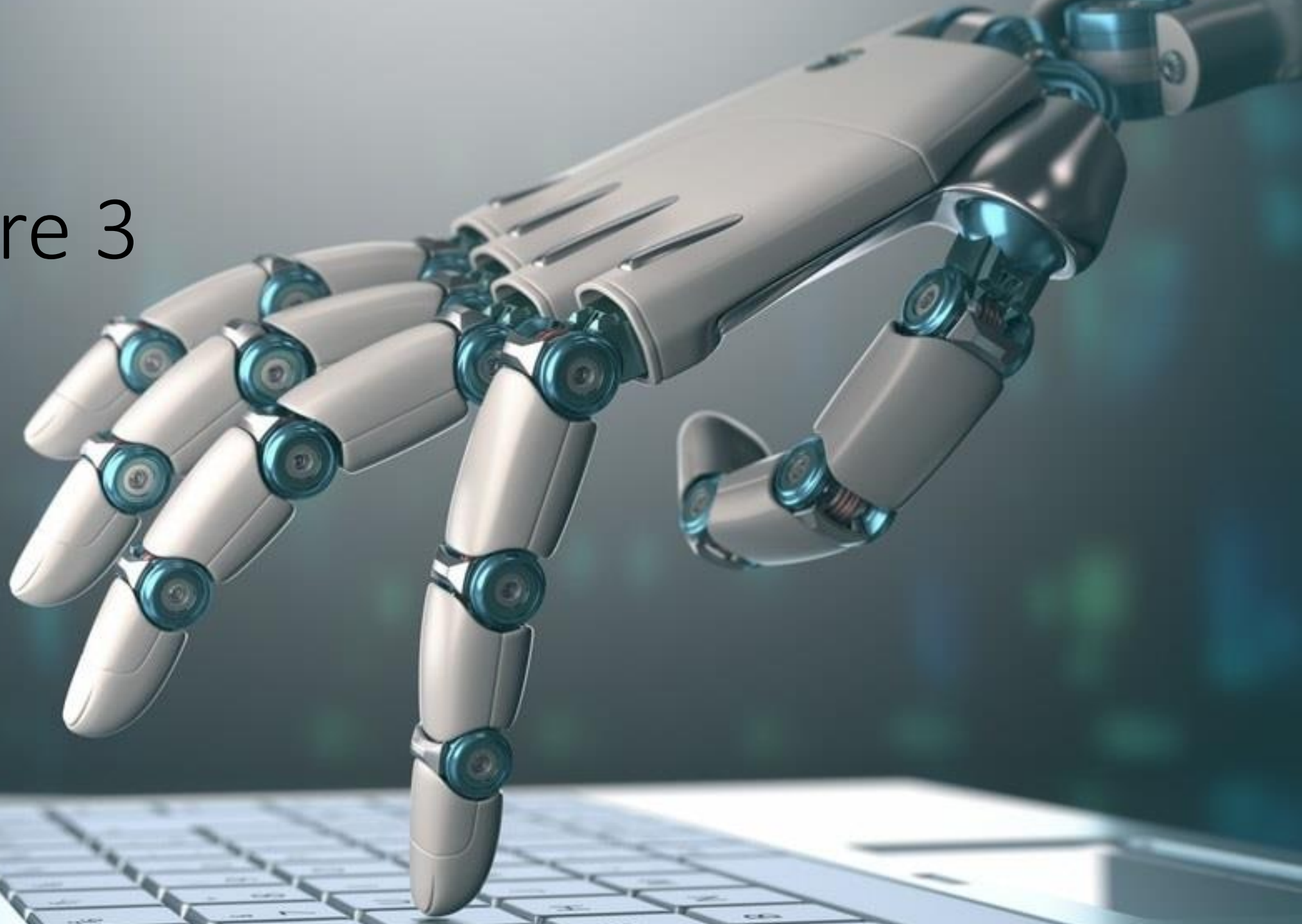


Lecture 3



Agenda

- **If flow**
- Switch
- Practice

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

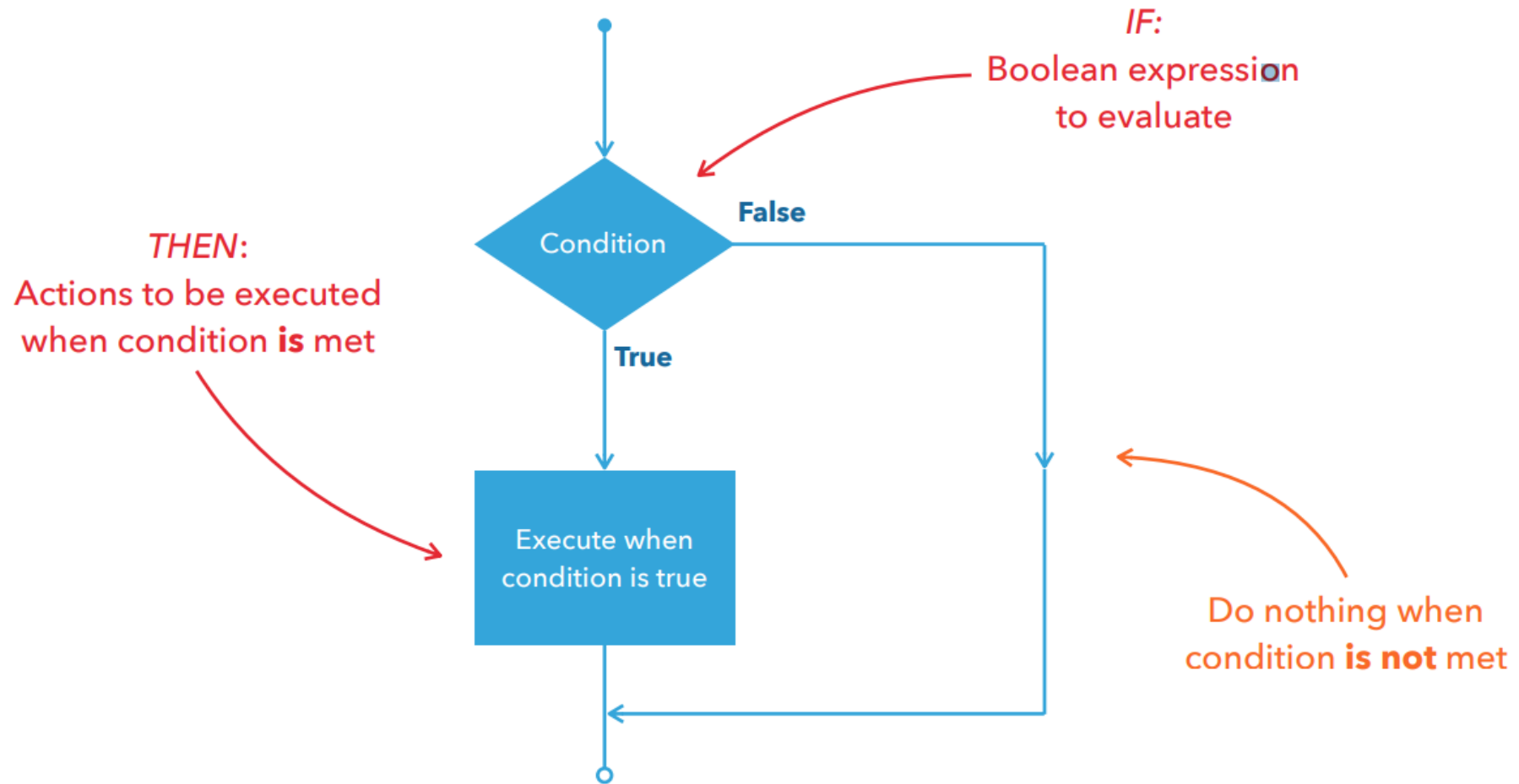
