ORACLE University

Glossary of Java terms

Here are a few useful Java terms and their definitions, followed by links to additional Java glossaries.

Abstract class
A class with the abstract reserved word in its header. Abstract classes are distinguished by the fact that you may not *directly* construct objects from them using the new operator. An abstract class may have zero or more *abstract methods.*

Abstraction
A simplified representation of something that is potentially quite complex. It is often not necessary to know the exact details of how something works, is represented or is implemented, because we can still make use of it in its simplified form. Object-oriented design often involves finding the right level of abstraction at which to work when modeling real-life objects. If the level is too high, then not enough detail will be captured. If the level is too low, then a program could be more complex and difficult to create and understand than it needs to be.

Abstract method
A method with the abstract reserved word in its header. An abstract method has no *method body.* Methods defined in an *interface* are always abstract. The body of an abstract method must be defined in a *sub class* of an *abstract class,* or the body of a class implementing an interface.

Applet
Applets are Java programs based around the Applet or JApplet classes. They are most closely associated with the ability to provide active content within Web pages. They have several features which distinguish them from ordinary Java graphical *applications,* such as their lack of a user-defined main method, and the security restrictions that limit their abilities to perform some normal tasks.

Application
Often used simply as a synonym for *program.* However, in Java, the term is particularly used of programs with a *Graphical User Interface (GUI)* that are not *applets.*

Application programming interface (API)
A set of definitions that you can make use of in writing programs. In the context of Java, these are the packages, classes, and interfaces that can be used to build complex applications without having to create everything from scratch.

Argument
Information passed to a *method.* Arguments are also sometimes called parameters. A method expecting to receive arguments must contain a *formal argument* declaration for each as part of its *method header.* When a method is called, the *actual argument* values are copied into the corresponding formal arguments.

Array
A fixed-size object that can hold zero or more items of the array's declared type.

Attributes – The data defined in the class from which objects are created.

Behaviors – The behaviors defined in a class are "methods" in a Java class.

Block
Statements and declarations enclosed between a matching pair of curly brackets ({ and }). For instance, a *class body* is a block, as is a *method body.* A block encloses a nested *scope* level.

Bookmark
Used by a Web browser to remember details of a *Uniform Resource Locator (URL).*

Boolean
One of Java's *primitive types.* The boolean type has only two values: true and false.

Boolean expression
An expression whose result is of type boolean, i.e. gives a value of either true or false. Operators such as && and || take boolean operands and produce a boolean result. The relational operators take operands different types and produce boolean results.

Boot
When a computer is switched on it is said to `boot up'. This term comes from the phrase, `Pulling yourself up by your bootstraps.' Before a computer is ready to be used, it must load the programs that it needs from its disks, but this means that it must have a program of some sort available in order to be able to load the programs it needs! The loading program is called a bootstrap.

Bootstrap classes
Classes that make up the *Java Platform Core Application Programming Interface (API),* such as those found in the java.lang, java.io and java.io packages.

ORACLE University

Glossary of Java terms

Boundary error
Errors that arise from programming mistakes made at the edges of a problem - indexing off the edge of an array, dealing with no items of data, loop termination and so on. Boundary errors are a very common type of logical error.

Class
A blueprint or prototype from which objects are created. An example might be a "Customer" class. The class can define attributes such as "name", "customer number", "address", etc. The class also defines behaviors (methods) for objects created with the class. Example methods might be used to set the customer's address, change the customer's address, and get the address.

Class body
The body of a *class* definition. The body groups the definitions of a class's *members - fields, methods* and *nested classes.*

Class constant
A variable defined as both final and static.

Class header
The header of a *class* definition. The header gives a name to the class and defines its *access.* It also describes whether the class extends a *super class* or implements any *interfaces.*

Compile
The action by a program (the compiler) of converting source code to a binary file that contains the low-level instructions that the computer's processor will execute.

