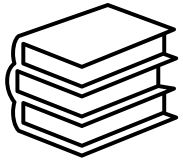
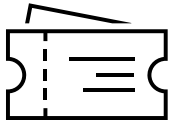
Data Structures . Introduction

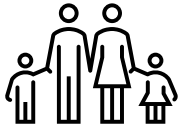
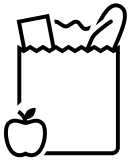
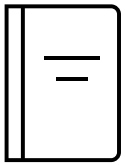
What is Data Structures?

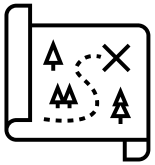
Where is data?





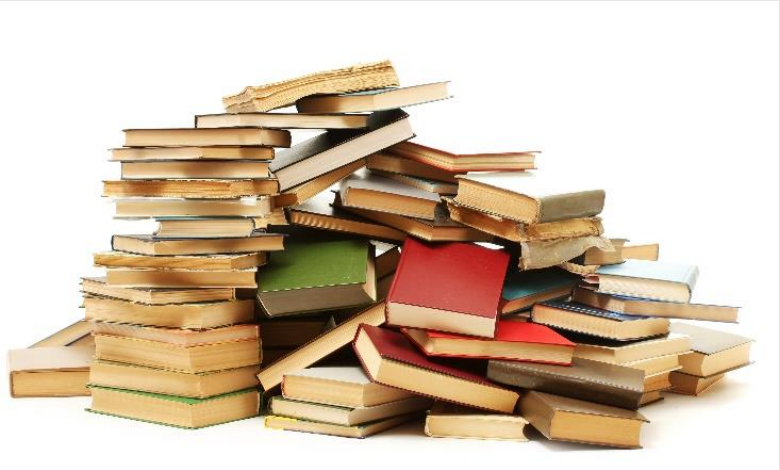

Data exists in various structures everywhere.



Data Structure

Before DS

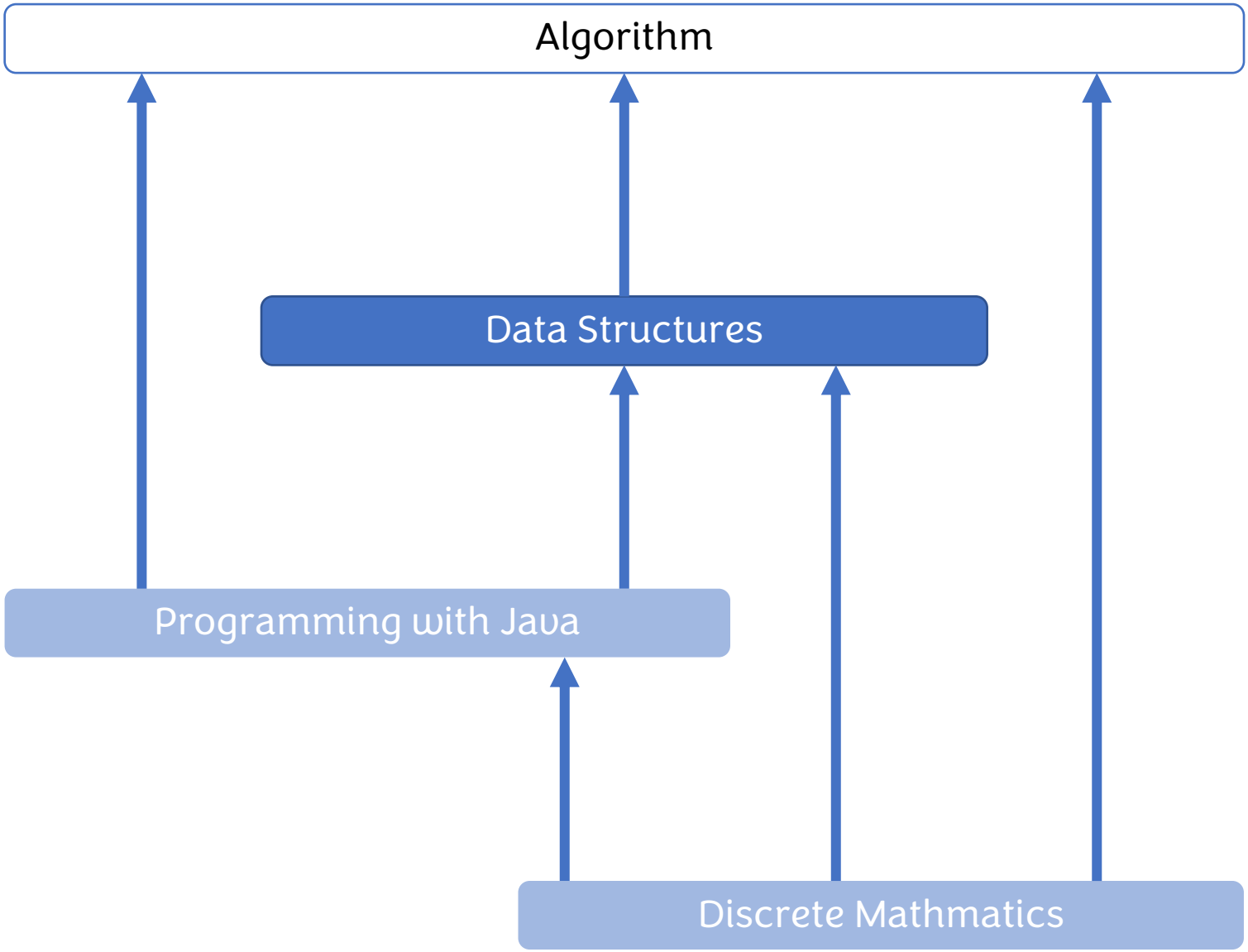


After DS

Role of DS

- Data structure = Component to solve problems
- Purpose of this module
  - Let you know various data structures
  - Let you can select a proper data structure for a given problem

Related Subjects



Programmer?