THE EQUALITY AND RELATIONAL OPERATORS

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

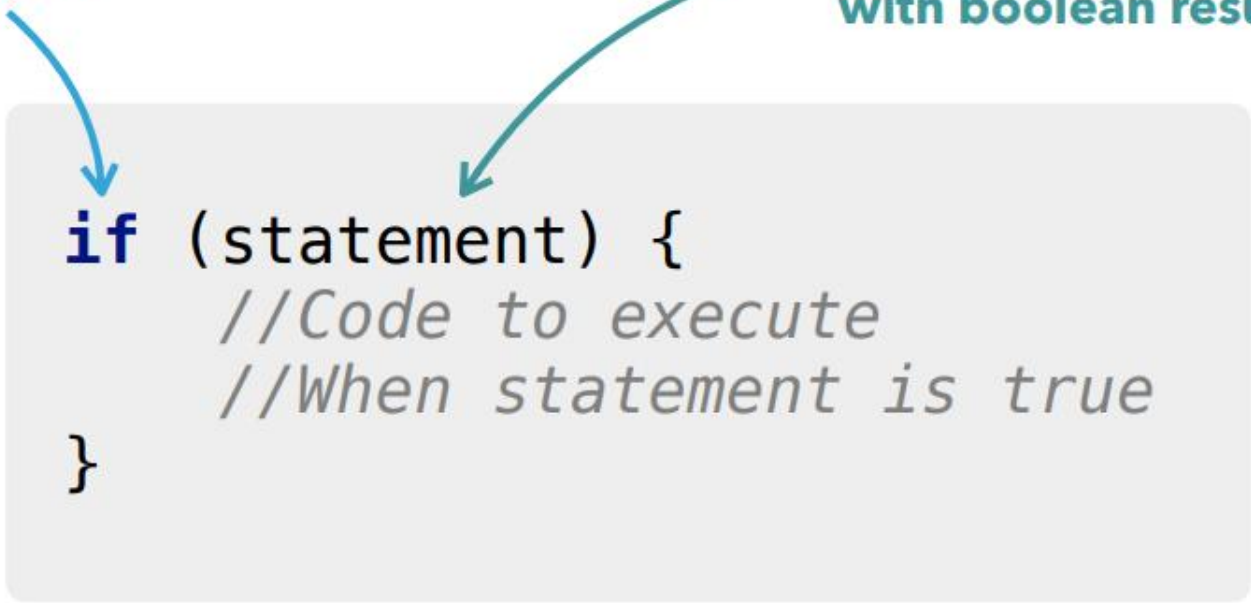
DECISION MAKING FLOWCHART: IF



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

