/ JAVA DB SCHOOL
Java Intro

/ Intro

Meet the Trainers

Meet the Students

1. Who are you?

2. What is your tech experience?

3. Why are you studying this course?

/ Java Setup

Required Setup

1. Download Java Development Kit (JDK) at least JDK – https://www.oracle.com/java/technologies/javase-jdk11-downloads.html

2. Unarchive JDK in a convenient location

3. Most IDEs will autodetect the JDK

4. Try to run Java: open terminal (CMD in Windows) and type: java -version

5. Install IntelliJ IDEA or Eclipse. You may also install Codota plugin for IntelliJ IDEA

6. Run IntelliJ IDEA

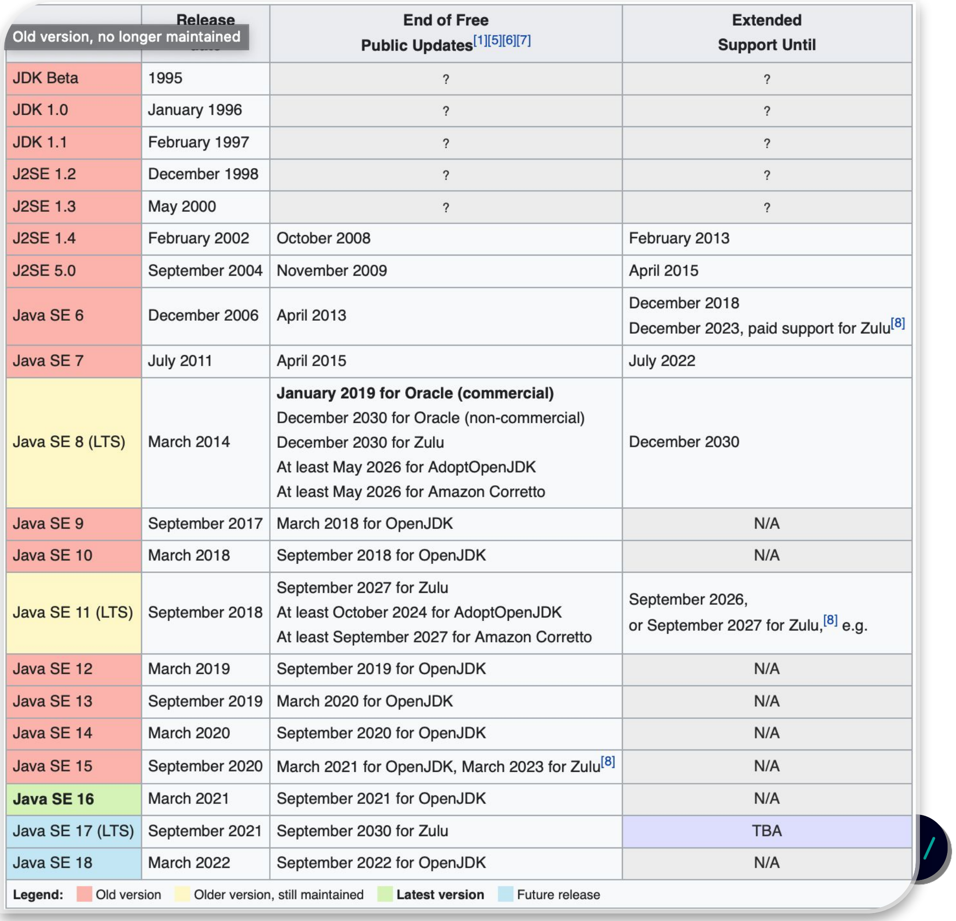
/ About Java

What is Java

- High-level, class-based, object-oriented programming language
- Fewer low-level facilities than other programming languages such as C/C++
- Every data item is an object (exception: primitive data types)
- Intended to let application developers write once, run anywhere
- Compiled Java code can run on all platforms that support Java without the need for recompilation (using Java Runtime Environment)

Java Versions

- Java 7 – very popular in the past
- Java 8 brings lambda expressions
- Java 11 – Java 10 was the last free Oracle JDK that could be downloaded