What my friends think I do
What my mom thinks I do
What society thinks I do
What my boss thinks I do
What I think I do
What I actually do

Programmer's Thinking

- Knowledge
  - Declarative knowledge
    - Kimchi

    - $y = \sqrt{x}$ if $x = y^2$
  - Imperative knowledge
    - Recipe of Kimchi
    - How to calculate $y = \sqrt{x}$

What you learn

Data structures

Linear data structures

- List
- Stack
- Queue

Index data structures

- Binary tree
- Hash table

Nonlinear data structures

- Tree
- Heap
- Graph

Advanced Java

Review of the concept of OO

Generics

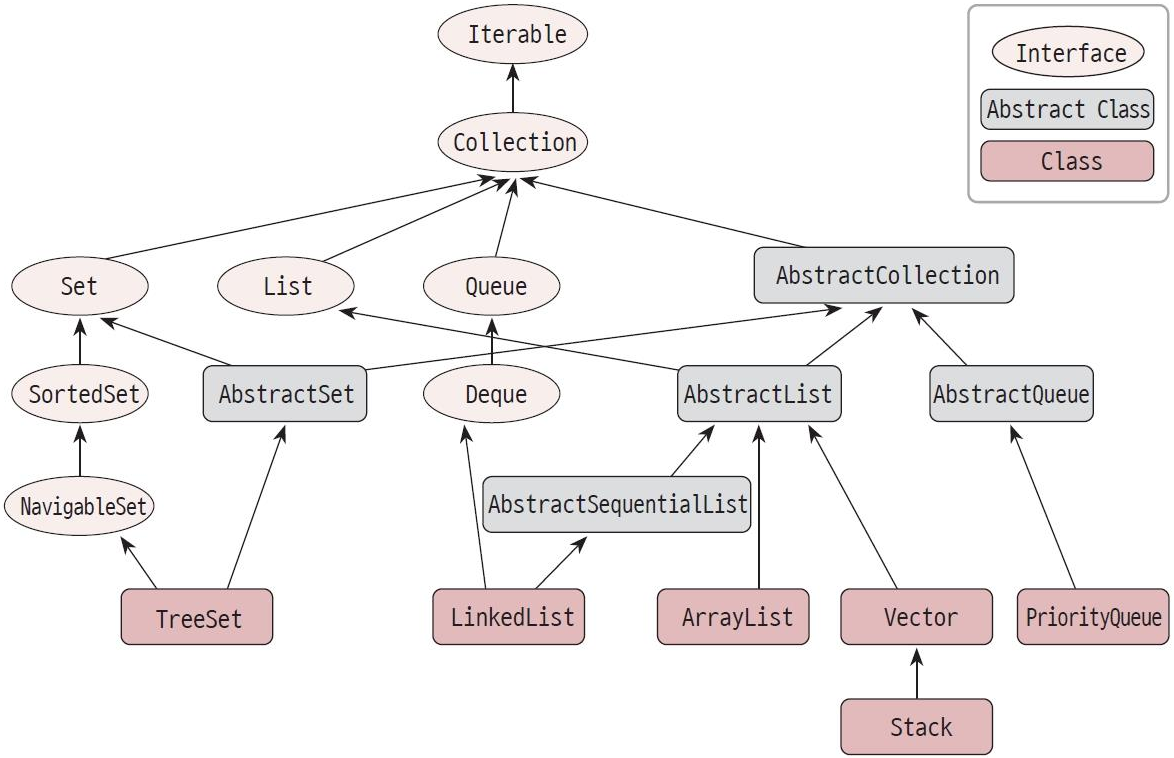
Thinking Tech.

Recursion

Sorting

What you learn

- Collections for data structures in Java package



Algorithm

- a procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation
- Data structures are components building an algorithm

- Representation
  - natural languages
  - Pseudocode
  - Flowcharts
    - pictorial representation of a program's logic.
  - programming languages
  - control tables (processed by interpreters)

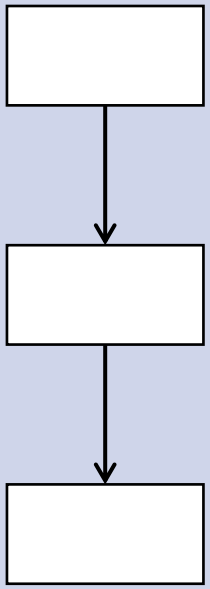
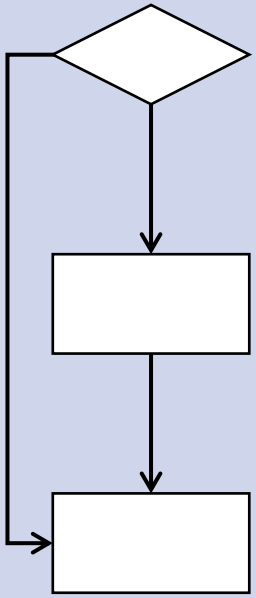
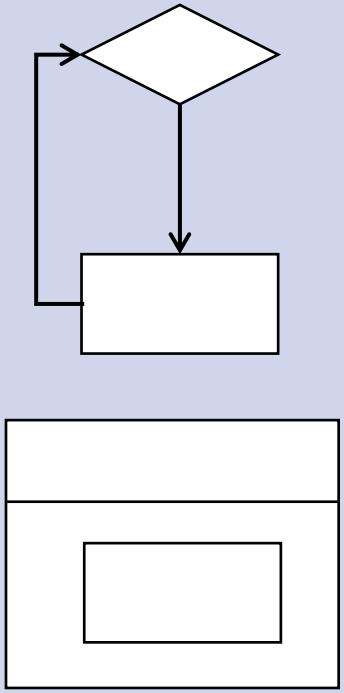
Symbols