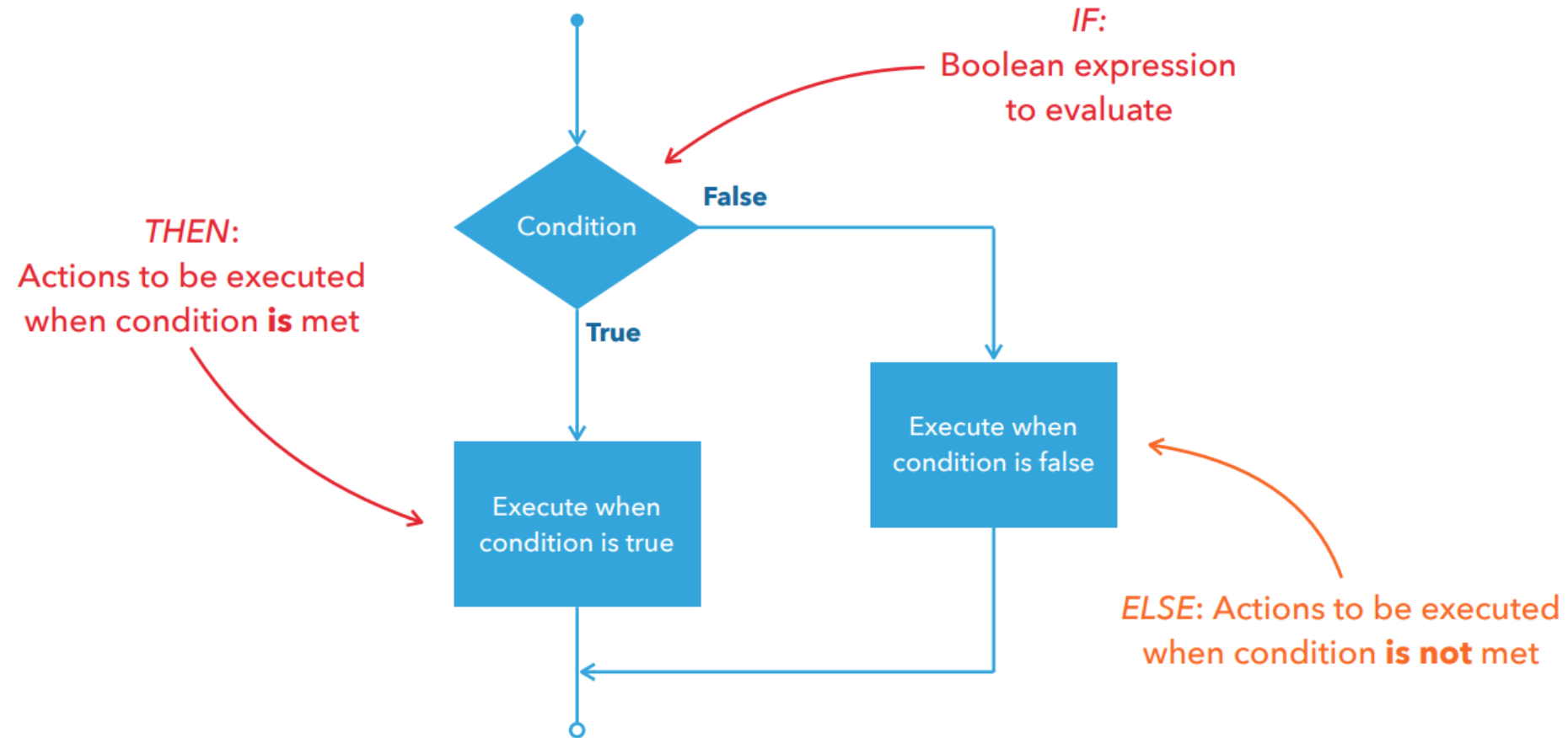
```
int currentPage = 5;
int lastPage = 10;

if (currentPage < lastPage) {
    System.out.print("Current
page less than last page");
}
```


