surroundings. Java also allows for the use of functions written in other languages, such as C and C++, in Java projects. These are referred to as "native techniques." At runtime, these methods are dynamically connected.

BASIC SYNTAX

Each programming language has its own syntax. We'll go over the syntax for everything in Java that you should know before learning and familiarizing yourself with it.

The syntax refers to the format in which a Java program is written, and the commands used to compile and run it. It will be difficult for a programmer or learner to get the desired outputs from a program if they lack proper syntax knowledge.

What Is the Syntax of Java?

The collection of rules that determine how to construct and understand a Java program is referred to as Java syntax. Java's syntax is derived from C and C++. However, there are numerous differences, such as the lack of global variables in Java compared to C++.

In Java, the code belongs to objects and classes. To avoid programming errors, Java does not include some features such as operator overloading and the use of explicit pointers.

So let's start with java's basic syntax.[3]

Identifiers are the fundamental components of a Java program. Identifiers are used to give names to various parts

[3] https://www.baeldung.com/java-syntax, Baeldung

of a program, such as variables, objects, classes, methods, and arrays.

- **Objects:** Objects are made up of states and actions. A dog, for example, has states such as color, name, breed, and behavior such as wagging its tail, barking, and eating. A class's instance is an object.
- **Class:** A class is a pattern that describes the state that an object of its kind may support.
- **Methods:** The term "method" refers to a type of behavior. Many ways can be found in a single class. Methods are where logic is expressed, data is processed, and all actions are executed. Methods mean functions in oop
- **Instance Variables:** Each object has a collection of instance variables that are unique to it. The values supplied to these instance variables determine the state of an object.

First Java Program

Consider the following code, which prints the words Hello.

Example:

```
public class FirstJavaProgram {

   /* This is my first java program.
    * This will print 'Hello Everyone '
    */

   public static void main(String []
args) {
      System.out.println("Hello Everyone");
   }
}
```

Let's take a look at how to save the file, compile it, and run it. Please proceed with the steps below:

- Open notepad and add the code as above.
- Save the file as: FirstJavaProgram.java.
- Open a command prompt window and go to the directory where you saved the class. Assume it's C:\.
- Type "javac MyFirstJavaProgram.java" and press enter to compile your code. If there are no errors in code, the command prompt will take you to the following line (Assumption: The path variable is set).
- Now, type "java FirstJavaProgram" to run your program and we will be able to see "Hello Everyone" printed on the window.

Syntax for Beginners

It's critical to remember the following considerations while working with Java applications.

- Java is case sensitive, which implies that the identifiers Hello and hello have distinct meanings in Java.
- The initial letter of each class name should be capitalized. If a class name is made up of many words, the initial letter of each inner word should be in upper case.

  Example: FirstJavaClass is an example of a class.
- Method names should all begin with a lower case letter. If the method's name is made up of many words, the initial letter of each inner word should be capitalized.

Example: public void myMethodName is an example of a public void method ()

- The name of the program file should be identical to the name of the class.

  When saving the file, use the class name and attach ".java" to the end of the name (remember, Java is case sensitive) (if the file name and the class name do not match, your program will not compile).

  If the file does not include a public class, the file name may differ from the class name. It is also not required that the file have a public class.

  Example: Assume the class is called "FirstJavaProgram." Then, save the program as "FirstJavaProgram.java."

- Public static void main(String args[]): The main() method, which is a required part of any Java program, starts the processing of the program.

Identifiers in Java

The basic building elements of a Java program are identifiers. Identifiers are used to give names to various components of a program, such as variables, objects, classes, methods, arrays, etc.

The following are the guidelines for naming identifiers in Java:

1. Alphabets, numbers, underscore (_), and dollar ($) sign characters can all be used in identifiers.
2. They can't be a Java reserved term or keyword like true, false, while, and so on.

3. Identifiers can't start with a digit.
4. Identifiers can be as long as you want them to be.
5. Because Java is case sensitive, identifiers in uppercase and lowercase are handled differently.

