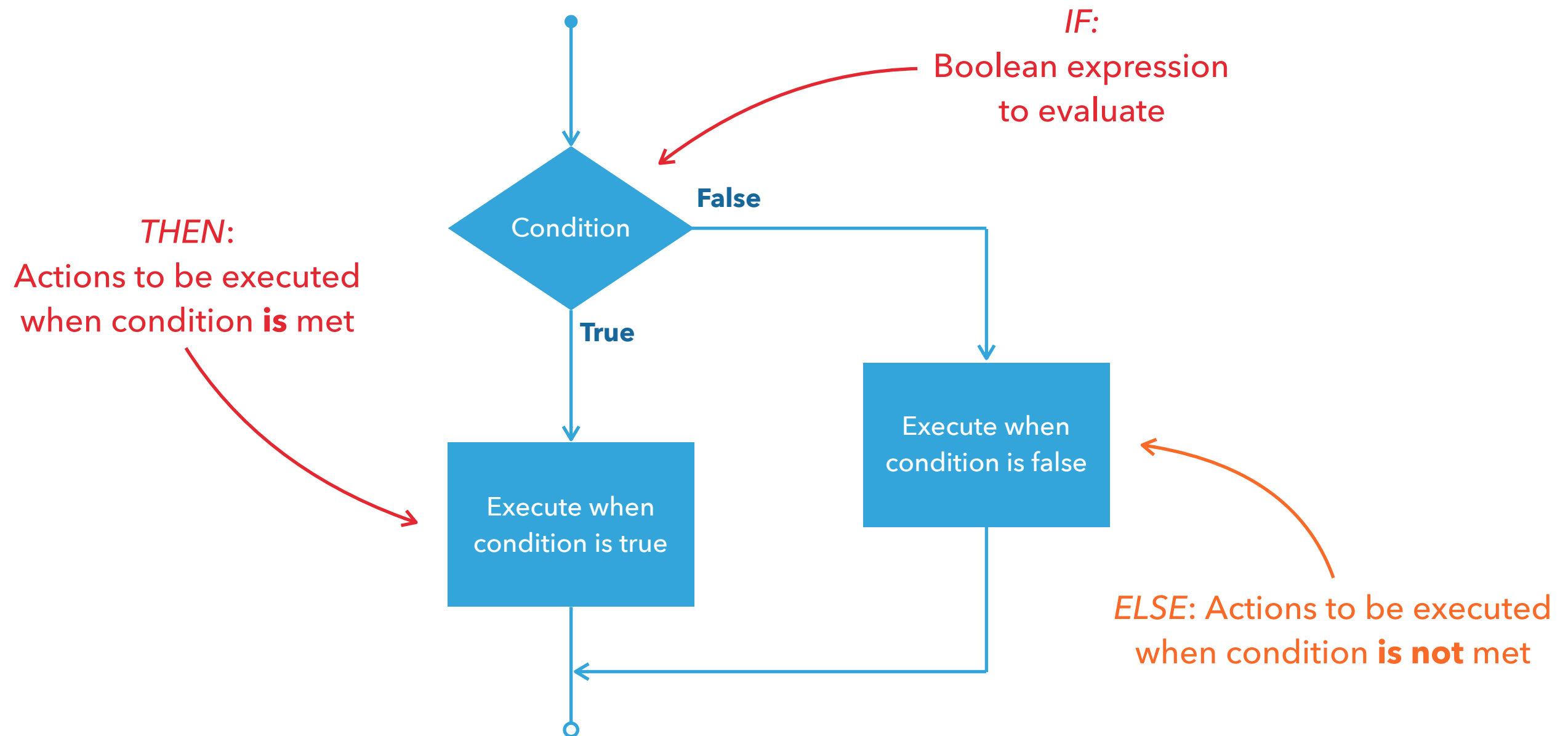
```
int x = 5;

if (x > 10) {
    System.out.print("x > 10");
}
```

IF STATEMENT RULES RECAP

- ▶ Consists of a **boolean expression** followed by **one or more** statements
- ▶ Boolean expression can be **composed** of multiple subexpressions