Usage	Symbol
Process	
Decision Used to denote a decision point where alternate paths are possible	
Start/End Used to indicate the beginning or ending of a flowchart	
Loop	
Input/Output	
Link	↓

Benefits

- Portrays the logical "flow" between tasks
- Visually successful in communicating the sequence of tasks
- Easy to know the sequential impact of changes
- Easy to spot redundant operations & other inefficiencies
- Training tool for new employees
- Spot internal control weaknesses
- Helps the auditor see the “whole picture”

3 control flows

Sequential	Branching	Loop
		

Sequential

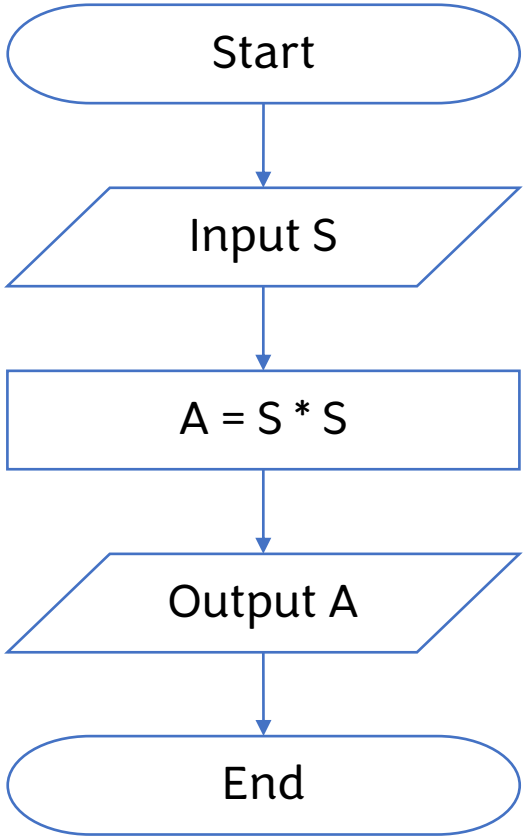
- Sequential statements are executed in sequence, one after the other. (examples – print, input, assignment.)




Wake up → Wash up → dress → Go to school

Example

- Write a flowchart that gets the length of a side as input and outputs the area of a square.



1. Input S for the length of a side of a square
2. Calculate the area of a square using a formula
3. Output the area

Branching

- Branching statements contain one or more choices and only one of the choices is executed
- “if” statements can be categorized like:
  - where you want to do something or nothing.
  - Use the second format (if, else) for problems where you want to do one thing or another thing
  - Use the third format (if, else, if, else) for problems where you want to do one thing out of three or more choices.

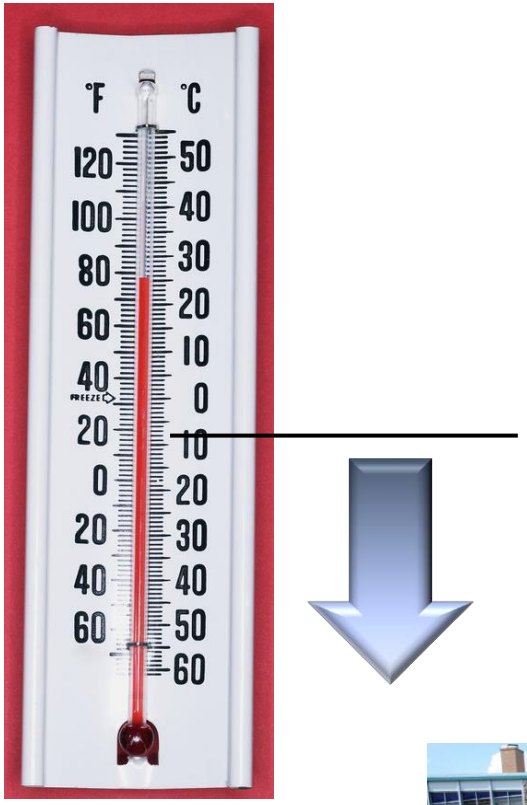
Branching 0 or 1

- where you want to do something or nothing.
- example:

  Write an algorithm that prints "No school!" if temperature is below -10 °C.