Old version, no longer maintained

	Release	End of Free Public Updates[1][5][6][7]	Extended Support Until
JDK Beta	1995	?	?
JDK 1.0	January 1996	?	?
JDK 1.1	February 1997	?	?
J2SE 1.2	December 1998	?	?
J2SE 1.3	May 2000	?	?
J2SE 1.4	February 2002	October 2008	February 2013
J2SE 5.0	September 2004	November 2009	April 2015
Java SE 6	December 2006	April 2013	December 2018 December 2023, paid support for Zulu[8]
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	**January 2019 for Oracle (commercial)** December 2030 for Oracle (non-commercial) December 2030 for Zulu At least May 2026 for AdoptOpenJDK At least May 2026 for Amazon Corretto	December 2030
Java SE 9	September 2017	March 2018 for OpenJDK	N/A
Java SE 10	March 2018	September 2018 for OpenJDK	N/A
Java SE 11 (LTS)	September 2018	September 2027 for Zulu At least October 2024 for AdoptOpenJDK At least September 2027 for Amazon Corretto	September 2026, or September 2027 for Zulu,[8] e.g.
Java SE 12	March 2019	September 2019 for OpenJDK	N/A
Java SE 13	September 2019	March 2020 for OpenJDK	N/A
Java SE 14	March 2020	September 2020 for OpenJDK	N/A
Java SE 15	September 2020	March 2021 for OpenJDK, March 2023 for Zulu[8]	N/A
Java SE 16	March 2021	September 2021 for OpenJDK	N/A
Java SE 17 (LTS)	September 2021	September 2030 for Zulu	TBA
Java SE 18	March 2022	September 2022 for OpenJDK	N/A

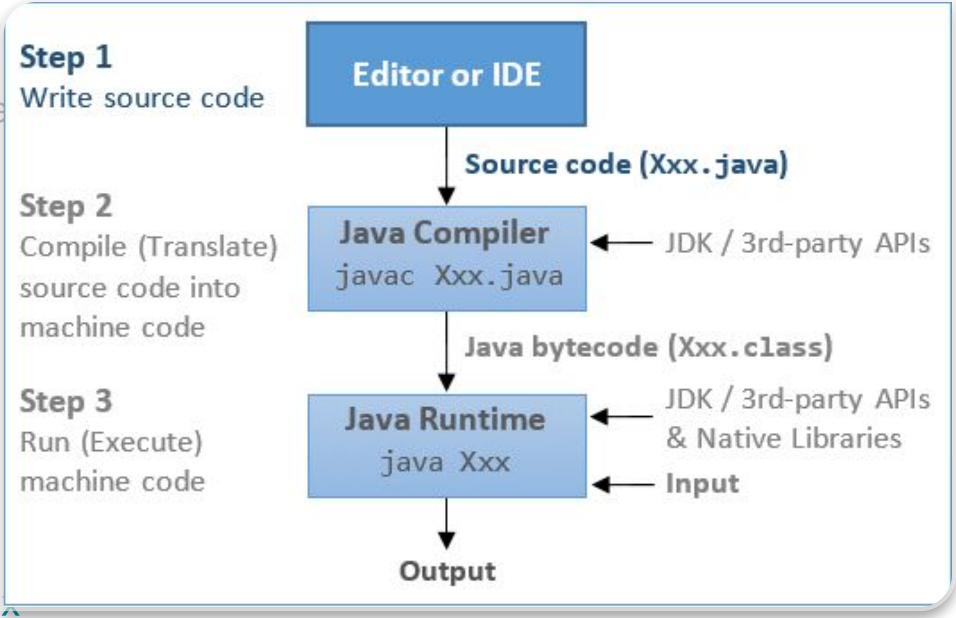
Legend: 🟥 Old version 🟨 Older version, still maintained 🟩 **Latest version** 🟦 Future release

Java Compilation

- Code is compiled in .class files using a terminal/command line compiler, or a compile installed inside an IDE

- .class files are run on Java Runtime Environment (JRE) using the java command in a terminal/command line, or using the Run function of an IDE

- Further lecture: https://www.geeksforgeeks.org/compilation-execution-java-program



/ Before We Go Further…

What is Java

- Always **R**ead **T**he **F*****ing **M**anual!
- Java Documentation – https://docs.oracle.com/en/java
- Very accurate and self explanatory
- Allows you to understand what each class, method and param does
- Allows you to see how to use every class, method or param
- Will be checked for (almost) every single line

/ First Java Program

Hello, World!

- Open IntelliJ IDEA and select File => New => Project…
- Check if Java is selected on the left-side section
- Check if Project JDK is set in the select menu on the right
- If not, click on the select menu and choose Add JDK… and navigate to the place where you previously moved JDK unarchived files, then click Next
- Check Create project from template
- Choose Command Line App, then click Next

Hello, World!

