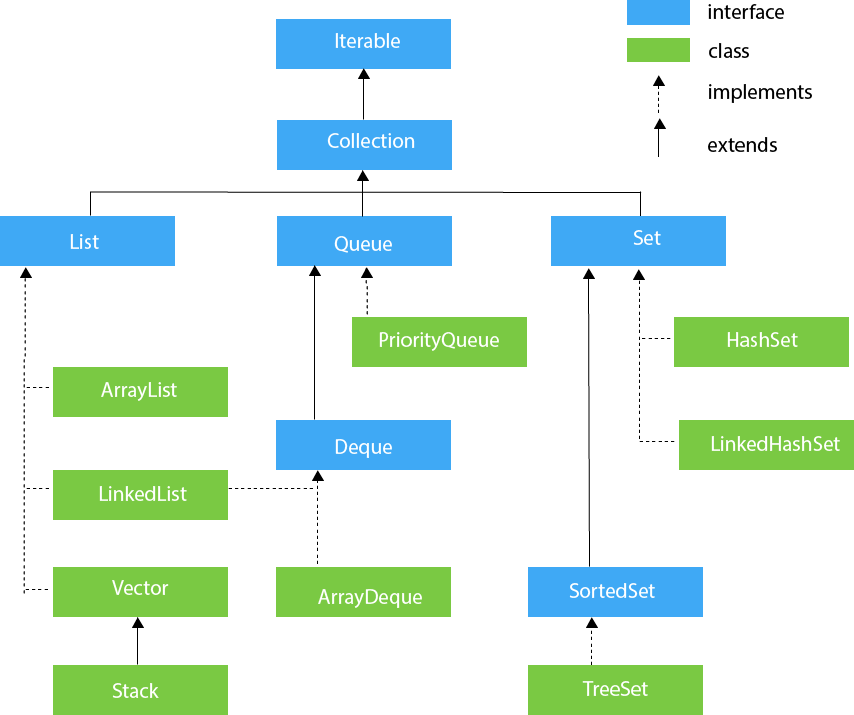
## List of Java Keywords ,Java 8,7,⅘ features ,Annotation, spring MVC architecture