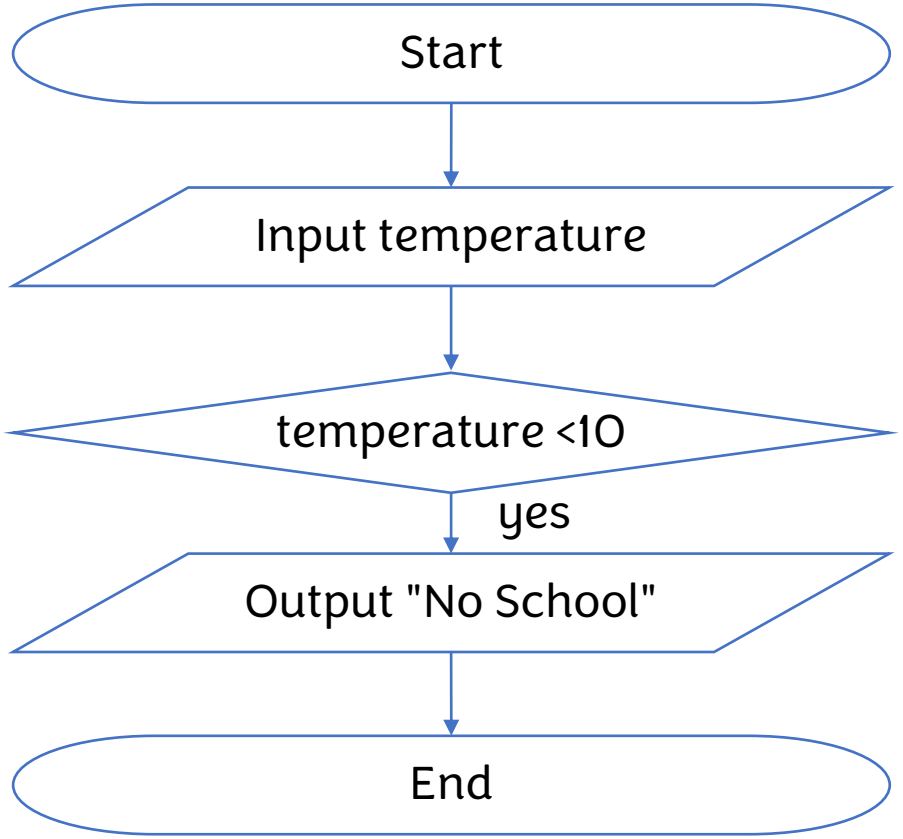
- Java code

```
if (temperature < 10)
    System.out.println("No School!");
```



Flowchart

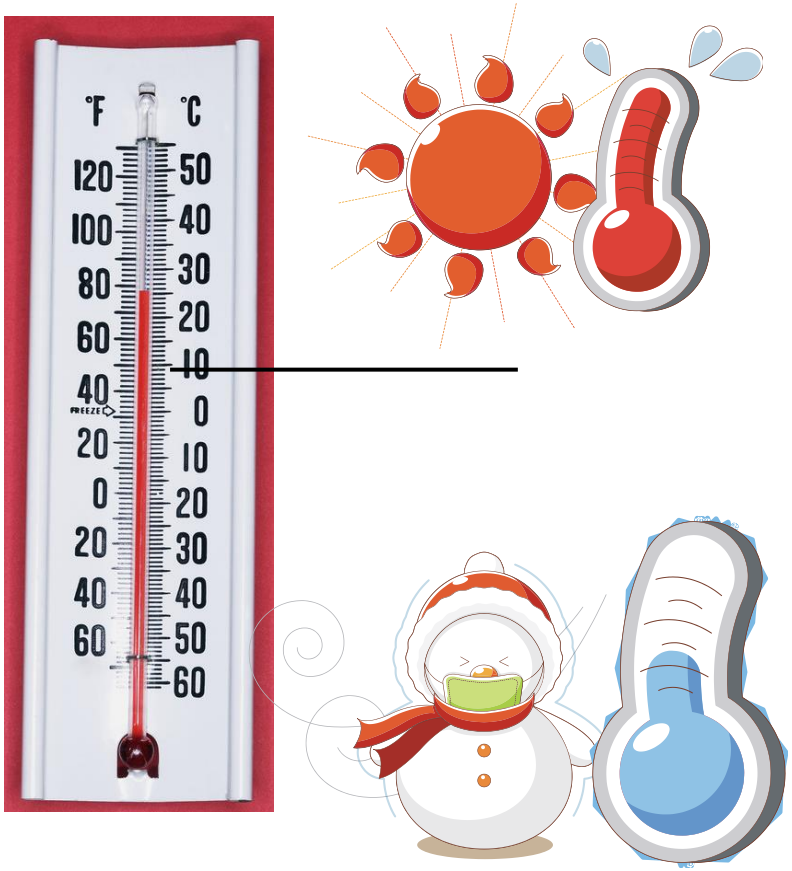
```
if (temperature < 10)
        System.out.println("No School!");
```