DECISION MAKING FLOWCHART: IF - ELSE



IF – ELSE STATEMENT: SYNTAX

Keyword specifying
conditional statement

Variable or expression
with boolean result

```
if (statement) {  
    //Code to execute  
    //When statement is true  
}  
else {  
    //Code to execute  
    //When statement is false  
}
```

Keyword specifying
alternative code block

IF – ELSE STATEMENT: EXAMPLE

Boolean variable expression

```
boolean flag = false;

if (flag) {
    System.out.print("True");
} else {
    System.out.print("False");
}
```

Inline expression

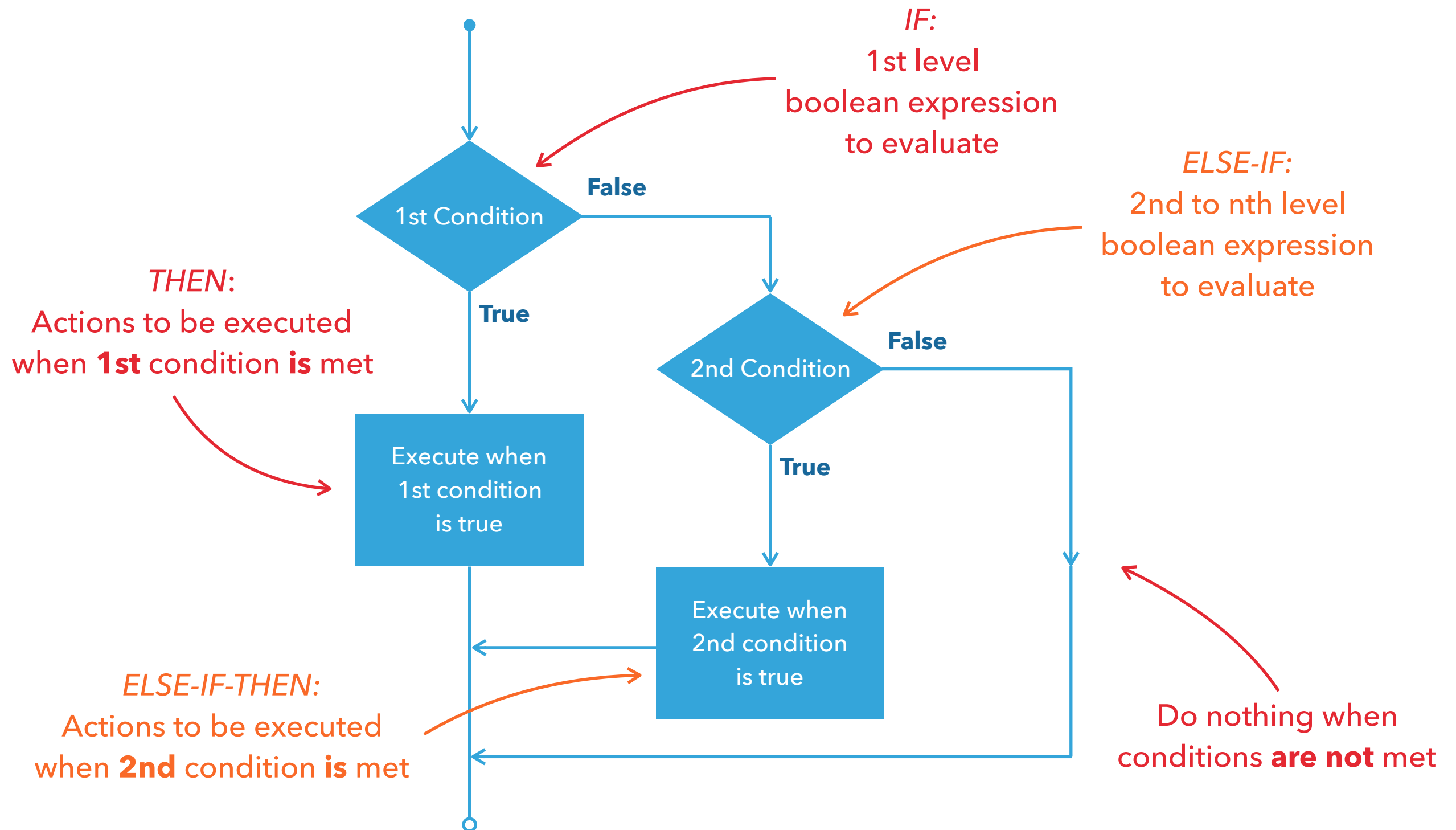
```
int x = 5;

if (x > 10) {
    System.out.print("x > 10");
} else {
    System.out.print("x <= 10");
}
```

