Java Learning Glossary

From:
- Think Java How to Think Like a Computer Scientist, Version 5.1.2, By Allen B. Downey, MIT University, 2012
- Java How to Program, Paul Deitel, Harvey Deitel, Early Objects-Pearson Education (2015)

Hint: Some entries are duplicated!
By: Mehdi Einali

Problem-solving: The process of formulating a problem, finding a solution, and expressing the solution.

High-level language: A programming language like Java that is designed to be easy for humans to read and write.

Low-level language: A programming language that is designed to be easy for a computer to run. Also called "machine language" or "assembly language."

Formal language: Any of the languages people have designed for specific purposes, like representing mathematical ideas or computer programs. All programming languages are formal languages.

Natural language: Any of the languages people speak that have evolved naturally.

Portability: A property of a program that can run on more than one kind of computer.

Interpret: To run a program in a high-level language by translating it one line at a time.

Compile: To translate a program in a high-level language into a low-level language, all at once, in preparation for later execution.

Source code: A program in a high-level language, before being compiled.

Object code: The output of the compiler, after translating the program.

Executable: Another name for object code that is ready to run.

Byte code: A special kind of object code used for Java programs. Byte code is similar to a low-level language, but it is portable, like a high-level language.
**Statement:** A part of a program that specifies a computation.

**Print statement:** A statement that causes output to be displayed on the screen.

**Comment:** A part of a program that contains information about the program, but that has no effect when the program runs.

**Method:** A named collection of statements.

**Library:** A collection of class and method definitions.

**Bug:** An error in a program.

**Syntax:** The structure of a program.

**Semantics:** The meaning of a program.

**Parse:** To examine a program and analyze the syntactic structure.

**Syntax error:** An error in a program that makes it impossible to parse (and therefore impossible to compile).

**Exception:** An error in a program that makes it fails at run-time. Also called a run-time error.

**Logic error:** An error in a program that makes it do something other than what the programmer intended.

**Debugging:** The process of finding and removing any of the three kinds of errors.

**Variable:** A named storage location for values. All variables have a type, which is declared when the variable is created.

**Value:** A number or string (or other thing to be named later) that can be stored in a variable. Every value belongs to a type.

**Type:** A set of values. The type of a variable determines which values can be stored there. The types we have seen are integers (int in Java) and strings (String in Java).

**Keyword:** A reserved word used by the compiler to parse programs. You cannot use keywords, like public, class and void as variable names.

**Declaration:** A statement that creates a new variable and determines its type.

**Assignment:** A statement that assigns a value to a variable.
Expression: A combination of variables, operators and values that represents a single value. Expressions also have types, as determined by their operators and operands.

Operator: A symbol that represents a computation like addition, multiplication or string on catenation.

Operand: One of the values on which an operator operates.

Precedence: The order in which operations are evaluated.

Concatenate: To join two operands end-to-end.

Composition: The ability to combine simple expressions and statements into compound statements and expressions to represent complex computations concisely.

Initialization: A statement that declares a new variable and assigns a value to it at the same time.

Floating-point: A type of variable (or value) that can contain fractions as well as integers. The floating-point type we will use is `double`.

Class: A named collection of methods.

Method: A named sequence of statements that performs a useful function. Methods may or may not take parameters, and may or may not return a value.

Parameter: A piece of information a method requires before it can run. Parameters are variables: they contain values and have types.

Argument: A value that you provide when you invoke a method. This value must have the same type as the corresponding parameter.

Invoke: Cause a method to execute.

Modulus: An operator that works on integers and yields the remainder when one number is divided by another. In Java it is denoted with a percent sign(%).

Conditional: A block of statements that may or may not be executed depending on some condition.

Chaining: A way of joining several conditional statements in sequence.

Nesting: Putting a conditional statement inside one or both branches of another conditional statement.

Typecast: An operator that converts from one type to another. In Java it appears as a type name in parentheses, like `( int )`. 
**Recursion:** The process of invoking the same method you are currently executing.

**Base case:** best scenario for performance in running for example in recursive method a condition that causes a recursive method **not** to make a recursive call.