Branching 2 choices

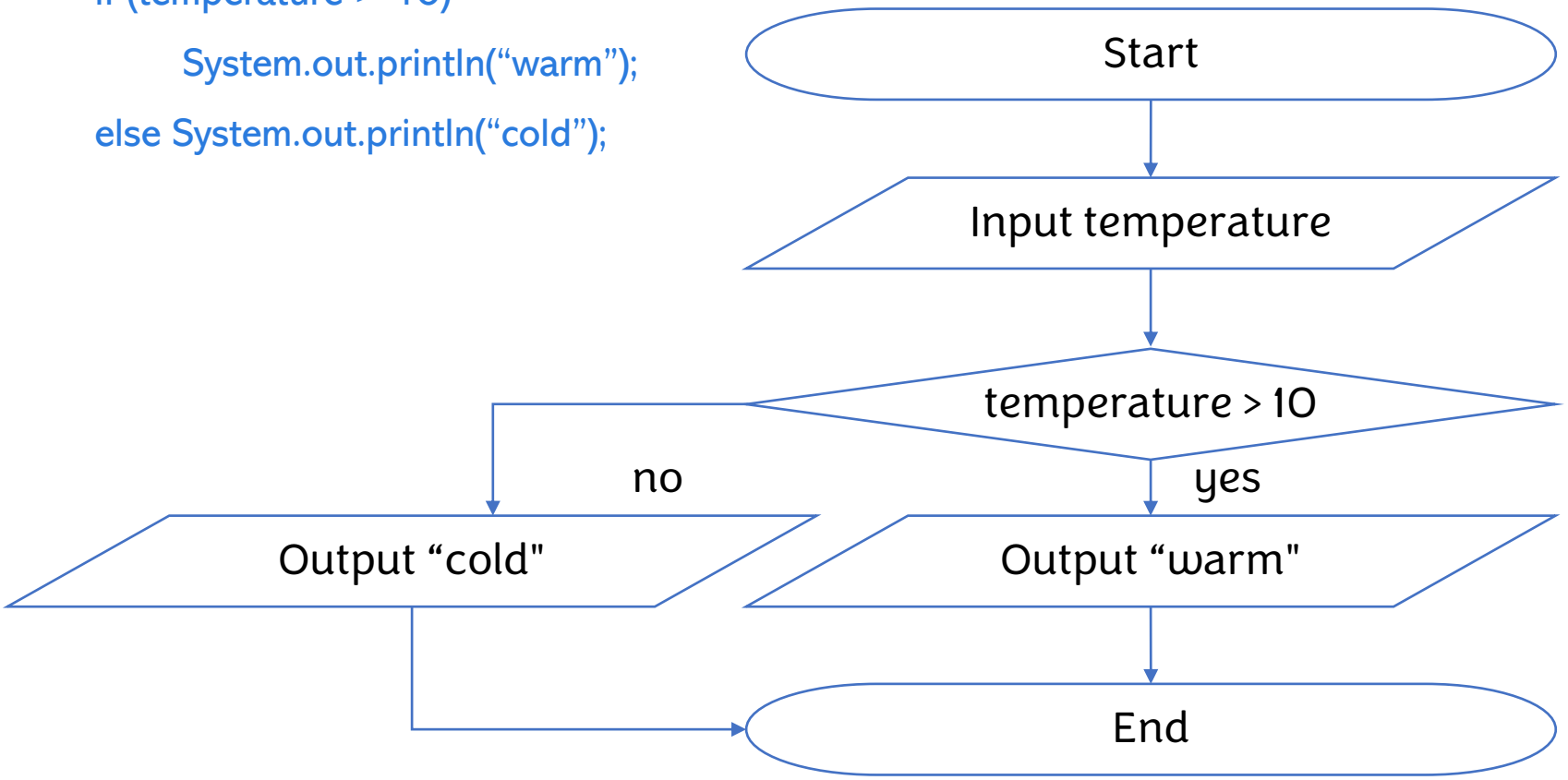
- where you want to do one thing or another thing.
- example:
  - Write an algorithm that prints "warm" if temperature (a variable) is above 10 °C and prints "cold" otherwise.
- Java code

```
if (temperature > 10)
        System.out.println("warm");
else System.out.println("cold");
```



Flowchart

```
if (temperature > 10)
        System.out.println("warm");
else System.out.println("cold");
```



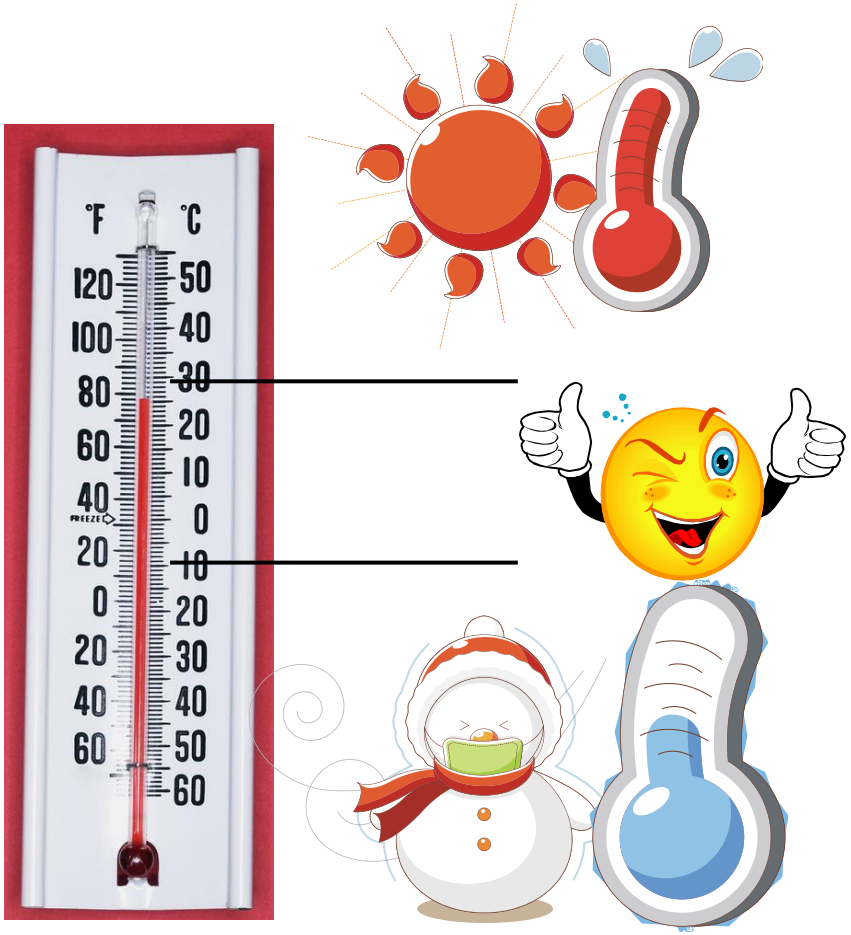
Branching multi choices

- where you want to do one thing out of three or more choices.
- example:

  Write an algorithm that prints "too cold" if temperature is below -10 °C, "OK" if the temperature is between 10 and 30 °C, and "too hot" if the temperature is above 30 °C.