- Choose a Project name (e.g., 01_Hello)
- Choose a Project location (use one that you will use for the entire course)
- Choose a Base package name (more on this later; e.g., com.db.hello)
- Click Finish
- Your first Java program is here… but it does nothing

Hello, World!

```
package com.db.hello;

public class Main {

    public static void main(String[] args) {

        System.out.println("Hello, world!");

    }

}
```

Let’s Analyze the Code

- There is a **class** called Main
- It has a main **method**, which gets a parameter (more on this later)
- A Java program can run from a file that contains a main method and the filename is the same as the class containing this file
- Keywords to check on later: package, public, static…
- The single line of code: `System.out.println(“Hello, world!”)`
- Check documentation about System class, out object, println method
- Prints the `Hello, world!` string

/ Variables and Data Types

Variables

- Have a name
- Have a data type (primitive type or object)
- Are automatically assigned at a location memory
- Can be declared and initialized in the same line of code, or separately
- Syntax:

```
type var_name = value;
```

Variables

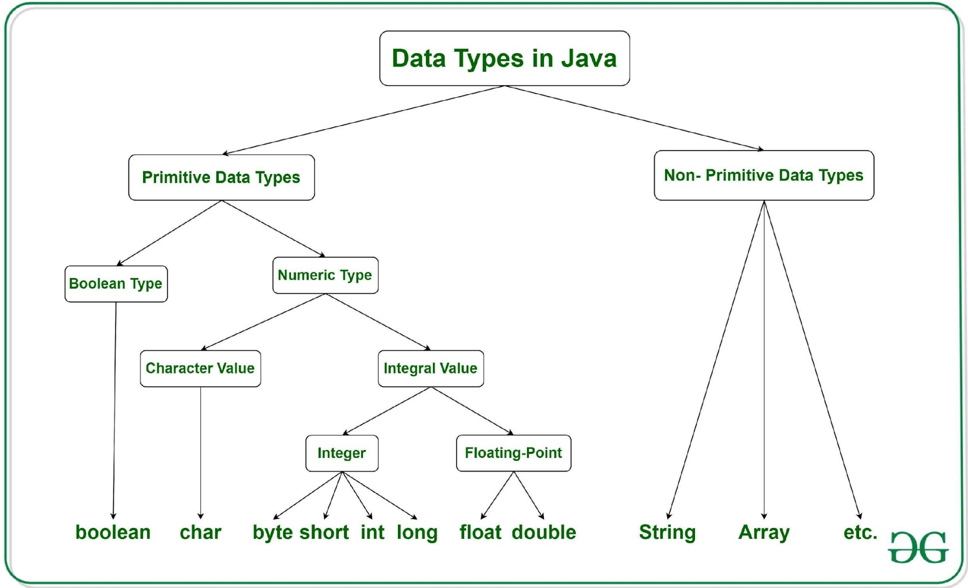
```
int a = 5; // declaration and initialization of integer primitive

float b; // declaration of float primitive

b = 2.4f; // initialization of float primitive

boolean notTrue = false; /* declaration and initialization of
boolean primitive */

String hiStr = “Hi!”; // declaration and initialization of a string

Object myObj = new Object(); /* declaration and allocation of memory
for a new object of type... Object*/
```

Primitive Types

- **int** – integers without decimals (e.g.: 14, -97), 4 bytes
- **float** – floating point numbers with decimals (e.g.: 3.14, -1.27), 4 bytes
- **double** – floating point numbers with higher precision (e.g.: 3.141592653589793), 8 bytes
- **char** – characters (e.g.: ‘x’, ‘Z’), 2 bytes
- **boolean** – values with two states: true or false, 1 byte
- Other primitive types: **byte**, **short**, **long**

Objects

- Everything in Java is an **object** (except: primitive types)
- **Object** is the parent of all (e.g., String; more on inheritance later)
- There are corresponding **wrapper classes** for primitive types: Integer for int, Float for float , etc.