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

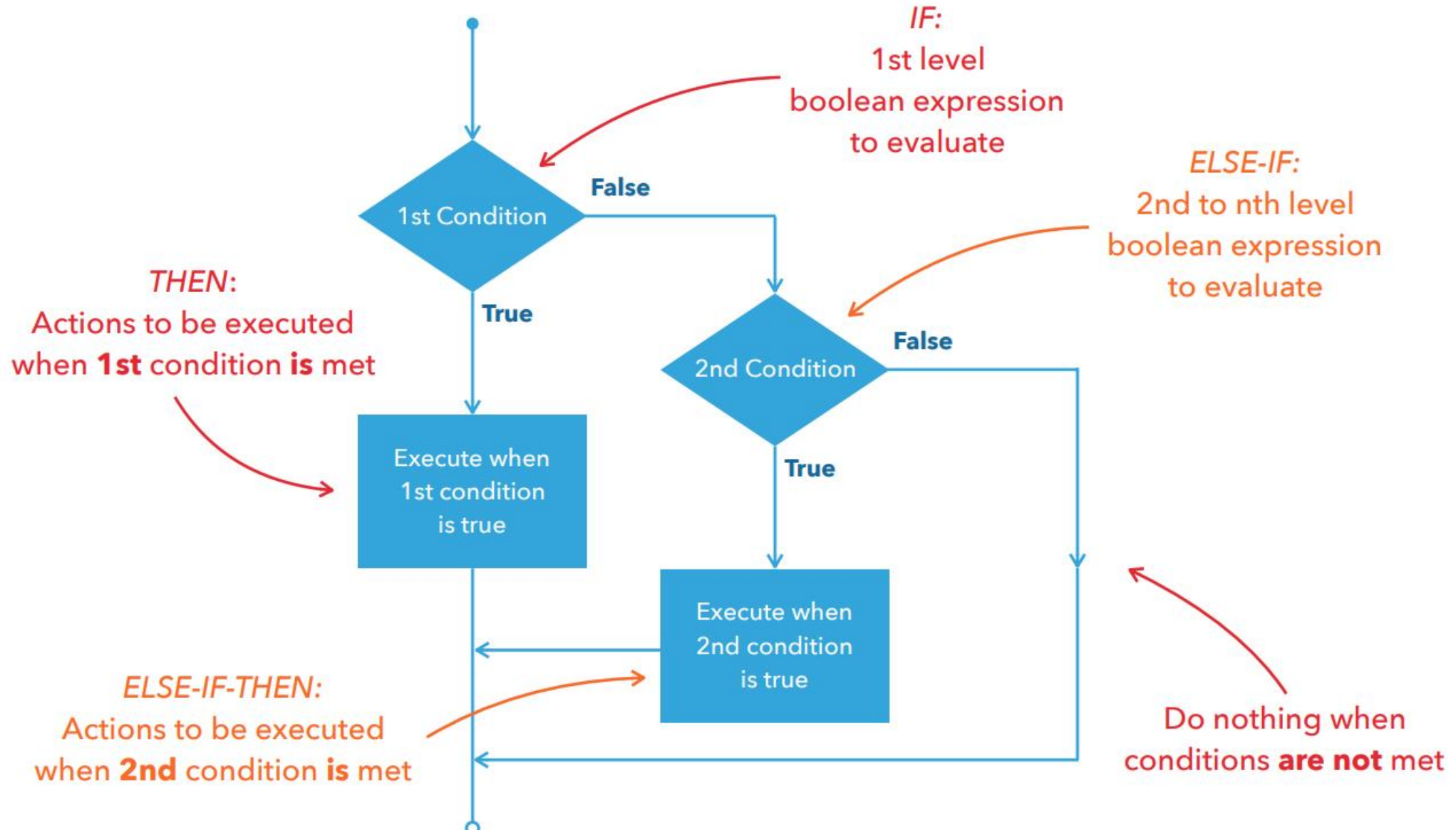
```
int x = 5;

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;