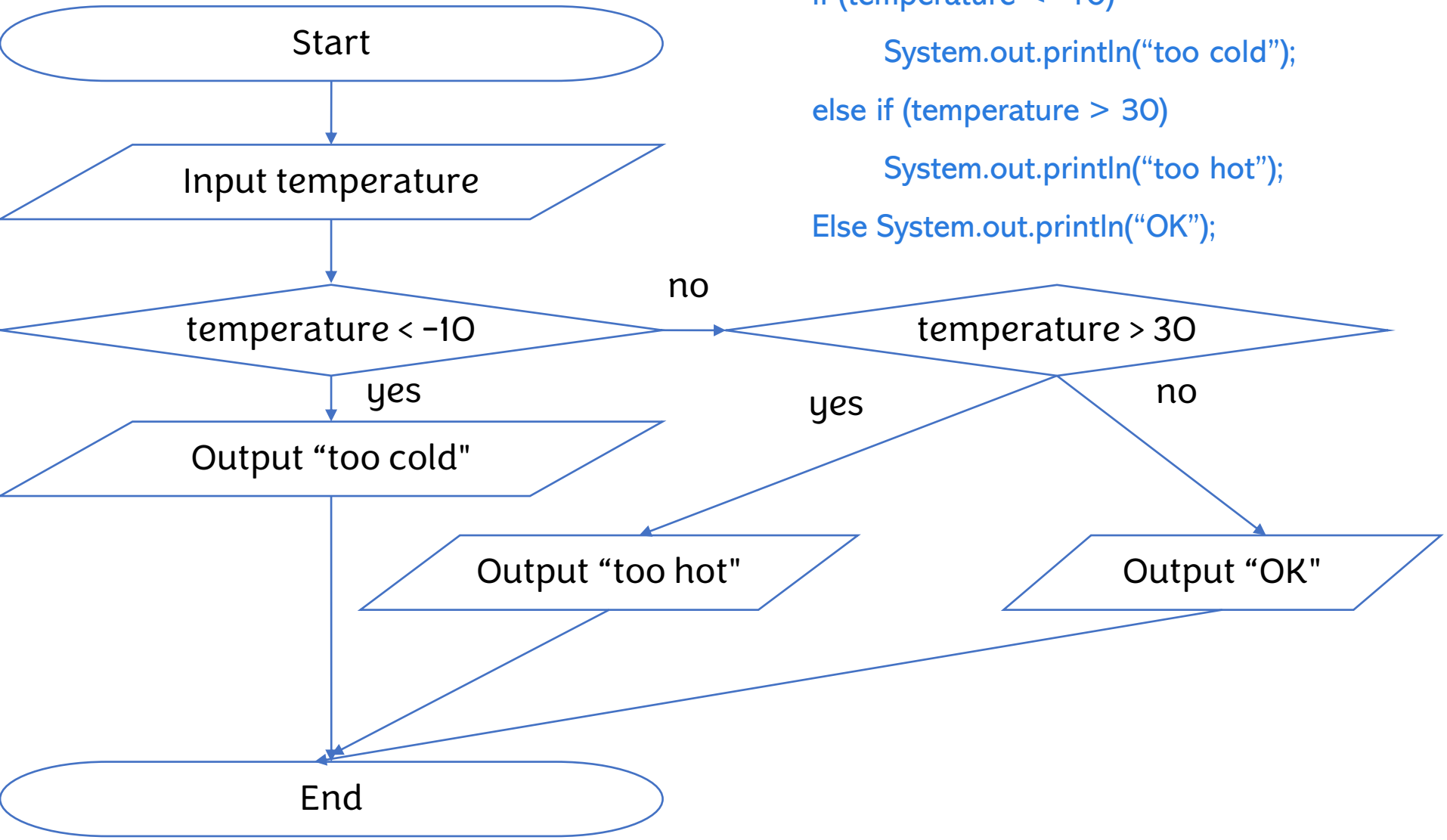
- Java Code

```
if (temperature < -10)
        System.out.println("too cold");
else if (temperature > 30)
        System.out.println("too hot");
Else System.out.println("OK");
```