## Java Keywords:

1. [**abstract**](https://www.javatpoint.com/abstract-keyword-in-java)**:** Java abstract keyword is used to declare an abstract class. An abstract class can provide the implementation of the interface. It can have abstract and non-abstract methods.
2. [**boolean:**](https://www.javatpoint.com/boolean-keyword-in-java) Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.
3. [**break**](https://www.javatpoint.com/java-break)**:** Java break keyword is used to break the loop or switch statement. It breaks the current flow of the program at specified conditions.
4. [**byte**](https://www.javatpoint.com/byte-keyword-in-java)**:** Java byte keyword is used to declare a variable that can hold 8-bit data values.
5. [**case**](https://www.javatpoint.com/case-keyword-in-java)**:** Java case keyword is used with the switch statements to mark blocks of text.
6. [**catch**](https://www.javatpoint.com/try-catch-block)**:** Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.
7. [**char**](https://www.javatpoint.com/char-keyword-in-java)**:** Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters
8. [**class**](https://www.javatpoint.com/class-keyword-in-java)**:** Java class keyword is used to declare a class.
9. [**continue**](https://www.javatpoint.com/java-continue)**:** Java continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
10. [**default**](https://www.javatpoint.com/default-keyword-in-java)**:** Java default keyword is used to specify the default block of code in a switch statement.
11. [**do**](https://www.javatpoint.com/java-do-while-loop)**:** Java do keyword is used in the control statement to declare a loop. It can iterate a part of the program several times.
12. [**double**](https://www.javatpoint.com/double-keyword-in-java)**:** Java double keyword is used to declare a variable that can hold 64-bit floating-point number.
13. [**else**](https://www.javatpoint.com/java-if-else)**:** Java else keyword is used to indicate the alternative branches in an if statement.
14. [**enum**](https://www.javatpoint.com/enum-in-java)**:** Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.
15. [**extends**](https://www.javatpoint.com/inheritance-in-java)**:** Java extends keyword is used to indicate that a class is derived from another class or interface.
16. [**final**](https://www.javatpoint.com/final-keyword)**:** Java final keyword is used to indicate that a variable holds a constant value. It is used with a variable. It is used to restrict the user from updating the value of the variable.
17. [**finally**](https://www.javatpoint.com/finally-block-in-exception-handling)**:** Java finally keyword indicates a block of code in a try-catch structure. This block is always executed whether an exception is handled or not.
18. [**float**](https://www.javatpoint.com/float-keyword-in-java)**:** Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.
19. [**for**](https://www.javatpoint.com/java-for-loop)**:** Java for keyword is used to start a for loop. It is used to execute a set of instructions/functions repeatedly when some condition becomes true. If the number of iteration is fixed, it is recommended to use for loop.
20. [**if**](https://www.javatpoint.com/java-if-else)**:** Java if keyword tests the condition. It executes the if block if the condition is true.
21. [**implements**](https://www.javatpoint.com/interface-in-java)**:** Java implements keyword is used to implement an interface.
22. [**import**](https://www.javatpoint.com/package)**:** Java import keyword makes classes and interfaces available and accessible to the current source code.
23. [**instanceof**](https://www.javatpoint.com/downcasting-with-instanceof-operator)**:** Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.
24. [**int**](https://www.javatpoint.com/int-keyword-in-java)**:** Java int keyword is used to declare a variable that can hold a 32-bit signed integer.
25. [**interface**](https://www.javatpoint.com/interface-in-java)**:** Java interface keyword is used to declare an interface. It can have only abstract methods.
26. [**long**](https://www.javatpoint.com/long-keyword-in-java)**:** Java long keyword is used to declare a variable that can hold a 64-bit integer.
27. **native:** Java native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface).
28. [**new**](https://www.javatpoint.com/new-keyword-in-java)**:** Java new keyword is used to create new objects.
29. [**null**](https://www.javatpoint.com/null-keyword-in-java)**:** Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.
30. [**package**](https://www.javatpoint.com/package)**:** Java package keyword is used to declare a Java package that includes the classes.
31. [**private**](https://www.javatpoint.com/private-keyword-in-java)**:** Java private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.
32. [**protected**](https://www.javatpoint.com/protected-keyword-in-java)**:** Java protected keyword is an access modifier. It can be accessible within the package and outside the package but through inheritance only. It can't be applied with the class.
33. [**public**](https://www.javatpoint.com/public-keyword-in-java)**:** Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.
34. [**return**](https://www.javatpoint.com/return-keyword-in-java)**:** Java return keyword is used to return from a method when its execution is complete.
35. [**short**](https://www.javatpoint.com/short-keyword-in-java)**:** Java short keyword is used to declare a variable that can hold a 16-bit integer.
36. [**static**](https://www.javatpoint.com/static-keyword-in-java)**:** Java static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is mainly used for memory management.
37. [**strictfp**](https://www.javatpoint.com/strictfp-keyword)**:** Java strictfp is used to restrict the floating-point calculations to ensure portability.
38. [**super**](https://www.javatpoint.com/super-keyword)**:** Java super keyword is a reference variable that is used to refer to parent class objects. It can be used to invoke the immediate parent class method.
39. [**switch**](https://www.javatpoint.com/java-switch)**:** The Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.
40. [**synchronized**](https://www.javatpoint.com/synchronization-in-java)**:** Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.
41. [**this**](https://www.javatpoint.com/this-keyword)**:** Java this keyword can be used to refer the current object in a method or constructor.
42. [**throw**](https://www.javatpoint.com/throw-keyword)**:** The Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exceptions. It is followed by an instance.
43. [**throws**](https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws)**:** The Java throws keyword is used to declare an exception. Checked exceptions can be propagated with throws.
44. [**transient**](https://www.javatpoint.com/transient-keyword)**:** Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.
45. [**try**](https://www.javatpoint.com/try-catch-block)**:** Java try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.
46. **void:** Java void keyword is used to specify that a method does not have a return value.
47. [**volatile**](https://www.javatpoint.com/volatile-keyword-in-java)**:** Java volatile keyword is used to indicate that a variable may change asynchronously.
48. [**while**](https://www.javatpoint.com/java-while-loop)**:** Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iteration is not fixed, it is recommended to use the while loop.

### Hierarchy of Collection Framework

The **java.util** package contains all the [classes](https://www.javatpoint.com/object-and-class-in-java) and [interfaces](https://www.javatpoint.com/interface-in-java) for the Collection framework.



#### 

#### [**JavaSE 8 Features**](https://www.javatpoint.com/java-8-features)

* [Java 8 Date/Time API (Java 8)](https://www.javatpoint.com/java-date)
* [Lambda Expressions (Java 8)](https://www.javatpoint.com/java-lambda-expressions)
* [Method References (Java 8)](https://www.javatpoint.com/java-8-method-reference)
* [Functional Interfaces (Java 8)](https://www.javatpoint.com/java-8-functional-interfaces)
* [Stream (Java 8)](https://www.javatpoint.com/java-8-stream)
* [Base64 Encode Decode (Java 8)](https://www.javatpoint.com/java-base64-encode-decode)
* [Default Methods (Java 8)](https://www.javatpoint.com/java-default-methods)
* [forEach method(Java 8)](https://www.javatpoint.com/java-8-foreach)
* [Collectors(Java 8)](https://www.javatpoint.com/java-8-collectors)
* [StringJoiner(Java 8)](https://www.javatpoint.com/java-stringjoiner)
* [Optional class (Java 8)](https://www.javatpoint.com/java-8-optional)
* [Nashorn JavaScript (Java 8)](https://www.javatpoint.com/java-nashorn)
* [Parallel Array Sorting (Java 8)](https://www.javatpoint.com/java-8-parallel-array-sorting)
* [Type Inference (Java 8)](https://www.javatpoint.com/java-8-type-inference)
* [Method Parameter Reflection (Java 8)](https://www.javatpoint.com/java-8-method-parameter-reflection)
* [Type annotations and repeating annotations (Java 8)](https://www.javatpoint.com/java-8-type-annotations-and-repeating-annotations)
* [Java JDBC Improvements (Java 8)](https://www.javatpoint.com/java-8-jdbc-improvements)
* Java IO Improvement (Java 8)
* Java Concurrency Improvement (Java 8)

