* **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.
* **Class** − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.
* **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
* **Instance Variables** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

Basic Syntax:

* **Case Sensitivity** − Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.
* **Class Names** − For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

**Example:** *class MyFirstJavaClass*

* **Method Names** − All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

**Example:** *public void myMethodName()*

* **Program File Name** − Name of the program file should exactly match the class name.

When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match, your program will not compile).

But please make a note that in case you do not have a public class present in the file then file name can be different than class name. It is also not mandatory to have a public class in the file.

**Example:** Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as *'MyFirstJavaProgram.java'*

* **public static void main(String args[])** − Java program processing starts from the main() method which is a mandatory part of every Java program.
* **Access Modifiers** − default, public , protected, private
* **Non-access Modifiers** − final, abstract, strictfp

Java Variables

* Local Variables
* Class Variables (Static Variables)
* Instance Variables (Non-static Variables)
* **Local variables** − Variables defined inside methods, constructors or blocks are called **local** variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
* **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
* **Class variables** − Class variables are variables declared within a class, outside any method, with the static keyword.

Java keywords:

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | assert | boolean | break |
| byte | case | catch | char |
| class | const | continue | default |
| do | double | else | enum |
| extends | final | finally | float |
| for | goto | if | implements |
| import | instanceof | int | interface |
| long | native | new | package |
| private | protected | public | return |
| short | static | strictfp | super |
| switch | synchronized | this | throw |
| throws | transient | try | void |
| volatile | while |  |  |

[abstract](https://en.wikipedia.org/wiki/Abstract_type)

A method with no definition must be declared as abstract and the class containing it must be declared as abstract. Abstract classes cannot be instantiated. Abstract methods must be implemented in the sub classes. The abstract keyword cannot be used with variables or constructors. Note that an abstract class isn't required to have an abstract method at all.

[assert](https://en.wikipedia.org/wiki/Assertion_(software_development)) (added in [J2SE 1.4](https://en.wikipedia.org/wiki/J2SE_1.4))

Assert describes a predicate (a true–false statement) placed in a Java program to indicate that the developer thinks that the predicate is always true at that place. If an assertion evaluates to false at run-time, an assertion failure results, which typically causes execution to abort. Optionally enable by ClassLoader method.

[boolean](https://en.wikipedia.org/wiki/Boolean_data_type)

Defines a boolean variable for the values "true" or "false" only. By default, the value of boolean primitive type is false. This keyword is also used to declare that a method returns a value of the primitive type [boolean](https://en.wikipedia.org/wiki/Boolean_data_type).

[break](https://en.wikipedia.org/wiki/Break_statement)

Used to end the execution in the current loop body.

[byte](https://en.wikipedia.org/wiki/Byte)

The byte keyword is used to declare a field that can hold an 8-bit signed two's complement integer. This keyword is also used to declare that a method returns a value of the primitive type byte.

[case](https://en.wikipedia.org/wiki/Switch_statement)

A statement in the [switch](https://en.wikipedia.org/wiki/List_of_Java_keywords#switch) block can be labeled with one or more [case](https://en.wikipedia.org/wiki/List_of_Java_keywords#case) or [default](https://en.wikipedia.org/wiki/List_of_Java_keywords#default) labels. The [switch](https://en.wikipedia.org/wiki/List_of_Java_keywords#switch) statement evaluates its expression, then executes all statements that follow the matching [case](https://en.wikipedia.org/wiki/List_of_Java_keywords#case) label; see [switch](https://en.wikipedia.org/wiki/List_of_Java_keywords#switch).

[catch](https://en.wikipedia.org/wiki/Exception_handling_syntax#Java)

Used in conjunction with a try block and an optional finally block. The statements in the catch block specify what to do if a specific type of exception is thrown by the try block.

[char](https://en.wikipedia.org/wiki/Character_(computing))

Defines a character variable capable of holding any character of the java source file's character set.