Keywords in Java

Keywords are reserved words in Java that provide the compiler a particular interpretation. The keywords can't be used as regular identifier names; they must be used for a specific reason.[4]

The following are some Java keywords:

- **Abstract:** The abstract keyword in Java is used to declare an abstract class. The interface can be implemented using an abstract class. Both abstract and non-abstract methods may be used.
- **Boolean:** The boolean keyword in Java is used to define a Boolean type variable. It can only store True and False values.
- **Break:** The break keyword in Java is used to end a loop or switch expressions. It interrupts the program's current flow when certain criteria are met.
- **Byte:** The byte keyword in Java is used to create a variable containing 8-bit data values.
- **Case:** The case keyword in Java is used with switch statements to mark text blocks.

[4] https://www.javatpoint.com/java-keywords, javaTpoint

- **Catch:** The catch keyword in Java is used to capture exceptions thrown by try statements. It can only be used after the try block.
- **Char:** The char keyword in Java is used to create a variable containing unsigned 16-bit Unicode characters.
- **Class:** To declare a class in Java, use the class keyword.
- **Continue:** The continue keyword in Java is used to keep the loop going. It continues the program's current flow while skipping the remaining code at the given circumstance.
- **Default:** The default block of code in a switch statement is specified by the Java default keyword.
- **Do:** The do keyword in Java is used to identify a loop in the control statement. It can repeat a section of the program several times.
- **Double:** The double keyword in Java is used to create a variable containing a 64-bit floating-point value.
- **Otherwise:** In the if statement, the else keyword in Java is used to represent alternate branches.
- **Enum:** The enum keyword in Java is used to specify a set of fixed constants. Private or default constructors are always used in enum constructors.
- **Extend:** The extend keyword in Java is used to show that a class is inherited from another class or interface.
- **Final:** The final keyword in Java is used to denote that a variable has a constant value. It's used in

conjunction with a variable to prevent the user from changing the variable's value.

- **Finally:** The finally keyword in Java denotes a code block in a try-catch structure. Whether or not an exception is handled, this block is always performed.
- **Float:** The float keyword in Java is used to create a variable containing a 32-bit floating-point integer.
- **For:** The for keyword in Java is used to begin a for loop. When a condition is met, it is used to execute a set of instructions/functions repeatedly. If the number of iterations is fixed, the for loop is preferred.
- **If:** The if keyword in Java is used to test a condition. If the condition is true, the if block is executed.
- **Implements:** The term implements are used in Java to implement an interface.
- **Keyword:** The import keyword in Java makes classes and interfaces available to the current source code.
- **Instanceof:** The instanceof keyword in Java determines if an object is an instance of a particular class or implements an interface.
- **Int:** The int keyword in Java is used to declare a variable that can store a signed 32-bit integer.
- **Interface:** The interface keyword in Java is used to declare an interface. It is limited to abstract techniques.
- **Long:** The long keyword in Java is used to specify a variable that may store a 64-bit integer.

- **Native:** The Java native keyword indicates that a method is implemented using JNI (Java Native Interface) in native code.
- **New:** The new keyword in Java is used to create new objects.
- **Null:** The null keyword in Java is used to indicate that a reference refers to nothing. It gets rid of the trash value.
- **Package:** The term package in Java is used to declare a Java package that contains the classes.
- **Private:** The private keyword in Java is an access modifier. It's used to say that a method or variable may only be accessible in the class where it's declared.
- **Protected:** The protected keyword in Java is an access modifier. It can be accessed both within and outside the package, but only through inheritance. It isn't possible to use it with the class.
- **Public:** The public keyword in Java is an access modifier. It's a phrase that means anything may be found anyplace. Among all the modifiers, it has the broadest use.
- **Return:** When a method's execution is complete, the Java return keyword is used to exit the method.
- **Short:** The Java short keyword is used to declare a variable with a 16-bit integer capacity.
- **Static:** The static keyword in Java indicates that a variable or function belongs to a class. In Java, the static keyword is mainly used to control memory.