Daemon Thread
Daemon threads are non-user threads. They are typically used to carry out low-priority tasks that should not take priority over the main task of the program. They can be used to do useful work when all other user threads are blocked. The *garbage collector* is one example of a daemon thread.

Datagram
A packet of information passed between two communicating processes across a network. Both the *Transmission Control Protocol (TCP)* and the *User Datagram Protocol (UDP)* are indirectly involved in sending datagrams to provide reliable or unreliable communication, respectively.

Data Type
There are eight primitive data types in Java; five of these represent numerical types of varying range and precision - double, float, int, long and short. The remaining three are used to representing single-bit values (boolean), single byte values (byte) and two-byte characters from the ISO Unicode character set (char).

Deadlock
A situation that arises when two *threads* each acquires the lock to one of a set of resources that they both need.

Decimal
Number representation in base 10. In base 10, the digits 0 to 9 are used. Digit positions represent successive powers of 10.

Declaration and initialization
A statement in which a variable is declared and immediately given its initial value.

Exception
An object representing the occurrence of an exceptional circumstance - typically, something that has gone wrong in the smooth running of a program. Exception objects are created from *classes* that extend the Throwable class. *See checked exception* and *unchecked exception.*

Exception handler
The *try statement* acts as an exception handler - a place where *exception* objects are caught and dealt with.

Exclusive-or operator
The exclusive-or operator (^) is both a *boolean operator* and a *bit manipulation operator.* The boolean version gives the value true if only one of its operands is true, otherwise it gives the value false. Similarly, the bit manipulation version produces a 1 bit wherever the corresponding bits in its operands are different.

Expression
A combination of *operands* and *operators* that produces a resulting value. Expressions have a resulting type, that affects the context in which they may be used. See *boolean expression* and *arithmetic expression,* for instance.

Executable – The binary file that is created when the source code is compiled. The executable is the "program" that the computer runs or "executes".

Functional programming
A style of programming associated with languages such as Haskell. Functional programming languages are more closely tied to a mathematical concept of `function' than *imperative programming languages.* This makes it easier to apply program-proving techniques and logical reasoning to functional programs. In particular, functional programs do not use the concept of *variables* in the traditional sense, i.e. a memory location whose contents might be changed from time to time as a program executes.

ORACLE University

Glossary of Java terms

Garbage collector
A *daemon thread* that recycles objects to which there are no extant references within a program.
Icon
An image intended to communicate a language- or culturally-independent meaning.

Identifier
A programmer-defined name for a *variable, method, class* or *interface.*

Image map
An image divided into logical areas, each of which has a *hot spot.*

Immutable object
An object whose *state* may not be changed. Objects of the String class are immutable, for instance - their length and contents are fixed once created.

Imperative programming
The style of programming usually associated with languages such as C, Fortran, Pascal and so on. Imperative programming is distinguished from *functional programming* in that the former is strongly tied to the concept of variables and memory locations. A variable is associated with a memory location and the contents of that memory location may be changed, via the variable, over the course of time.

Inheritance – A mechanism where one class acquires the properties of another class. For example, a child inherits the traits (attributes and behaviors) of the parent. The child class uses the traits that are common with those of the parents, but adds the traits unique to the child. Inheritance facilitates reusability and is an important object-oriented concept.

Java Archive file
A Java Archive (JAR) file makes it possible to store multiple *bytecode* files in a single file.

Java Bytecode – The instruction set of the Java virtual machine (JVM). Compiling Java source code results in a Java Bytecode that can be executed on any computer with an installed JVM. This allows for the concept of "write once, run anywhere" or WORA.

Java 2 SDK
A particular implementation of the abstract functionality described in Sun's specification of the Java 2 Platform.

Java Virtual Machine (JVM) – The computer program that is installed on every computer capable of running Java programs. It accepts Java Bytecode and converts the Bytecode to the machine language of the given computer.