[class](https://en.wikipedia.org/wiki/Class_(computer_science)#Java)

A type that defines the implementation of a particular kind of object. A class definition defines [instance](https://en.wikipedia.org/wiki/Object_(computer_science)) and class [fields](https://en.wikipedia.org/wiki/Field_(computer_science)), [methods](https://en.wikipedia.org/wiki/Method_(computer_science)), and [inner classes](https://en.wikipedia.org/wiki/Inner_class) as well as specifying the [interfaces](https://en.wikipedia.org/wiki/Interface_(computer_science)) the class implements and the immediate [superclass](https://en.wikipedia.org/wiki/Superclass_(computer_science)) of the class. If the superclass is not explicitly specified, the superclass is implicitly [Object](https://docs.oracle.com/javase/10/docs/api/java/lang/Object.html). The class keyword can also be used in the form Class**.class** to get a Class object without needing an instance of that class. For example, **String.class** can be used instead of doing **new String().getClass()**.

[const](https://en.wikipedia.org/wiki/Const)

Unused but reserved.

[continue](https://en.wikipedia.org/wiki/Continue_(Java))

Used to resume program execution at the end of the current loop body. If followed by a label, continue resumes execution at the end of the enclosing labeled loop body.

default

The default keyword can optionally be used in a [switch statement](https://en.wikipedia.org/wiki/Switch_statement) to label a block of statements to be executed if no case matches the specified value; see [*switch*](https://en.wikipedia.org/wiki/List_of_Java_keywords#switch).[[8]](https://en.wikipedia.org/wiki/List_of_Java_keywords#cite_note-switch-8)[[9]](https://en.wikipedia.org/wiki/List_of_Java_keywords#cite_note-FOOTNOTEFlanagan200546-48-9) Alternatively, the default keyword can also be used to declare default values in a [Java annotation](https://en.wikipedia.org/wiki/Java_annotation). From Java 8 onwards, the default keyword can be used to allow an interface to provide an implementation of a method.

[do](https://en.wikipedia.org/wiki/Do_while_loop)

The do keyword is used in conjunction with [while](https://en.wikipedia.org/wiki/List_of_Java_keywords#while) to create a [do-while loop](https://en.wikipedia.org/wiki/Do-while_loop), which executes a block of statements associated with the loop and then tests a boolean expression associated with the while. If the expression evaluates to true, the block is executed again; this continues until the expression evaluates to false.[[10]](https://en.wikipedia.org/wiki/List_of_Java_keywords#cite_note-do-while-10)[[11]](https://en.wikipedia.org/wiki/List_of_Java_keywords#cite_note-FOOTNOTEFlanagan200548-49-11)

[double](https://en.wikipedia.org/wiki/Double_precision)

The double keyword is used to declare a variable that can hold a 64-bit [double precision](https://en.wikipedia.org/wiki/Double_precision) [IEEE 754](https://en.wikipedia.org/wiki/IEEE_754) [floating-point number](https://en.wikipedia.org/wiki/Floating-point_number). This keyword is also used to declare that a method returns a value of the primitive type double.

[else](https://en.wikipedia.org/wiki/Conditional_(programming))

The else keyword is used in conjunction with [if](https://en.wikipedia.org/wiki/List_of_Java_keywords#if) to create an [if-else statement](https://en.wikipedia.org/wiki/Conditional_(programming)), which tests a [boolean expression](https://en.wikipedia.org/wiki/Boolean_expression); if the expression evaluates to true, the block of statements associated with the if are evaluated; if it evaluates to false, the block of statements associated with the else are evaluated.

[enum](https://en.wikipedia.org/wiki/Enumerated_type) (added in [J2SE 5.0](https://en.wikipedia.org/wiki/J2SE_5.0))

A Java keyword used to declare an [enumerated type](https://en.wikipedia.org/wiki/Enumerated_type). Enumerations extend the base class [Enum](https://docs.oracle.com/javase/10/docs/api/java/lang/Enum.html).