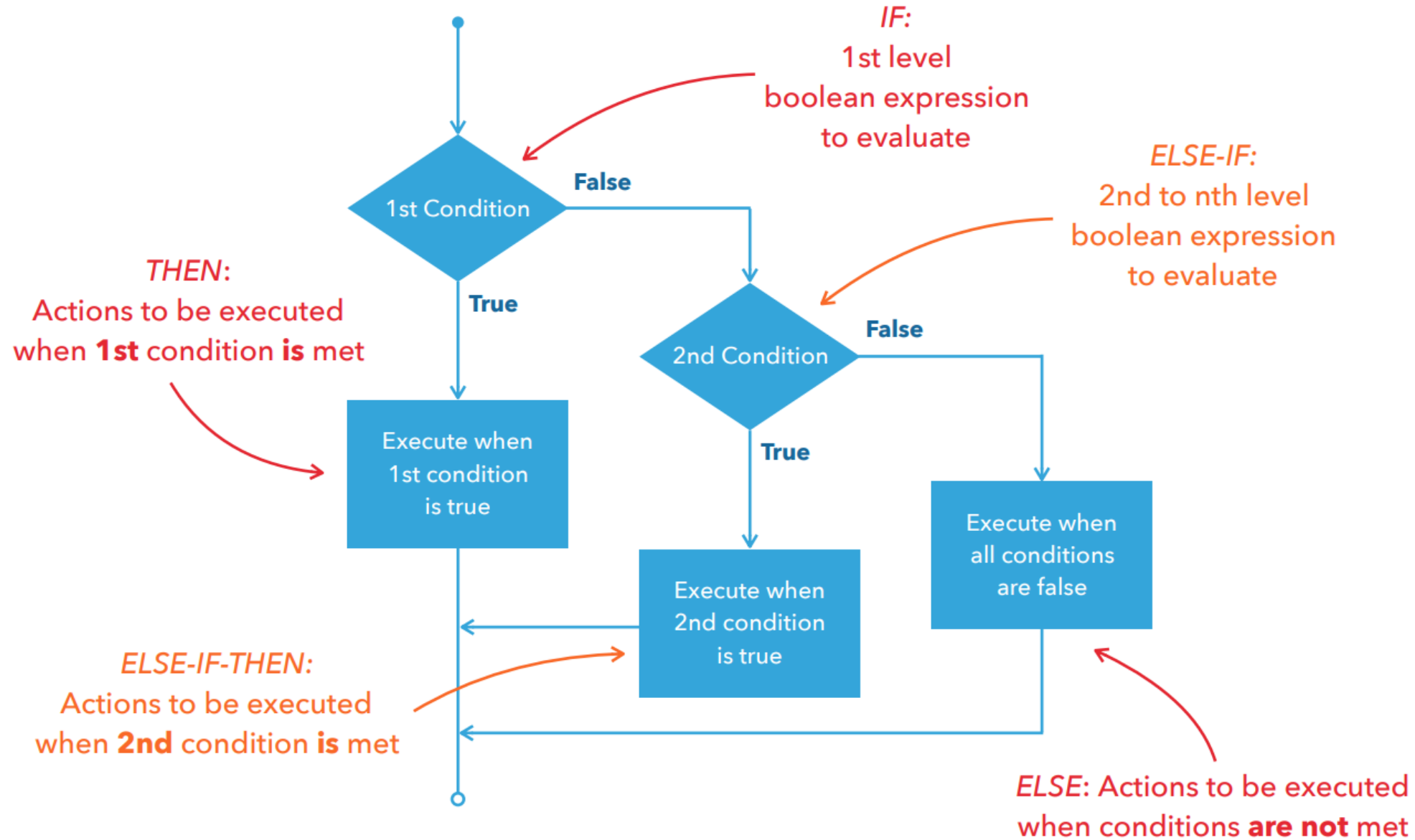
if (flag1) {
    System.out.print("flag1");
} else if (flag2) {
    System.out.print("flag2");
}
```