Flowchart

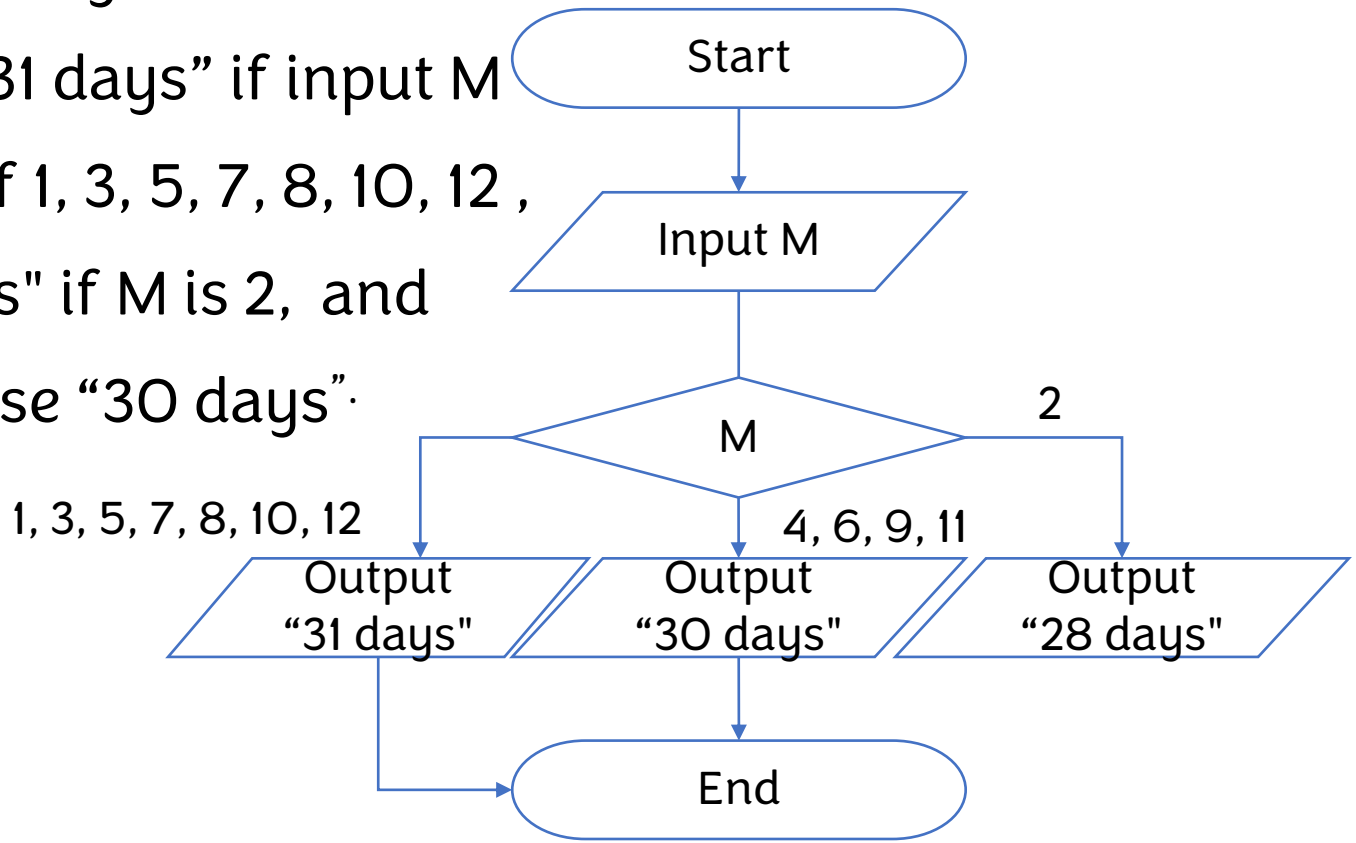
```
if (temperature < -10)
        System.out.println("too cold");
else if (temperature > 30)
        System.out.println("too hot");
Else System.out.println("OK");
```



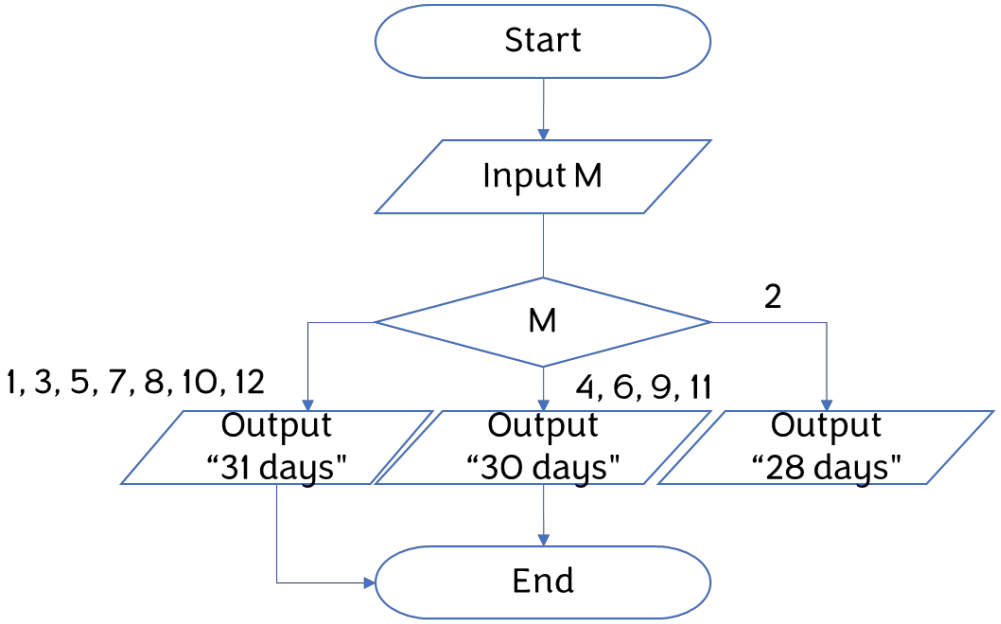
Branching multi choices

- example:

  Write an algorithm that prints "31 days" if input M is one of 1, 3, 5, 7, 8, 10, 12 , "28 days" if M is 2, and otherwise "30 days".