Type Casting

- Performed when assigning a value of a primitive type to another primitive type
- Widening casting (performed automatically) – converts a more specific type to a more general type

```
double d = 4;
```

- Narrowing casting (performed explicitly by the programmer) – converts a more general type to a more specific type

```
int c = (int) 3.1f;
```

/ Operators

Operators

- Used to perform operations on variables and/or values
- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Bitwise operators

Arithmetic Operators

Operator	Description
+	Additive operator (also used for String concatenation)
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remainder operator
+	Unary plus operator; indicates positive value (numbers are positive without this, however)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean

Assignment Operators

Operator	Example	Equivalent to
=	a = b;	a = b;
+=	a += b;	a = a + b;
-=	a -= b;	a = a - b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;

Comparison Operators

Operator	Description	Example
==	Is Equal To	3 == 5 returns **false**
!=	Not Equal To	3 != 5 returns **true**
>	Greater Than	3 > 5 returns **false**
<	Less Than	3 < 5 returns **true**
>=	Greater Than or Equal To	3 >= 5 returns **false**
<=	Less Than or Equal To	3 <= 5 returns **true**

Logical Operators

Operator	Example	Meaning
&&	expression1 **&&** expression2	true only if both expression1 and expression2 are true
||	expression1 **||** expression2	true if either expression1 or expression2 is true
!	**!**expression	true if expression is false and vice versa

Bitwise Operators

Operator	Description
~	Bitwise Complement
<<	Left Shift
>>	Right Shift
>>>	Unsigned Right Shift
&	Bitwise AND
^	Bitwise exclusive OR
Operator	Description
~	Bitwise Complement

Operator Precedence

Operators	Precedence
explicit cast	(int) a
postfix	expr++ expr--
unary	++expr --expr +expr -expr ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :

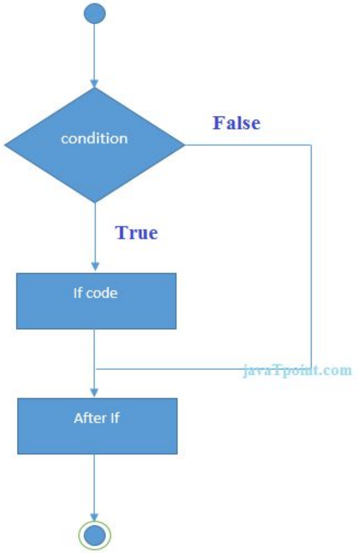
/ Control Statements

Control Statements

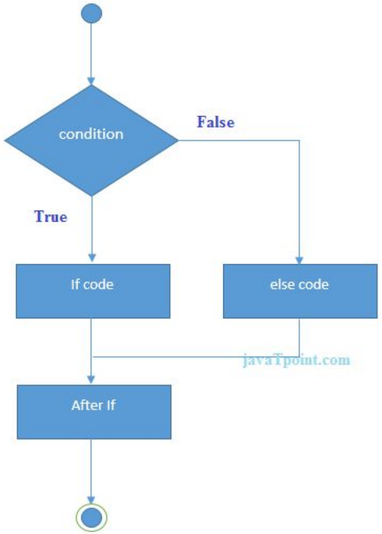
- Decision-making statements: **if** (if-else), **switch**
- Looping statements: **for**, **while**, **do-while**
- Branching statements: **break**, **continue**