IF – ELSE STATEMENT RULES RECAP

- ▶ If statement can be followed by an **optional** else statement, which executes when the boolean expression is **false**

DECISION MAKING FLOWCHART: IF – ELSE IF



IF – ELSE IF STATEMENT: SYNTAX

Keyword specifying
conditional statement

Variable or expression
with boolean result

```
if (statement1) {  
    //Code to execute  
    //When statement1 is true  
}  
else if (statement2) {  
    //Code to execute  
    //When statement2 is true  
}
```

Keyword specifying
alternative conditional
code block

IF – ELSE IF STATEMENT: EXAMPLE

Boolean variable expression

```
boolean flag1 = false;
boolean flag2 = true;

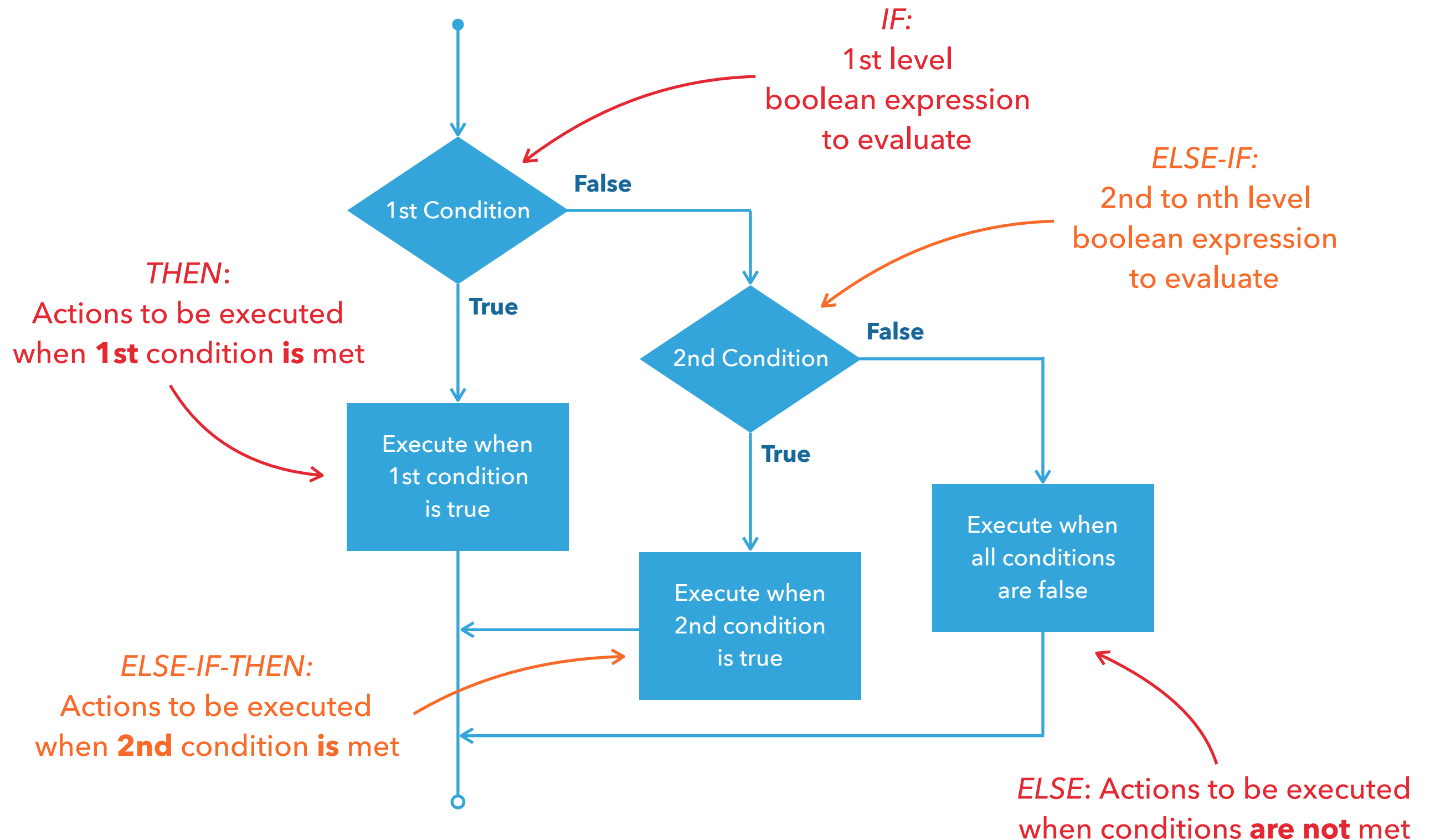
if (flag1) {
    System.out.print("flag1");
} else if (flag2) {
    System.out.print("flag2");
}
```