**Return type:** The part of a method declaration that indicates what type of value the method returns.

**Return value:** The value provided as the result of a method invocation.

**Dead code:** Part of a program that can never be executed, often because it appears after a return statement.

**Scaffolding:** Code that is used during program development but is not part of the final version.

**Void:** A special return type that indicates a void method; that is, one that does not return a value.

**Overloading:** Having more than one method with the same name but different parameters. When you invoke an overloaded method, Java knows which version to use by looking at the arguments you provide.

**Boolean:** A type of variable that can contain only the two values true and false.

**Flag:** A variable (usually **boolean**) that records a condition or status information.

**Conditional operator:** An operator that compares two values and produces a boolean that indicates the relationship between the operands.

**Logical operator:** An operator that combines boolean values and produces boolean values.

**Loop:** A statement that executes repeatedly while some condition is satisfied.

**Infinite loop:** A loop whose condition is always true.

**Body:** The statements inside the loop.

**Iteration:** One pass through (execution of) the body of the loop, including the evaluation of the condition.

**Encapsulate:** To divide a large complex program into components (like classes) and isolate the components internal structure and behavior from each other (for example, by using methods).

**Local variable:** A variable that is declared inside a method and that exists only within that method. Local variables cannot be accessed from outside their home method, and do not interfere with any other methods.
Generalize: To replace something unnecessarily specific (like a constant value) with something appropriately general (like a variable or parameter). Generalization makes code more versatile, more likely to be reused, and sometimes even easier to write.

Program development: A process for writing programs. So far we have seen "incremental development" and "encapsulation and generalization".

Object: A collection of related data that comes with a set of methods that operates on it.

Index: A variable or value used to select one of the members of an ordered set, like a character from a string.

Exception: A run-time error.

Throw: Cause an exception.

StackTrace: A report that shows the state of a program when an exception occurs.

Method Signature (Prototype): The first line of a method, which specifies the name, parameters and return type.

Traverse: To iterate through all the elements of a set performing a similar operation on each.

Counter: A variable used to count something, usually initialized to zero and then incremented.

Increment: Increase the value of a variable by one. The increment operator in Java is ++.

Decrement: Decrease the value of a variable by one. The decrement operator in Java is --.

Package: A collection of classes. Java classes are organized in packages.

AWT: The Abstract Window Toolkit, one of the biggest and commonly-used Java packages.

Instance: An example from a category. My cat is an instance of the category "feline things." Every object is an instance of some class.

Instance Variable: One of the named data items that make up an object. Each object (instance) has its own copy of the instance variables for its class.

Reference: A value that indicates an object. In a state diagram, a reference appears as an arrow.

Aliasing: The condition when two or more variables refer to the same object.

Garbage collection: The process of finding objects that have no references and reclaiming their storage space.
State: A complete description of all the variables and objects and their values, at a given point during the execution of a program.

State diagram: A snapshot of the state of a program, shown graphically

Class: Previously, I defined a class as a collection of related methods. In this chapter we learned that a class definition is also a template for a new type of object.

Instance: A member of a class. Every object is an instance of some class.

Constructor: A special method that initializes the instance variables of a newly-constructed object.

Startup class: The class that contains the main method where execution of the program begins.

Pure function: A method whose result depends only on its parameters, and that has no side-effects other than returning a value.

Modifier Method: A method that changes one or more of the objects it receives as parameters, and usually returns void.

Fill-in method: A type of method that takes an "empty" object as a parameter and fills in its instance variables instead of generating a return value.

Algorithm: A set of instructions for solving a class of problems by a mechanical process.

Array: A collection of values, where all the values have the same type, and each value is identified by an index.

Element: One of the values in an array. The [] operator selects elements.

Index: An integer variable or value used to indicate an element of an array.

Deterministic: A program that does the same thing every time it is invoked.

Pseudorandom: A sequence of numbers that appear to be random, but which are actually the product of a deterministic computation.

Histogram: An array of integers where each integer counts the number of values that fall into a certain range.

Encode: To represent one set of values using another set of values, by constructing a mapping between them.

Identity: Equality of references. Two references that point to the same object in memory.