IF Statement, IF-ELSE Statement

```
if(condition){
//code to be executed
}
```

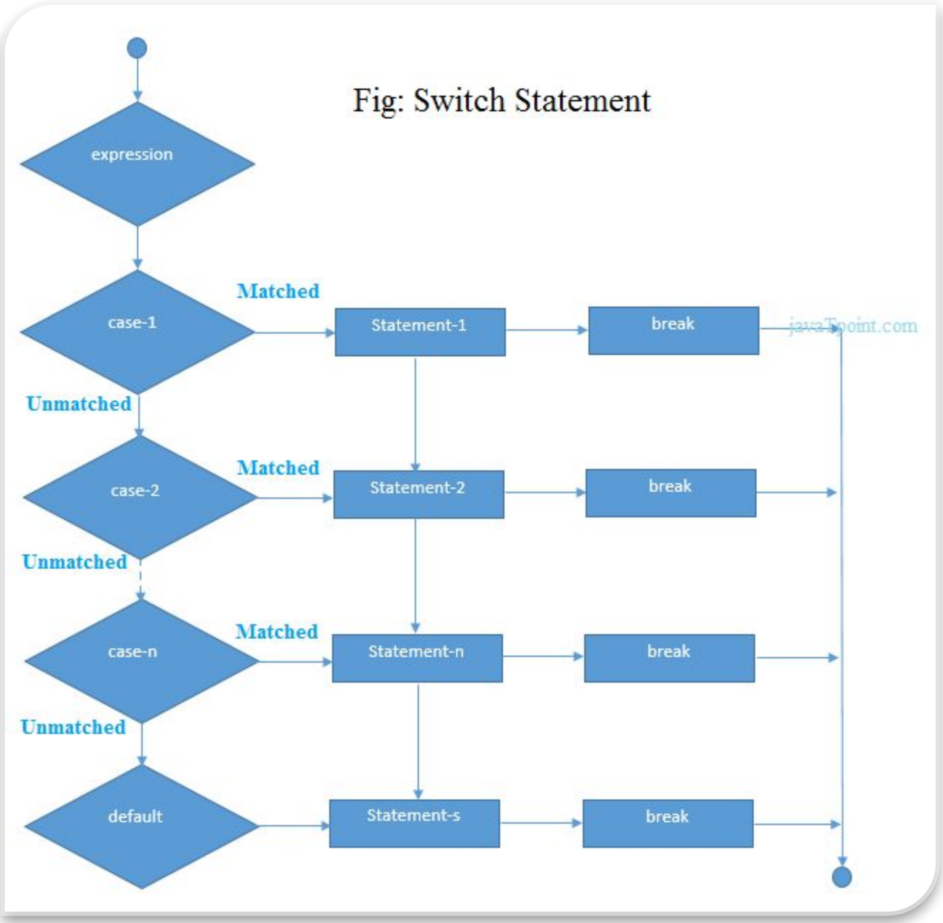


```
if(condition){
//code if condition is true
}else{
//code if condition is false
}
```



SWITCH Statement

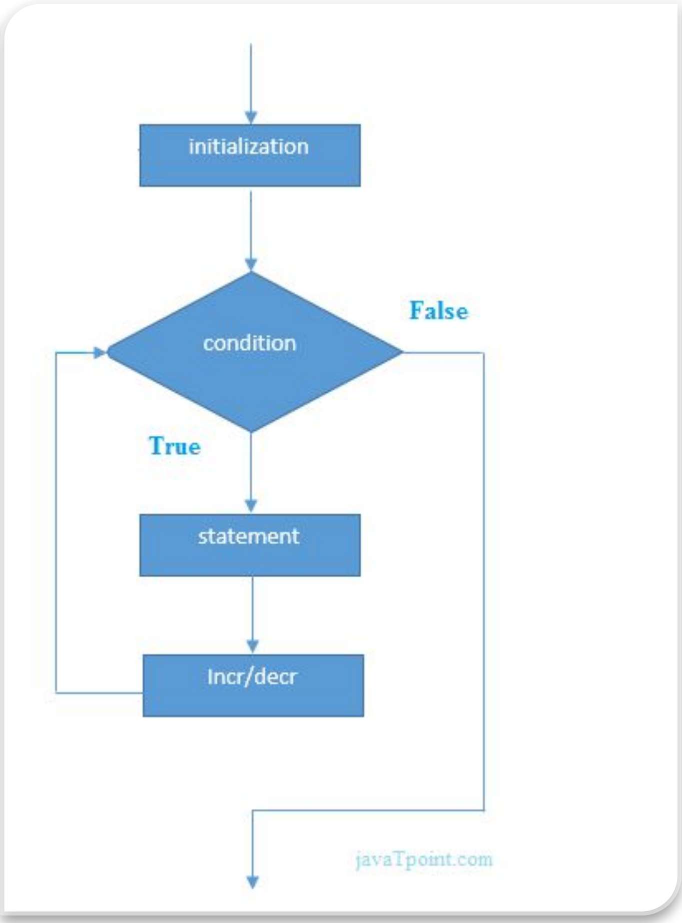
```
switch(expression){
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
......

default:
 code to be executed if all cases
are not matched;
}
```



FOR Statement

```
for(initialization; condition;
incr/decr){
//statement or code to be executed
}
```