## **Java 7 Features**

* [Binary Literals](https://www.javatpoint.com/java-binary-literals)
* [Switch with String](https://www.javatpoint.com/java-switch-with-string)
* [Java 7 Multi Catch](https://www.javatpoint.com/java-7-catch-multiple-exceptions)
* [Try with Resources](https://www.javatpoint.com/java-try-with-resources)
* [Type Inference](https://www.javatpoint.com/java-type-inference-for-generics)
* [Numeric Literals](https://www.javatpoint.com/java-numeric-literals-with-underscore)
* [Java 7 JDBC](https://www.javatpoint.com/java-7-jdbc-improvement)

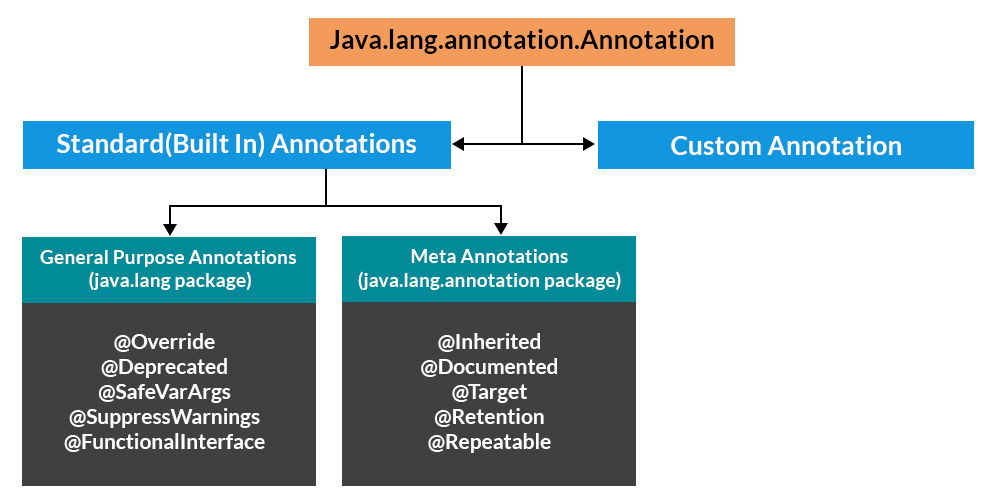
## **Java 4/5 Features**

* [Java Assertion](https://www.javatpoint.com/assertion-in-java)
* [Java For-each Loop](https://www.javatpoint.com/for-each-loop)
* [Java Varargs](https://www.javatpoint.com/varargs)
* [Java Static Import](https://www.javatpoint.com/static-import-in-java)
* [Java Autoboxing](https://www.javatpoint.com/autoboxing-and-unboxing)
* [Java Enums](https://www.javatpoint.com/enum-in-java)
* [Java Annotations](https://www.javatpoint.com/java-annotation)
* [Java Generic](https://www.javatpoint.com/generics-in-java)

# 

# **Annotations in Java**

# Annotations are used to provide supplemental information about a program.

* Annotations start with ‘**@**’.
* Annotations do not change the action of a compiled program.
* Annotations help to associate *metadata* (information) to the program elements i.e. instance variables, constructors, methods, classes, etc.
* Annotations are not pure comments as they can change the way a program is treated by the compiler. See below code for example.****

1. **@Deprecated** - A program element annotated @Deprecated is one that programmers are discouraged from using, typically because it is dangerous, or because a better alternative exists.
2. **@Override** - Indicates that a method declaration is intended to override a method declaration in a supertype.
3. **@FunctionalInterface** - An informative annotation type used to indicate that an interface type declaration is intended to be a functional interface as defined by the Java Language Specification.
4. **@SafeVarargs** - A programmer assertion that the body of the annotated method or constructor does not perform potentially unsafe operations on its **varargs** parameter.
5. **@SuppressWarnings** - Indicates that the named compiler warnings should be suppressed in the annotated element (and in all program elements contained in the annotated element).