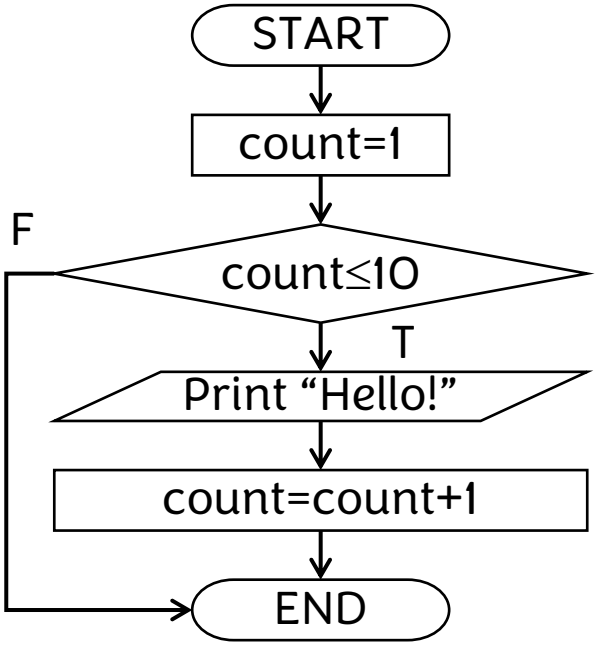
Branching multi choices

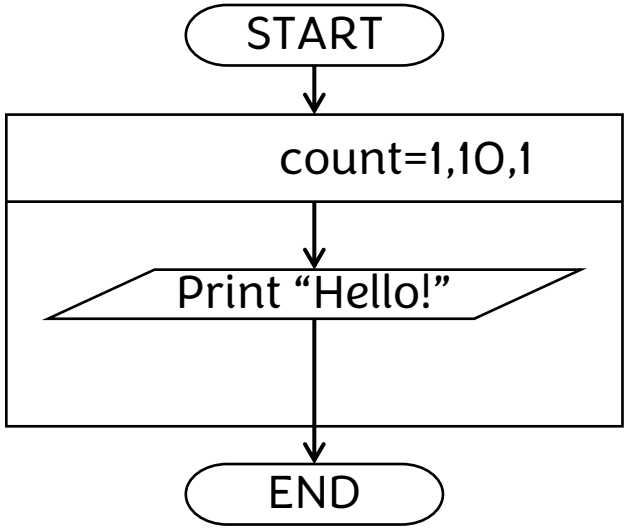



```
switch(M) {
        case 4:
        case 6:
        case 9:
        case 11:
                System.out.println(“30 days“);
                break;
        case 2:
                System.out.println(“28 days“);
                break;
        default:
                System.out.println(“31 days“);
}
```

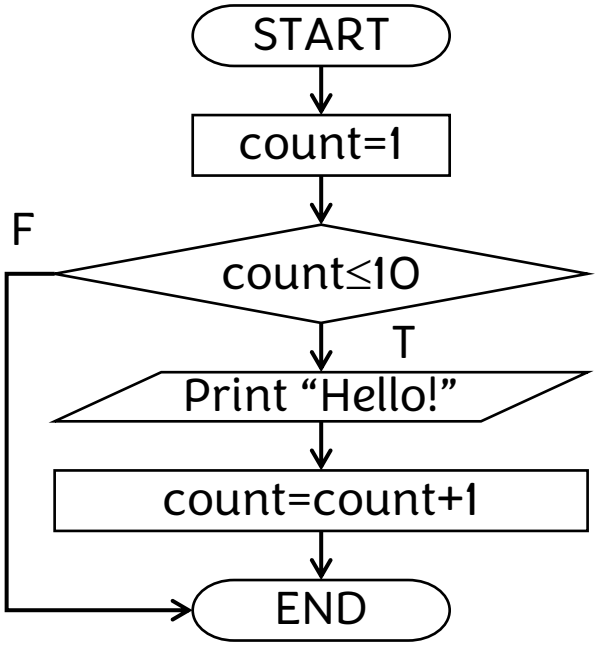
Loop

- Loop statements cause you to jump back to a previously executed statement. By continuing at that previous statement, a loop is formed.
- print “Hello!” 10 times





Java Code



FlowchartLoop.java

```java
public class FlowchartLoop {
    public static void main(String[] args) {
        int count = 1;
        while (count <= 10) {
            System.out.println("Hello!");
            count = count +1;
        }
    }
}
```

Problems @ Javadoc Declaration Console

<terminated> FlowchartLoop [Java Application] C:₩Users₩user₩.p2

```
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
```

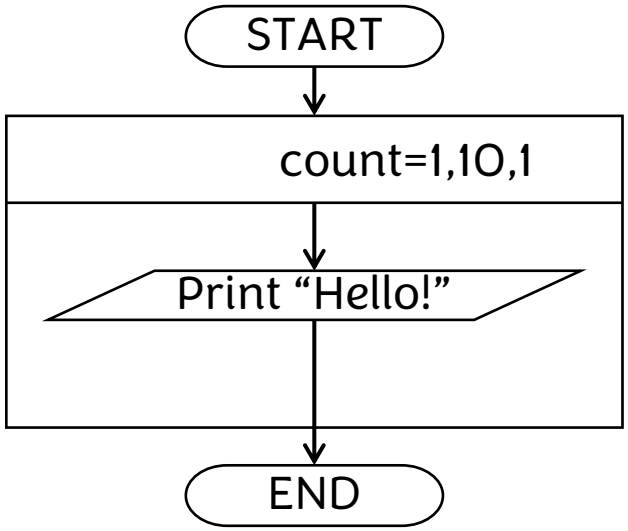
Java Code

FlowchartLoop.java

```java
public class FlowchartLoop {
    public static void main(String[] args) {
        for(int count = 0 ; count < 10 ; count ++ )
            System.out.println("Hello!");
    }
}
```

Problems @ Javadoc Declaration Console

```
<terminated> FlowchartLoop [Java Application] C:₩Users₩user₩.p2₩pool₩
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
```



Thank you!

Questions?

Exit