Inline expression

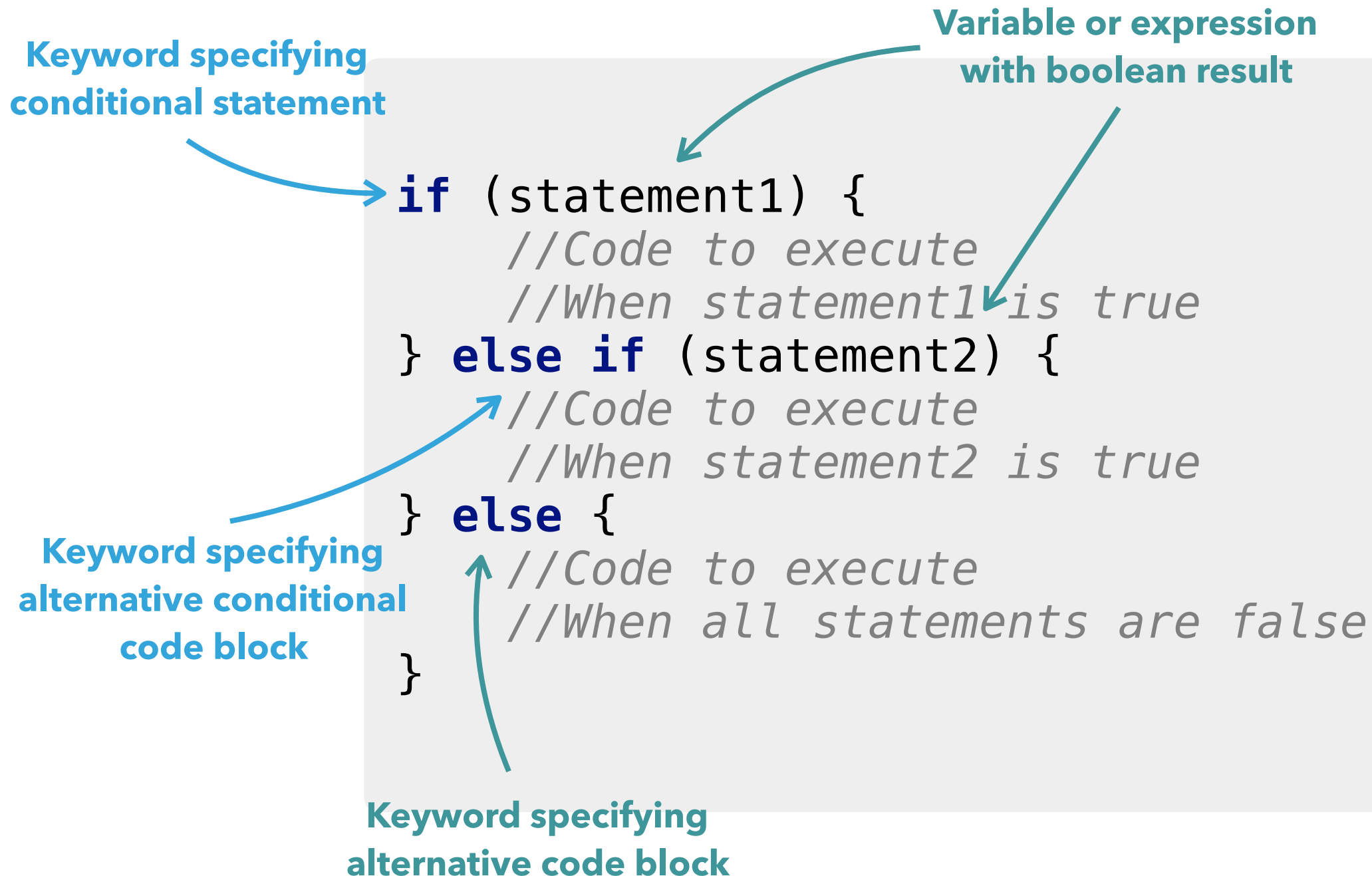
```
int x = 7;

if (x == 3) {
    System.out.print("x == 3");
} else if (x == 7) {
    System.out.print("x == 7");
}
```