[extends](https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming))

Used in a class declaration to specify the superclass; used in an interface declaration to specify one or more superinterfaces. Class X extends class Y to add functionality, either by adding fields or methods to class Y, or by overriding methods of class Y. An interface Z extends one or more interfaces by adding methods. Class X is said to be a subclass of class Y; Interface Z is said to be a subinterface of the interfaces it extends.

Also used to specify an upper bound on a type parameter in Generics.

[final](https://en.wikipedia.org/wiki/Final_(Java))

Define an entity once that cannot be changed nor derived from later. More specifically: a final class cannot be subclassed, a final method cannot be overridden, and a final variable can occur at most once as a left-hand expression on an executed command. All methods in a final class are implicitly final.

[finally](https://en.wikipedia.org/wiki/Exception_handling_syntax#Java)

Used to define a block of statements for a block defined previously by the try keyword. The finally block is executed after execution exits the try block and any associated catch clauses regardless of whether an exception was thrown or caught, or execution left method in the middle of the try or catch blocks using the return keyword.

[float](https://en.wikipedia.org/wiki/Single_precision)

The float keyword is used to declare a variable that can hold a 32-bit [single precision](https://en.wikipedia.org/wiki/Single_precision) IEEE 754 floating-point number. This keyword is also used to declare that a method returns a value of the primitive type float.

[for](https://en.wikipedia.org/wiki/For_loop)

The for keyword is used to create a [for loop](https://en.wikipedia.org/wiki/For_loop), which specifies a variable initialization, a [boolean expression](https://en.wikipedia.org/wiki/Boolean_expression), and an incrementation. The variable initialization is performed first, and then the boolean expression is evaluated. If the expression evaluates to true, the block of statements associated with the loop are executed, and then the incrementation is performed. The boolean expression is then evaluated again; this continues until the expression evaluates to false.

As of [J2SE 5.0](https://en.wikipedia.org/wiki/J2SE_5.0), the for keyword can also be used to create a so-called "[enhanced for loop](https://en.wikipedia.org/wiki/Foreach)", which specifies an [array](https://en.wikipedia.org/wiki/Array_data_type) or [Iterable](https://docs.oracle.com/javase/10/docs/api/java/lang/Iterable.html) object; each iteration of the loop executes the associated block of statements using a different element in the array or Iterable.

[goto](https://en.wikipedia.org/wiki/Goto)

Unused

[if](https://en.wikipedia.org/wiki/If_statement)

The if keyword is used to create an [if statement](https://en.wikipedia.org/wiki/If_statement), which tests a [boolean expression](https://en.wikipedia.org/wiki/Boolean_expression); if the expression evaluates to true, the block of statements associated with the if statement is executed. This keyword can also be used to create an [if-else statement](https://en.wikipedia.org/wiki/Conditional_(programming)); see [*else*](https://en.wikipedia.org/wiki/List_of_Java_keywords#else).

implements

Included in a class declaration to specify one or more [interfaces](https://en.wikipedia.org/wiki/Interface_(Java)) that are implemented by the current class. A class inherits the types and abstract methods declared by the interfaces.

import

Used at the beginning of a [source file](https://en.wikipedia.org/wiki/Source_file) to specify classes or entire [Java packages](https://en.wikipedia.org/wiki/Java_package) to be referred to later without including their package names in the reference. Since J2SE 5.0, import statements can import static members of a class.

instanceof

A [binary operator](https://en.wikipedia.org/wiki/Operator_(programming)) that takes an object reference as its first operand and a class or interface as its second operand and produces a boolean result. The instanceof operator evaluates to true if and only if the runtime type of the object is assignment compatible with the class or interface.

[int](https://en.wikipedia.org/wiki/Integer_(computer_science))

The int keyword is used to declare a variable that can hold a 32-bit signed two's complement integer. This keyword is also used to declare that a method returns a value of the primitive type int.

[interface](https://en.wikipedia.org/wiki/Interface_(Java))

Used to declare a special type of class that only contains abstract or default methods, constant (static final) fields and static interfaces. It can later be implemented by classes that declare the interface with the implements keyword. As multiple inheritance is not allowed in Java, interfaces are used to circumvent it. An interface can be defined within another interface.