Inline expression

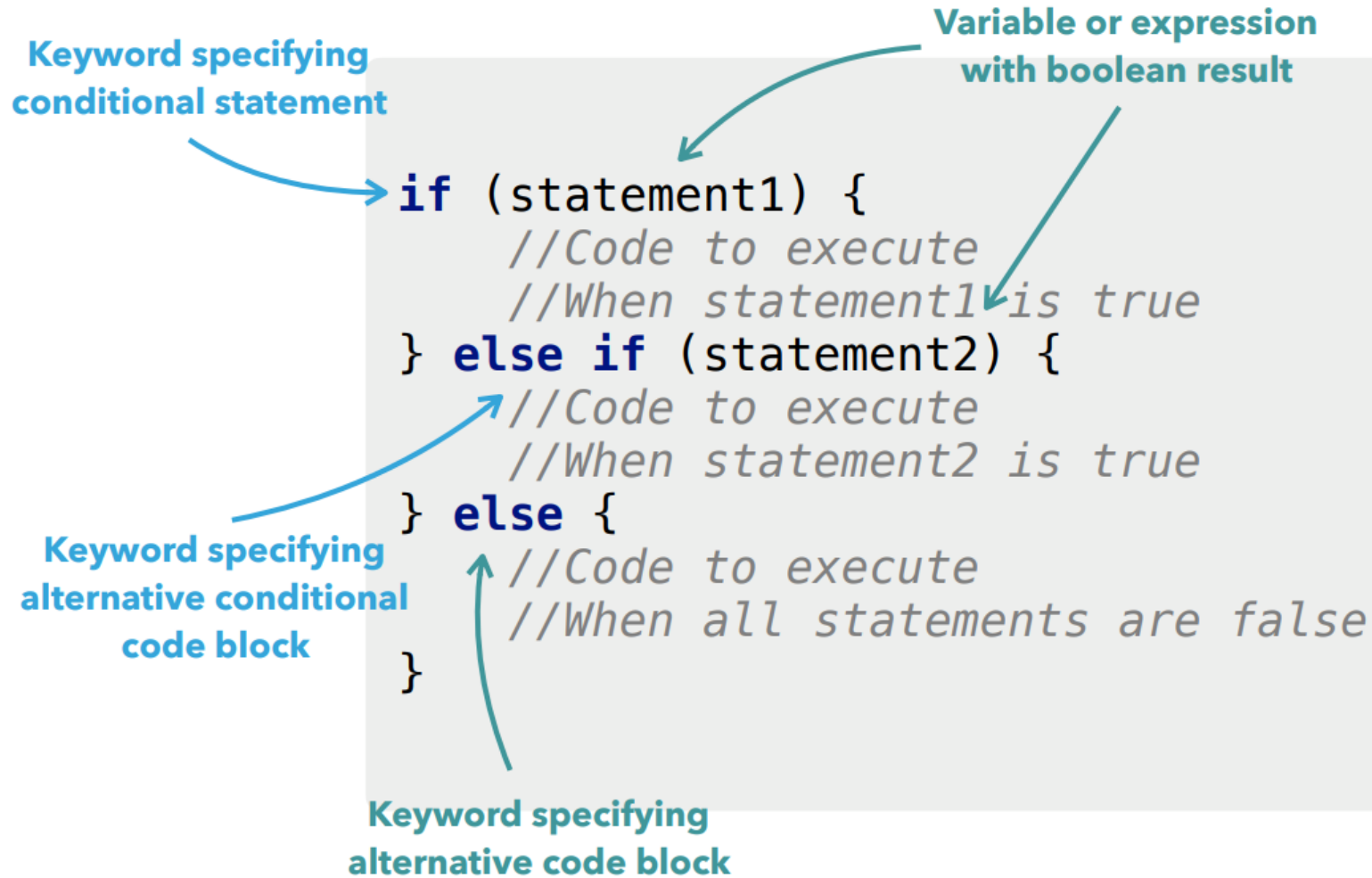
```
int currentPage = 7;

if (currentPage == 3) {
    System.out.print("Current
page: 3");
} else if (currentPage == 7) {
    System.out.print("Current