DECISION MAKING FLOWCHART: IF - ELSE IF - ELSE



IF – ELSE IF – ELSE STATEMENT: SYNTAX



IF – ELSE IF – ELSE STATEMENT: EXAMPLE

Boolean variable expression

```
boolean flag1 = false;
boolean flag2 = false;

if (flag1) {
    System.out.print("flag1");
} else if (flag2) {
    System.out.print("flag2");
} else {
    System.out.println("none");
}
```

Inline expression

```
int x = 7;

if (x == 3) {
    System.out.print("x == 3");
} else if (x == 7) {
    System.out.print("x == 7");
} else {
    System.out.print("NOTA");
}
```

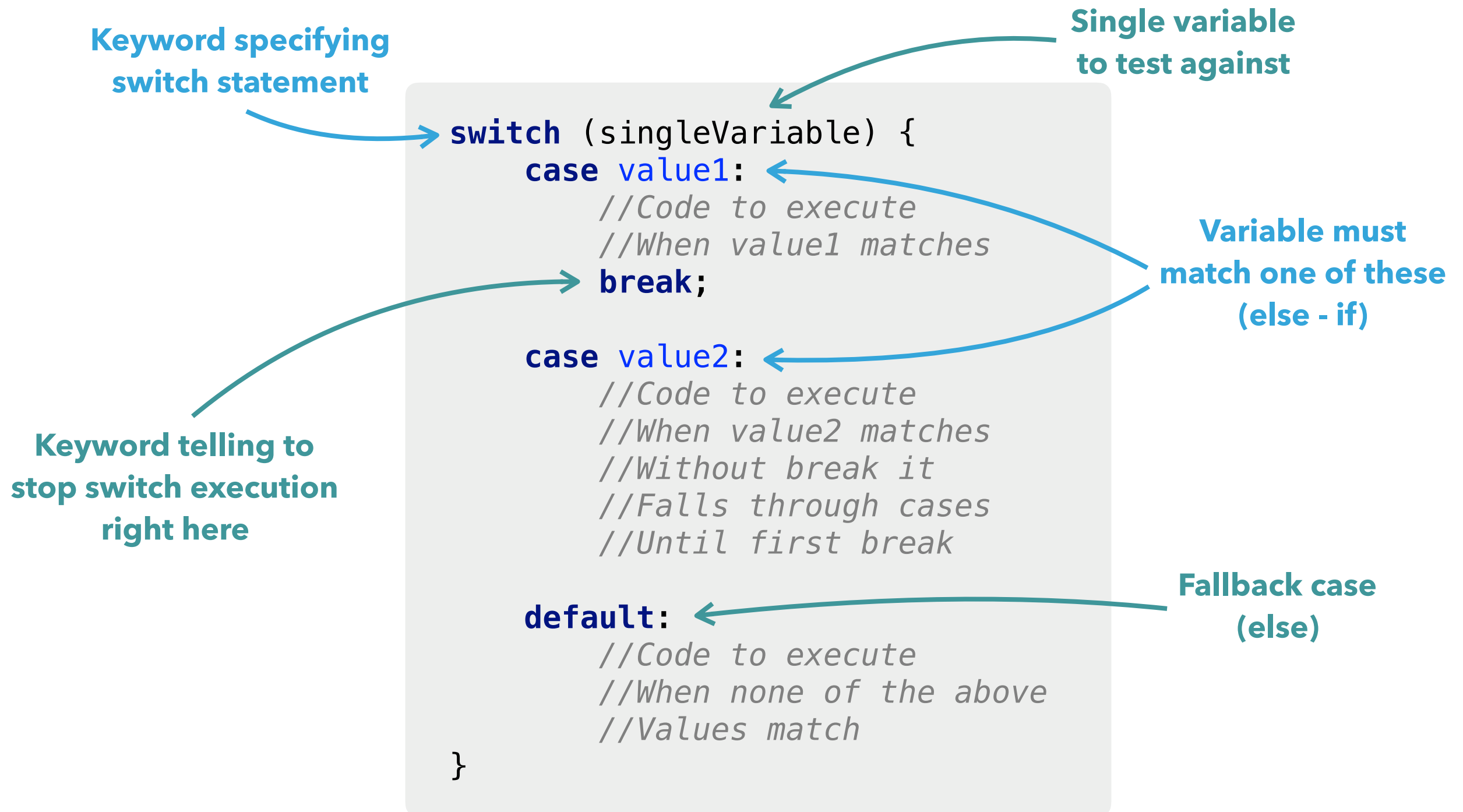
IF – ELSE IF – ELSE STATEMENT RULES RECAP