Key value
The object used to generate an associated *hash* code for lookup in an associative data structure.

Lambda expression
A feature which is introduced in Java 8. A lambda expression is an anonymous function. A function that doesn't have a name and doesn't belong to any class. The concept of lambda expression was first introduced in LISP programming language.

Java Lambda Expression Syntax
To create a lambda expression, we specify input parameters (if there are any) on the left side of the lambda operator ->, and place the expression or block of statements on the right side of lambda operator. For example, the lambda expression (x, y) -> x + y specifies that lambda expression takes two arguments x and y and returns the sum of these.

Loop variable
A *variable* used to control the operation of a loop, such as a *for loop.* Typically, a loop variable will be given an initial value and it is then incremented after each *iteration* until it reaches or passes a terminating value.

Low level programming languages
Often known as `assembly languages', these provide little more than the basic instruction set of a particular *Central Processing Unit.* Hence programs written in low level programming languages tend to be less *portable* than those written in *high level languages.*

Machine language – The instructions that a given computer processor can execute.

Method
The part of a *class definition* that implements some of the behavior of objects of the class. The body of the method contains *declarations* of *local variables* and *statements* to implement the behavior. A method receives input via its *arguments*, if any, and may return a result if it has not been declared as void.

Method body
The body of a method: everything inside the outermost *block* of a method.

Method overloading
Two or more methods with the same name defined within a class are said to be overloaded. This applies to both constructors and other methods. Overloading applies through a class hierarchy, so a sub class might overload a method defined in one of its super classes. It is important to distinguish between an overloaded method and an *overridden method.* Overloaded methods must be distinguishable in some way from each other; either by having different numbers of arguments, or by the types of those arguments being different. Overridden methods have identical formal arguments.

ORACLE University

Glossary of Java terms

Modularity
This is a general concept. In software, it applies to writing and implementing a program or computing system as a number of unique modules, rather than as a single, monolithic design. A standardized interface is then used to enable the modules to communicate. Partitioning an environment of software constructs into distinct modules helps us minimize coupling, optimize application development, and reduce system complexity.
Modularity enables programmers to do functionality testing in isolation and engage in parallel development efforts during a given sprint or project. This increases efficiency throughout the entire software development lifecycle.

Module
A group of program components, typically with restricted visibility to program components in other modules. Java uses *packages* to implement this concept.

Namespace
The area of a program in which particular *identifiers* are visible. Java uses *packages* to provide namespaces, and its visibility rules - private, package, protected, public - variously contain identifiers within namespaces.

Native method
A method written in a language other than Java, but accessible to a Java program.

Object – A software bundle of related state (data) and behavior. Software objects are often used to model the real-world objects that you find in everyday life. Objects are created (instantiated) from the class defining the object that is being instantiated.

Object-oriented – Object-oriented analysis (OOA), object-oriented design (OOD) and object-oriented programming (OOP) represent a paradigm for creating software applications. The paradigm is organized around objects rather than actions, and around data rather than logic. The primary concepts utilized are abstraction, encapsulation, polymorphism and inheritance.

Package declaration
A declaration used to name a *package.* This must be the first item in a source file, preceding any *import statements.* For instance, package java.lang;

Parallel programming
A style of programming in which statements are not necessarily executed in an ordered sequence but in parallel. Parallel programming languages make it easier to create programs that are designed to be run on multi-processor hardware, for instance. Java's *thread* features support a degree of parallel programming.

Parsing
Usually applied to the action of a *compiler* in analyzing a program source file for *syntax errors.* It is also used more widely to mean the analysis of the structure of input.

Pattern
A recurring theme in class design or usage. Interfaces such as Iterator encapsulate a pattern of access to the items in a collection, while freeing the client from the need to know details of the way in which the collection is implemented.

Polymorphism
The ability of an object reference to be used as if it referred to an object with different forms. Polymorphism in Java results from both *class inheritance* and *interface inheritance.* The apparently different forms often result from the *static type* of the variable in which the reference is stored. Given the following class header class Rectangle extends Polygon implements Comparable.