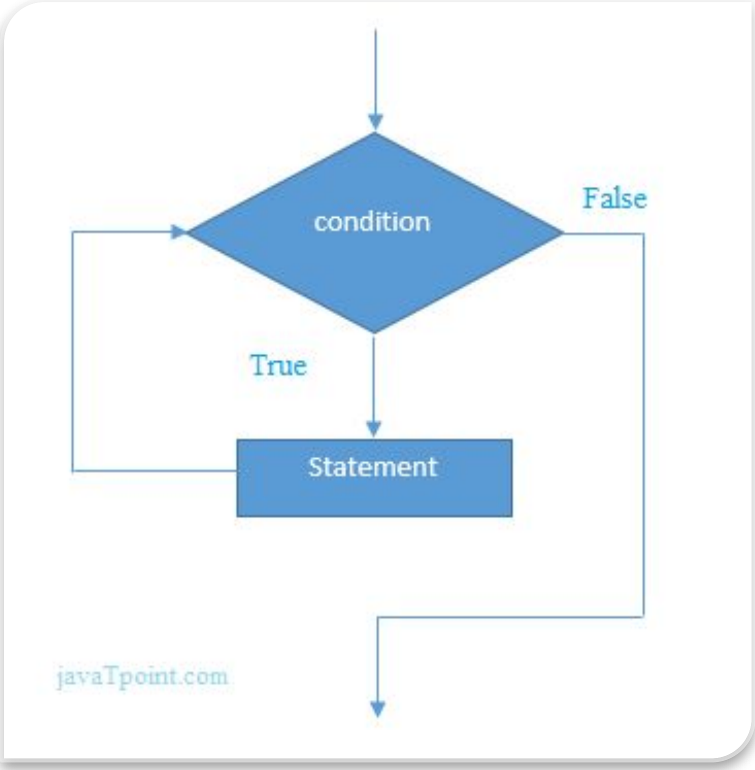
- Any statement (initialization, condition, incr/decr) may be omitted
- What happens if all of them are omitted?



WHILE Statement

```
while(condition){
//code to be executed
}
```

- Code may be never executed (if condition is not true initially)
- What happens if you run `while (true)`?

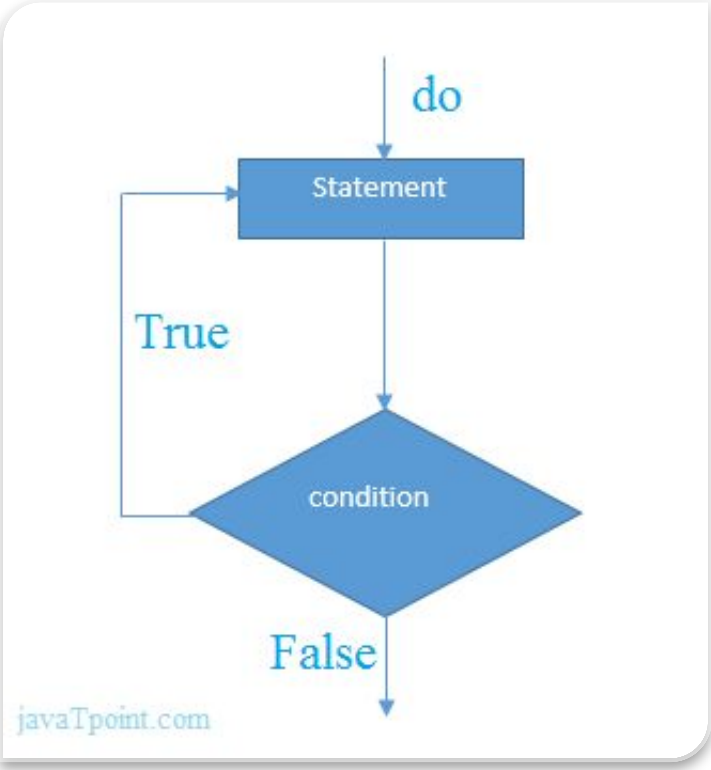


DO-WHILE Statement

```
do{
//code to be executed
} while(condition);
```

- Code will always be executed at least once (and as long as condition is true)

- What happens if you run `do … while (false)`?



BREAK

```
for(int i=1;i<=10;i++){
    if(i==5){
        //breaking the loop
        break;
    }
    System.out.println(i);
}
```