page: 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

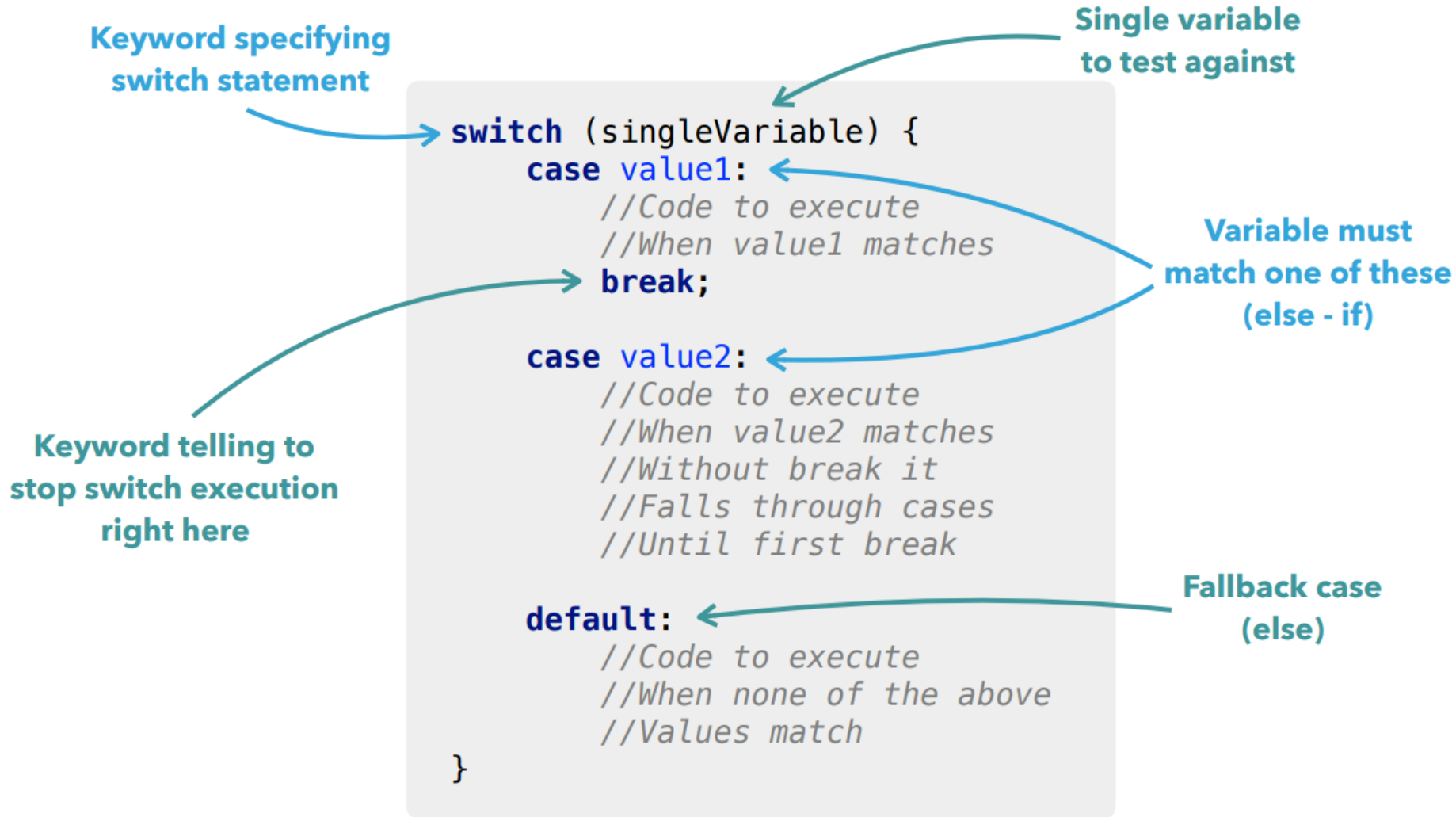
Agenda

- If flow
- **Switch**
- Practice

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?");
}
```