- ▶ An if can have **zero** or **one** else's and its must come after any else if's
- ▶ An if can have **zero** to **many** else if's and they must come **before** else
- ▶ Once an else if **succeeds**, **none** of the **remaining** else if's or else's will be tested

SWITCH STATEMENT OVERVIEW

- ▶ Provides an **effective** way to deal with a section of code that could branch in **multiple directions** based on **single variable**
- ▶ **Doesn't** support the conditional operators that the **if statement** does
- ▶ **Can't** handle **multiple** variables

SWITCH STATEMENT: SYNTAX



SWITCH STATEMENT: EXAMPLE

```
String drink = "coffee";

switch (drink) {
    case "coffee":
        System.out.println("I would go for Java!");
        break;

    case "tea":
        System.out.println("Everything but Lipton");
        break;

    default:
        System.out.println("Ugh.. What?");
}
```

BUILDING BOOLEAN EXPRESSIONS

THE EQUALITY AND RELATIONAL OPERATORS

Operator	Operation
==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

CONDITIONAL OPERATORS

Operator	Operation
&&	Conditional AND
	Conditional OR
!	Conditional NOT

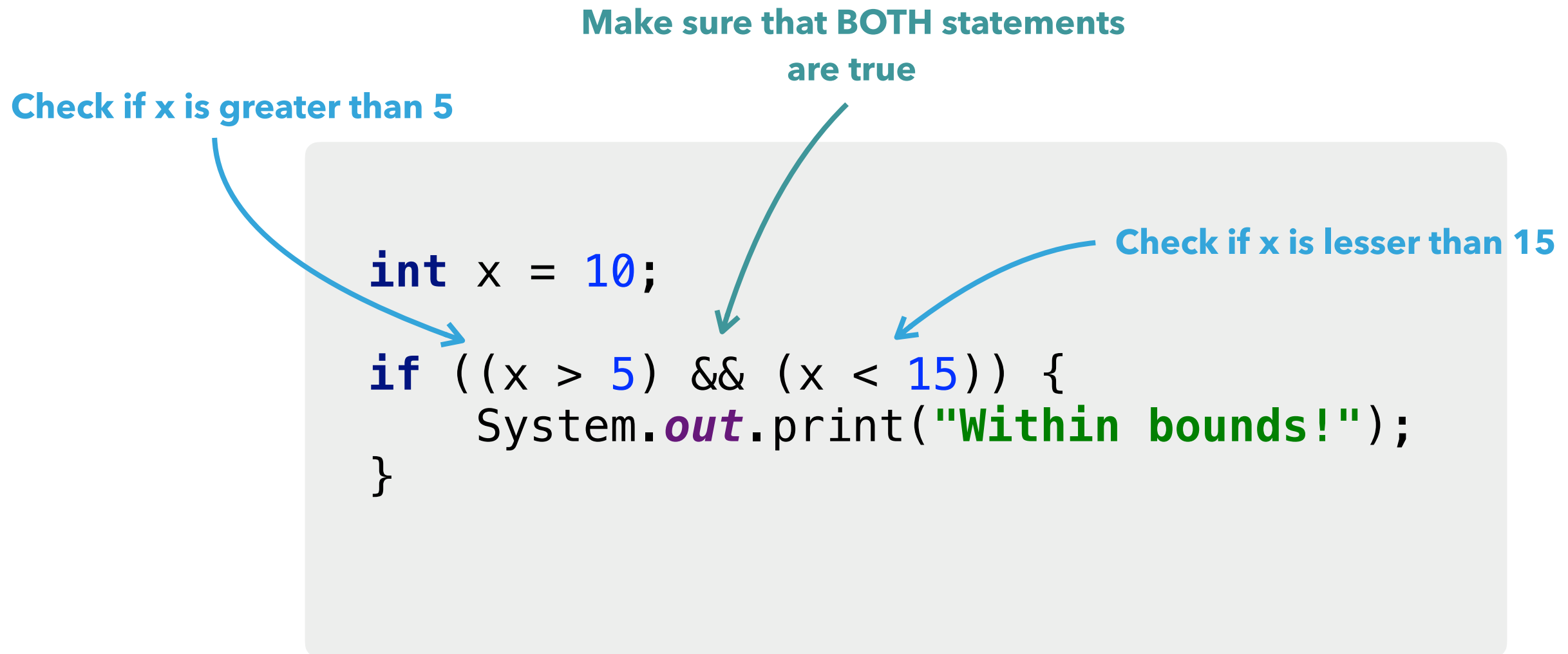
COMPLEX BOOLEAN STATEMENT EXAMPLE

Make sure that **BOTH** statements are true

Check if x is greater than 5

Check if x is lesser than 15

```
int x = 10;  
if ((x > 5) && (x < 15)) {  
    System.out.print("Within bounds!");  
}
```



BASIC CODE TESTING APPROACH

TASK OBJECTIVES