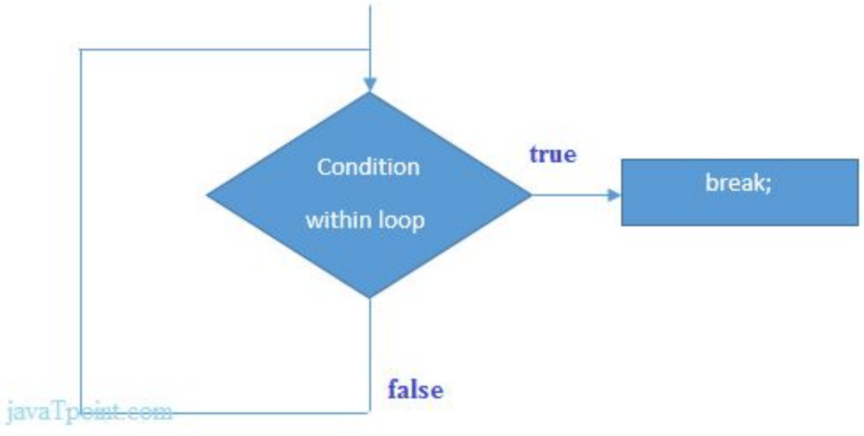


Figure: Flowchart of break statement

CONTINUE

```
for(int i=1; i<=10; i++){
    if(i==5){
        //using continue statement
        continue;

//it will skip the rest statement
    }
    System.out.println(i);
}
```

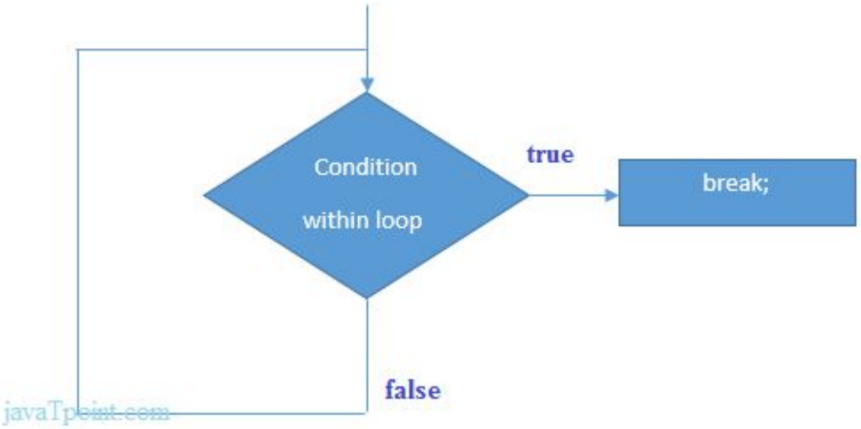


Figure: Flowchart of break statement

/ Practice, practice, practice

Hello, Compile!

Compile and run the *Hello, World!* Application from the terminal/command line. You need a compiler (you may use javac) and Java available via terminal to run the program.

Expected commands:

```
> javac MyHello.java
> java MyHello
```

Expected output:

```
Hello, World!
```

JDK Versions

Install different versions of JDK on your machine. Link them into your IDE (be it IntelliJ IDEA, Netbeans, or another one) and try to compile and run the *Hello, World!* application using different JDK versions.

Matrix Sum and Product (Optional)

Read from standard input (keyboard) an integral value n (n <= 10) and then two matrices of size n x n. Compute and print the sum matrix and the product matrix between the two.

Input sample:

```
n = 3;
a = 4 1 2
    3 4 6
    2 7 5
b = 9 1 2
    3 4 5
    7 1 2
```

Output sample

```
sum  = 13 2 4
       6 8 11
       9 8 7
prod = 53 10 17
       81 25 38
       74 35 49
```

Expressions

- Consider the following code:

```
boolean yes = true;
boolean no = false;
int loVal = -999;
int hiVal = 999;
double grade = 87.5;
double amount = 50.0;
String hello = "world";
```

- What is the result of the expressions listed in the table?

Expression	Result
yes == no || grade > amount	
amount == 40.0 || 50.0	
hiVal != loVal || loVal < 0	
true || hello.length() > 0	
hello.isEmpty() && yes	
grade <= 100 && !false	
!yes || no	
grade > 75 > amount	
amount <= hiVal && amount >= loVal	
no && !no || yes && !yes	

Type Casting

What do the following lines print?

```
public class Demo {
    public static void main(String[] args) {
        float a = 100.25f;
        long b = (long)a;
        System.out.println("value of a: "+a);
        System.out.println("value of b:"+b);

        int c = (int)b;
        System.out.println("value of c:"+c);

        byte d = (byte)c;
        System.out.println("value of d:"+d);
    }
}
```

Operators

What do the following lines print?

```
int a = 5;
System.out.println(a + -a - a++ % 10);
System.out.println(a - a + --a / 10);
for (int i = 2; i < 5;) {
        i++;
        a += a;
}
System.out.println(“a = “ + a);
```

/ Q&A

MOBILE/ACADEMY