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!");  
}
```

Agenda

- If flow
- Switch
- **Practice**

Task 1

- Create a noise detector:

Уровень громкости (dB)	Эффект
< 39	Faint
40 – 69	Moderate
70 – 99	Very Loud
100 – 129	Extremely Loud
130 >	Painful

Task 2

В зависимости от числа (1 – 7) будет выводиться соответствующее название дня недели. Любое другое число будет возвращать ошибку. Дополнительные классы для реализации логики создаваться не будут, ограничимся только классом с методом `main()`.

Task 3

- Modify our bank program
- Withdrawal = balance < 0 – display error – you don't have enough money
- Withdrawal = amount > 700 – display error - daily limit exceeded
- Deposit = amount >= 10000 – display error – need to register sum in VID

Task 4

Описание:

Разработать программу, которая работает в соответствии с требованиями, описанными ниже.

Функциональные требования:

Программа должна определять цвет в зависимости от длины волны в соответствии со следующими правилами:

- 380 ... 449 - Фиолетовый ("*Violet*")
- 450 ... 494 - Синий ("*Blue*")
- 495 ... 569 - Зеленый ("*Green*")
- 570 ... 589 - Желтый ("*Yellow*")
- 590 ... 619 - Оранжевый ("*Orange*")
- 620 ... 750 - Красный ("*Red*")
- Вне диапазонов - невидимый спектр ("*Invisible Light*")

Логика с определением цвета должна быть реализована в отдельном классе `LightColorDetector`:

```
public class LightColorDetector {  
  
    public String detect(int wavelength) {  
        //TODO  
    }  
  
}
```

Task 5

Описание:

Разработать программу, которая работает в соответствии с логикой, описанной ниже.

Функциональные требования:

Программа должна определять тип числа и возвращать описание знака числа в соответствии со следующими правилами:

- "Number is positive", если число положительное (больше 0);
- "Number is negative", если число отрицательное (меньше 0);
- "Number is equal to zero", если число равно 0;

Логика с определением знака числа должна быть реализована в отдельном классе:

```
public class SignComparator {  
  
    public String compare(int number) {  
        //TODO  
    }  
  
}
```

OBJECT INSTANTIATION IN JAVA: SYNTAX

- ▶ Object instantiation **without** assignment

```
new Window();
```

- ▶ Object instantiation **with** assignment

```
Window firstWindow = new Window();
```


CONSTRUCTORS

- ▶ **Every class** has a constructor
- ▶ If **explicit** constructor(s) is **not specified** in code, Java Compiler will generate **default** constructor implicitly
- ▶ **Each time** a new object is created, **at least one** constructor will be **invoked**
- ▶ **Each** defined constructor must have **unique** signature (i.e. ordered number and type of arguments)

CONSTRUCTOR DECLARATION IN JAVA: EXAMPLE BREAKDOWN

```
public class Window {  
    private String title;  
    public Window() {  
    }  
    public Window(String title) {  
        this.title = title;  
    }  
}
```

**Explicit default
constructor without
arguments** →

**Explicit constructor
with argument
and initialisation** ←

Task 6 – Part 1

Описание:

Разработать программу, которая работает в соответствии с требованиями, описанными ниже.

Функциональные требования:

Необходимо реализовать класс `Stock` ("акция") таким образом, чтобы была возможность узнать текущую цену акции, а также ее максимальную и минимальную стоимость за период существования.

Класс `Stock` должен обладать следующими характеристиками:

- Свойства
 - Имя компании
 - Текущая стоимость
 - Минимальная стоимость
 - Максимальная стоимость
- Методы
 - Обновить текущую стоимость акции `updatePrice()`
 - Распечатать информацию об акции `printInformation()`

Имя компании и начальную стоимость необходимо задавать в момент создания акции. Текущая, минимальная и максимальная стоимость должны меняться только через метод `updatePrice()`.

Task 6 – Part 2 - Example

Пример:

Работа с классом:

```
Stock google = new Stock("GOOG", 10);  
google.printInformation();  
  
google.updatePrice(15);  
google.updatePrice(7);  
google.updatePrice(14);  
  
google.printInformation();
```

Вывод в консоль:

```
Company = "GOOG", Current Price = 10, Min Price = 10, Max Price = 10
```

```
Company = "GOOG", Current Price = 14, Min Price = 7, Max Price = 15
```



THANK YOU FOR YOUR ATTENTION