- **Strictfp:** To guarantee portability, Java strictfp is used to limit floating-point calculations.
- **Super:** The super keyword in Java is a reference variable that refers to parent class objects. It may be used to call the method of the immediate parent class.
- **Switch:** The switch keyword in Java includes a switch statement that executes code based on the test value. The switch statement compares several values to see whether they are equal.
- **Synchronised:** In multithreaded programming, the synchronized keyword is used to identify the critical portions or functions.
- **This:** In a method Java, this keyword can be used to refer to the current object.
- **Throw:** The throw keyword in Java is used to throw an exception explicitly. Throwing custom exceptions is the most common usage of the throw keyword. After that, there is an example.
- **Throws:** The throws keyword in Java is used to declare an exception. Throws can be used to propagate checked exceptions.
- **Transient:** In serialization, the Java temporary keyword is utilized. Any data member that is marked as transitory will not be serialized.
- **Try:** The try keyword in Java is used to begin a block of code that will be checked for errors. Either a catch or a final block must come after the try block.

- **Void:** The void keyword in Java is used to indicate that a method has no return value.
- **Volatile:** In Java, the volatile keyword is used to indicate that a variable may change asynchronously.
- **While:** The while keyword in Java is used to initiate a while loop. This loop repeats a section of the program several times. The while loop is recommended if the number of iterations is not fixed.

Modifiers in Java

Modifiers may be used to alter classes, methods, and other objects, much as in other languages. Modifiers are divided into two groups:

1. Default, public, protected, and private are the access modifiers.
2. Final, abstract, and strictfp are non-access modifiers.

Variables in Java

The kinds of variables in Java are as follows:

- Local Variables
- Class Variables (Static Variables)
- Instance Variable (Non-static Variables)

Enums in Java

In Java 5.0, enums were introduced. Enums limit a variable's value to one of a few preset options. Enums are the

values in this enumerated list. It is possible to decrease the number of defects in your code by using enums.

For example, if we consider an application for a juice business, the glass sizes may be limited to small, medium, and large. This would ensure that no one could order anything other than a small, medium, or big size.

Literals in Java

Constants or data objects with fixed values are referred to as literals in Java. In Java, there are several sorts of Literals. Numeric, Floating, Character, Strings, Boolean, and Null are the types. They're also divided into subcategories. Let's take a look at each one separately:

1. **Numeric:** Numbers are represented via numeric literals. In Java, there are four different forms of numeric literals:
    i. **Integer Literals:** Integer literals are integers in base 10 that are entire numbers without any fractional parts. For instance, 108.
    ii. **Binary Literals:** A binary literal is an integer with a base of two, for instance, 011.
    iii. **Octal Literals:** The numbers with base 8 are known as octal literals. They can't have the numbers 8 or 9 in them. For instance, 565.
    iv. **Hexadecimal Literals:** Hexadecimal literals are integers that have a base of 16 characters. They can include numbers ranging from 0 to 9 as well as alphabets ranging from A to F. For instance, 24D6.
2. **Floating-point:** Floating-point literals are also known as real literals. Only a fractional point is used in the

floating-point literal to specify numeric values (.). They can take the shape of a fractional or exponential function. For instance, -15.6, 1.876, 11.4D08.

3. **Character:** Character literals deal with characters contained in a single quote. Within single quotes ‘ ’ they can only contain one character. They are divided into the following categories:
   i. **Single quoted character:** A single-quoted character encloses all uni-length characters in single quotes. For example, ‘b’, ‘p’, ‘S’.
   ii. **Escape Sequences:** These are the characters that appear after the backslash and perform a specific purpose on the screen, such as tabs, newlines, and so on. For example, ‘b,’ ‘n,’ ‘t’.
   iii. **Unicode Representation:** We may express it by putting the character’s relevant Unicode value after the letter ‘u’. For instance, ‘u0057’.
4. **Boolean:** True and false are the two possible values for a boolean literal. ASCII letters are used to create these values. True or false, for example.
5. **String:** String-literals are multi-character constants enclosed in double quotes “.” For example, “javaclass,” “Hello,” “\cde,” and so on.

Comments in Java

When programmers add documentation to a method or a line specified within the program, they can use comments. While compiling, the compiler does not read the comments and ignores them. The comments improve the program’s readability and comprehension.