[long](https://en.wikipedia.org/wiki/Long_integer)

The long keyword is used to declare a variable that can hold a 64-bit signed two's complement integer. This keyword is also used to declare that a method returns a value of the primitive type long.

[native](https://en.wikipedia.org/wiki/Java_Native_Interface)

Used in method declarations to specify that the method is not implemented in the same Java source file, but rather in another language.

[new](https://en.wikipedia.org/wiki/Object_lifetime#Java)

Used to create an instance of a class or array object. Using keyword for this end is not completely necessary (as exemplified by [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language))), though it serves two purposes: it enables the existence of different namespace for methods and class names, it defines statically and locally that a fresh object is indeed created, and of what runtime type it is (arguably introducing dependency into the code).

[package](https://en.wikipedia.org/wiki/Java_package)

Java package is a group of similar classes and interfaces. Packages are declared with the package keyword.

[private](https://en.wikibooks.org/wiki/Java_Programming/Classes,_Objects_and_Types)

The private keyword is used in the declaration of a method, field, or inner class; private members can only be accessed by other members of their own class.

[protected](https://en.wikibooks.org/wiki/Java_Programming/Classes,_Objects_and_Types)

The protected keyword is used in the declaration of a method, field, or inner class; protected members can only be accessed by members of their own class, that class's [subclasses](https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)) or classes from the same [package](https://en.wikipedia.org/wiki/Java_package).

[public](https://en.wikibooks.org/wiki/Java_Programming/Classes,_Objects_and_Types)

The public keyword is used in the declaration of a class, method, or field; public classes, methods, and fields can be accessed by the members of any class.

[return](https://en.wikipedia.org/wiki/Method_(computer_science))

Used to finish the execution of a method. It can be followed by a value required by the method definition that is returned to the caller.

[short](https://en.wikipedia.org/wiki/Short_integer)

The short keyword is used to declare a field that can hold a 16-bit signed two's complement integer. This keyword is also used to declare that a method returns a value of the primitive type short.

[static](https://en.wikipedia.org/wiki/Static_variable)

Used to declare a field, method, or inner class as a class field. Classes maintain one copy of class fields regardless of how many instances exist of that class. static also is used to define a method as a class method. Class methods are [bound](https://en.wikipedia.org/wiki/Name_binding) to the class instead of to a specific instance, and can only operate on class fields. (Classes and interfaces declared as static members of another class or interface are actually top-level classes and are *not* inner classes.)

[strictfp](https://en.wikipedia.org/wiki/Strictfp) (added in [J2SE 1.2](https://en.wikipedia.org/wiki/J2SE_1.2))

A Java keyword used to restrict the precision and rounding of floating point calculations to ensure portability.

[super](https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming))

Inheritance basically used to achieve dynamic binding or run-time polymorphism in java. Used to access members of a class inherited by the class in which it appears. Allows a subclass to access [overridden](https://en.wikipedia.org/wiki/Method_overriding_(programming)) methods and hidden members of its superclass. The super keyword is also used to forward a call from a constructor to a constructor in the superclass.

Also used to specify a lower bound on a type parameter in Generics.

[switch](https://en.wikipedia.org/wiki/Switch_statement)

The switch keyword is used in conjunction with [case](https://en.wikipedia.org/wiki/List_of_Java_keywords#case) and [default](https://en.wikipedia.org/wiki/List_of_Java_keywords#default) to create a [switch statement](https://en.wikipedia.org/wiki/Switch_statement), which evaluates a variable, matches its value to a specific case, and executes the block of statements associated with that case. If no case matches the value, the optional block labelled by default is executed, if included.