1. Write class that returns **max number** from two given numbers
2. Write test scenarios to **verify** method works as **expected**
3. **Run** test scenarios

1. WRITE CLASS SOLVING GIVEN PROBLEM

```
public class QuickMaths {  
    public int max(int a, int b) {  
        if (a > b) {  
            return a;  
        } else {  
            return b;  
        }  
    }  
}
```

2.A. WRITE TEST CLASS WITH VERIFICATION SCENARIOS

```
public class QuickMathsTest {  
    public void test1() {  
        QuickMaths victim = new QuickMaths();  
  
        int a = 3;  
        int b = 5;  
  
        int expectedResult = 5;  
  
        int actualResult = victim.max(3, 5);  
  
        check(actualResult, expectedResult, "test1");  
    }  
  
    public void check(int actualResult, int expectedResult, String testName) {  
        if (actualResult == expectedResult) {  
            System.out.println(testName + " has passed!");  
        } else {  
            System.out.println(testName + " has failed!");  
            System.out.println("Expected " + expectedResult + " but was " + actualResult);  
        }  
    }  
}
```

2.B. INSTANTIATE TEST CLASS AND CALL VERIFICATION METHODS

```
public class QuickMathsTest {  
    public static void main(String[] args) {  
        QuickMathsTest testRunner = new QuickMathsTest();  
        testRunner.test1();  
    }  
    ...  
}
```

3. RUN AND CHECK RESULTS

test1 has passed!

Process finished with exit code 0