# Basics Java Syntax

## 1. Overview

In programming, syntax refers to the set of rules which dictate how we, developers, should write our code. These rules ensure the code is structured in a way the compiler can understand.

It’s important to learn the fundamentals of Java syntax in the early stages, so we won’t have problems later on. If we know syntax well, then we can write clean, free-from-compiler-errors code.

Now, let’s explore the fundamental elements of Java syntax.

## 2. Statements and Semicolons

In Java, code is organized into statements. Simply put, a statement is a single line of code that performs a specific task.

Additionally, statements in Java always end with the semicolon (;). It tells the compiler where each

statement ends, and the next one is about to begin:

```
int x = 5; // This is a statement
System.out.println("Hello, world!"); // This is another statement
```

If you forget to put the semicolon at the end of the statement, you will encounter syntax errors. However, programming languages such as JavaScript of Python do not require semicolons.

## 3. Comments

Comments are parts of the code that do not execute. Furthermore, when the compiler compiles our code, it removes all the comments from our code.

Additionally, in Java, we can write comments in three different formats:

- “//” for a single.line comments
- “/* */” for multi-line comments
- “/** */” for Javadoc

```
// This is a single-line comment

/*
This is a
multi-line comment
*/

/**
* This is a Javadoc comment
*/
```

Comments serve as additional notes for our code. It can help make code more understandable for you and other developers working on the same project. They are especially helpful for methods that perform complex actions.

## 4. Variables and Data Types

We use variables to store and manipulate data.

Each variable has its data type and name:

```
int x = 5;
```

Here, the variable is of type int and has the name x. Additionally, we assign this variable a value of 5.

Furthermore, we distinguish primitive and reference data types.

## 5. Operators and Expressions

Operators allow you to perform operations on variables and values. Java supports various types of operators.

You can use these operators to create expressions, which are combinations of variables, values, and

operators that produce a result. For example:

```
int x = 5;
int y = 3;
int sum = x + y; // The expression x + y evaluates to 8
boolean isGreater = x > y; // The expression x > y evaluates to true
```

## 6. Blocks and Indentation

It is common to combine multiple statements that should be executed in certain scenarios. For this purpose, we use blocks of code. They are enclosed withing curly braces ({}).

Additionally, a block can contain one or more statements. We often use it when defining methods, conditional statements, and loops. On a class level, curly braces represent the beginning and the end of the class.

```
if (condition) { // This is a block of code
statement1;
statement2;
}
else { // Another block of code
statement3;
}
```

Furthermore, proper indentation and code format make it more readable and helps maintain code structure.

Moreover, Java ignores whitespace characters such as spaces, tabs, and line breaks, in our code, so we can use them to improve readability.

```
if (condition) { // This is a block of code statement1; statement2; } else { // Another block of code statement3; }
```

The code above is not as readable as the first code snippet.

## 7. Case Sensitivity

One of the features of Java is case sensitivity. It distinguishes uppercase and lowercase characters. For example, “beenary” and “Beenary” are considered to be different variables.

## 8. Reserved Keywords

Now, when naming variables, classes, and methods, we should be careful. Java has a set of reserved keywords that have special meanings for the compiler and thus, we can not use them as names in our code.

Here is the list of all reserved words in Java:

| | | |
|---|---|---|
| case | catch | char |
| continue | default | do |
| enum | extends | false |
| float | for | goto* |
| import | instanceof | int |
| native | new | null |
| protected | public | return |
| strictfp | super | switch |
| throw | throws | transient |
| void | volatile | while |

However, we can use them in different case formats. For instance, the class word is a reserved word in Java and we can not use it. But, the Class word is not. Since Java is case-sensitive, it differentiates those two words. Although it will not result in error, it still does not mean we should use it.

## 9. CamelCase Naming Convention

Additionally, it is a common practice to use the CamelCase naming convention for variables and other identifiers. This means that variable names start with a lowercase letter and, for subsequent words, capitalize the first letter of each word.

```
int myCoolVariable = 5;

void calculatePrice(){
    // ...
}
```

## 10. Key Takeaways

In this tutorial, we learned the basics of Java syntax. They represent the foundation of understanding Java programming language.

To sum up, each statement in Java must end with a semicolon. When naming our variables, we should think about the Camel Case naming convention and reserved words in Java. In addition, we should pay attention to code formatting in order to maintain readability.

Read More