Equivalence: Equality of values. Two references that point to objects that contain the same data.
Abstract parameter: A set of parameters that act together as a single parameter.

Abstraction: The process of interpreting a program (or anything else) at a higher level than what is literally represented by the code.

Pseudocode: A way of designing programs by writing rough drafts in a combination of English and Java (or other programming language).

Helper method: Often a small method that does not do anything enormously useful by itself, but which helps another, more useful method.

Class variable (Static Variable): A variable declared within a class as static; there is always exactly one copy of this variable in existence.

Object method: A method that is invoked on an object, and that operates on that object. Object methods do not have the keyword static.

Class method: A method with the keyword static. Class methods are not invoked on objects and they do not have a current object.

Current object: The object on which an object method is invoked. Inside the method, the current object is referred to by this.

Implicit: Anything that is left unsaid or implied. Within an object method, you can refer to the instance variables implicitly (i.e., without naming the object).

Explicit: Anything that is spelled out completely. Within a class method, all references to the instance variables have to be explicit.

Coordinate: A variable or value that specifies a location in a two-dimensional graphical window.

Pixel: The unit in which coordinates are measured.

Bounding box: A common way to specify the coordinates of a rectangular area.

Bottom-up Development: Start by writing simple methods, then assemble them into a solution.

Top-down Development: Use pseudocode to design the structure of the computation and identify the methods you'll need. Then write the methods and replace the pseudocode with real code.

Agile software development: Agile software development is a set of methodologies that try to get software implemented faster and using fewer resources. Check out the Agile Alliance (www.agilealliance.org) and the Agile Manifesto (www.agilemanifesto.org).

Refactoring: Refactoring involves reworking programs to make them clearer and easier to maintain while preserving their correctness and functionality. It’s widely employed with agile development
methodologies. Many IDEs contain built-in refactoring tools to do major portions of the reworking automatically.

**Design patterns:** Design patterns are proven architectures for constructing flexible and maintainable object-oriented software. The field of design patterns tries to enumerate those recurring patterns, encouraging software designers to reuse them to develop better-quality software using less time, money and effort.

**LAMP:** LAMP is an acronym for the open-source technologies that many developers use to build web applications—it stands for Linux, Apache, MySQL and PHP (or Perl or Python—two other scripting languages). MySQL is an open-source database management system. PHP is the most popular opensource server-side “scripting” language for developing web applications. Apache is the most popular web server software. The equivalent for Windows development is WAMP—Windows, Apache, MySQL and PHP.

**Software as a Service (SaaS):** Software has generally been viewed as a product; most software still is offered this way. If you want to run an application, you buy a software package from a software vendor—often a CD, DVD or web download. You then install that software on your computer and run it as needed. As new versions appear, you upgrade your software, often at considerable cost in time and money. This process can become cumbersome for organizations that must maintain tens of thousands of systems on a diverse array of computer equipment. With Software as a Service (SaaS), the software runs on servers elsewhere on the Internet. When that server is updated, all clients worldwide see the new capabilities—no local installation is needed. You access the service through a browser. Browsers are quite portable, so you can run the same applications on a wide variety of computers from anywhere in the world. Salesforce.com, Google, and Microsoft’s Office Live and Windows Live all offer SaaS.

**Platform as a Service (PaaS):** Platform as a Service (PaaS) provides a computing platform for developing and running applications as a service over the web, rather than installing the tools on your computer. Some PaaS providers are Google App Engine, Amazon EC2 and Windows Azure™.

**Cloud computing:** SaaS and PaaS are examples of cloud computing. You can use software and data stored in the “cloud”—i.e., accessed on remote computers (or servers) via the Internet and available on demand—rather than having it stored on your desktop, notebook computer or mobile device. This allows you to increase or decrease computing resources to meet your needs at any given time, which is more cost effective than purchasing hardware to provide enough storage and processing power to meet occasional peak demands. Cloud computing also saves money by shifting the burden of managing these apps to the service provider.

**Software Development Kit (SDK):** Software Development Kits (SDKs) include the tools and
documentation developers use to program applications. For example, you'll use the Java Development Kit (JDK) to build and run Java applications.