[synchronized](https://en.wikipedia.org/wiki/Mutual_exclusion)

Used in the declaration of a method or code block to acquire the [mutex](https://en.wikipedia.org/wiki/Mutex) lock for an object while the current [thread](https://en.wikipedia.org/wiki/Thread_(computer_science)) executes the code. For static methods, the object locked is the class's Class. Guarantees that at most one thread at a time operating on the same object executes that code. The mutex lock is automatically released when execution exits the synchronized code. Fields, classes and interfaces cannot be declared as *synchronized*.

[this](https://en.wikipedia.org/wiki/This_(Java))

Used to represent an instance of the class in which it appears. this can be used to access class members and as a reference to the current instance. The this keyword is also used to forward a call from one constructor in a class to another constructor in the same class.

[throw](https://en.wikipedia.org/wiki/Exception_handling_syntax#Java)

Causes the declared exception instance to be thrown. This causes execution to continue with the first enclosing exception handler declared by the catch keyword to handle an assignment compatible exception type. If no such exception handler is found in the current method, then the method returns and the process is repeated in the calling method. If no exception handler is found in any method call on the stack, then the exception is passed to the thread's uncaught exception handler.

[throws](https://en.wikipedia.org/wiki/Exception_handling_syntax#Java)

Used in method declarations to specify which exceptions are not handled within the method but rather passed to the next higher level of the program. All uncaught exceptions in a method that are not instances of RuntimeException must be declared using the throws keyword.

[transient](https://en.wikipedia.org/wiki/Transient_(computer_programming))

Declares that an instance field is not part of the default [serialized](https://en.wikipedia.org/wiki/Serialization) form of an object. When an object is serialized, only the values of its non-transient instance fields are included in the default serial representation. When an object is deserialized, transient fields are initialized only to their default value. If the default form is not used, e.g. when a *serialPersistentFields* table is declared in the class hierarchy, all transient keywords are ignored.

[try](https://en.wikipedia.org/wiki/Exception_handling_syntax#Java)

Defines a block of statements that have exception handling. If an exception is thrown inside the try block, an optional catch block can handle declared exception types. Also, an optional finally block can be declared that will be executed when execution exits the try block and catch clauses, regardless of whether an exception is thrown or not. A try block must have at least one catch clause or a finally block.

[void](https://en.wikipedia.org/wiki/Void_return_type)

The void keyword is used to declare that a method does not return any value.[[6]](https://en.wikipedia.org/wiki/List_of_Java_keywords#cite_note-return-6)

[volatile](https://en.wikipedia.org/wiki/Volatile_variable)

Used in field declarations to guarantee visibility of changes to variables across threads. Every read of a volatile variable will be read from main memory, and not from the CPU cache, and that every write to a volatile variable will be written to main memory, and not just to the CPU cache. Methods, classes and interfaces thus cannot be declared *volatile*, nor can local variables or parameters.

[while](https://en.wikipedia.org/wiki/Do_while_loop)

The while keyword is used to create a [while loop](https://en.wikipedia.org/wiki/While_loop), which tests a [boolean expression](https://en.wikipedia.org/wiki/Boolean_expression) and executes the block of statements associated with the loop if the expression evaluates to true; this continues until the expression evaluates to false. This keyword can also be used to create a [do-while loop](https://en.wikipedia.org/wiki/Do-while_loop); see [*do*](https://en.wikipedia.org/wiki/List_of_Java_keywords#do).

**Reserved words for literal values**

[true](https://en.wikipedia.org/wiki/Truth_value)

A boolean literal value.

[false](https://en.wikipedia.org/wiki/Truth_value)

A boolean literal value.

[null](https://en.wikipedia.org/wiki/Null_pointer)

A reference literal value.

**Special identifiers**

var

A special identifier that cannot be used as a type name (since Java 10).

\_

Added in Java 9, the underscore has become a keyword and cannot be used as a variable name anymore.

**Unused**

[const](https://en.wikipedia.org/wiki/Constant_(programming))

Although reserved as a keyword in Java, const is not used and has no function. For defining constants in Java, see the final keyword.

[goto](https://en.wikipedia.org/wiki/GOTO)

Although reserved as a keyword in Java, goto is not used and has no function.