Primitive type
Java's eight standard non-class types are primitive types: boolean, byte, char, double, float, int, long and short.

Priority level
Each *thread* has a priority level, which indicates to the *scheduler* where it should be placed in the pecking order for being run. An eligible un-blocked thread with a particular priority will *always* be run before an eligible thread with a lower priority.

Protocol
A set of rules for interaction between two *processes.* A protocol is usually specified in a *Uniform Resource Locator (URL)* to indicate how a particular *resource* should be transferred from a Web server to the requesting client.

Scheduler
The part of the *Java Virtual Machine (JVM)* that is responsible for managing threads.

Server-side language – In a client-server computing environment the client (for example, a web browser acting as a client) makes requests of the server to do something (send some information or process information submitted). The web server software on a computer, usually somewhere else on the network, responds to the client's request. The software program on the server is often written in Java.

Source code – A plain text listing of commands to be compiled or assembled into an executable computer program. The commands of the higher-level language (e.g., Java) are defined by the grammar, rules and syntax of the language.

Stack
See *last in, first out (LIFO) stack.*

ORACLE University

Glossary of Java terms

Stack overflow
Stack overflow occurs when too many items are pushed onto a *stack* with a finite capacity.

Stack trace
A display of the *runtime stack.*

String
An instance of the String class. Strings consist of zero or more *Unicode* characters, and they are *immutable,* once created. A literal string is written between a pair of string delimiters (“), as in “hello, world”.

Structured programming
A style of programming usually associated with languages such as C, Fortran, Pascal and so on. Using structured programming techniques, a problem is often solved using a *divide and conquer* approach such as *stepwise* refinement. An initially large problem is broken into several smaller sub-problems. Each of these is then progressively broken into even smaller sub-problems, until the level of difficulty is considered to be manageable. At the lowest level, a solution is implemented in terms of data structures and procedures. This approach is often used with *imperative programming* languages that are not *object-oriented languages,* i.e. the data structures and procedures are not implemented as classes.

Sub class
A class that extends its *super class.* A sub class *inherits* all of the members of its super class. All Java classes are sub classes of the Object class, which is at the root of the inheritance *hierarchy.* See *sub type.*

Sub type
A type with a parent *super type.* The sub-type/super-type relationship is more general than the sub-class/super-class relationship. A class that implements an *interface* is a sub type of the interface. An interface that extends another interface is also a sub type.

Swing
The Swing classes are part of a wider collection known as the *Java Foundation Classes (JFC).* Swing classes are defined in the javax. swing packages. They provide a further set of components that extend the capabilities of the Abstract *Windowing Toolkit (AWT).* Of particular significance is the greater control they provide over an application’s *look-and-feel.*

Thread
A lightweight *process* that is managed by the *Java Virtual Machine (JVM).* Support for threads is provided by the Thread class in the java.lang package.

Thread starvation
A condition that applies to a *thread* that is prevented from running by other threads that do not yield or become blocked.

Throw an exception
When an exceptional circumstance arises in a program - often as a result of a *logical error,* and *exception* object is created and thrown. If the exception is not caught by an *exception handler,* the program will terminate with a *runtime error.*

While loop
A file used to store compressed versions of files. In connection with Java *bytecode* files, these have largely been superseded by *Java Archive (JAR) files.*

Zip file
One of Java’s three *control structures* used for looping. The other two are the *do loop* and *for loop.* A while loop consists of a *boolean expression* and a loop body. The condition is tested before the loop body is entered for the first time and re-tested each time the end of the body is completed. The loop terminates when the condition gives the value false. The statements in the loop body might be executed zero or more times.

Content Source: Some of terms were taken from the book, Object Oriented-Programming with Java: An Introduction, by David J. Barnes, published by Prentice Hall.