Rule BB. Glossary

alien method
From the perspective of a class C, an alien method is one whose behavior is not fully specified by C. This includes methods in other classes as well as overridable methods (neither private nor final) in C itself [Goetz 2006a].

anti-pattern
An anti-pattern is a pattern that may be commonly used but is ineffective and/or counterproductive in practice [Laplante 2005].

availability
The degree to which a system or component is operational and accessible when required for use. Often expressed as a probability [IEEE Std 610.12 1990].

big-endian
"Multibyte data items are always stored in big-endian order, where the high bytes come first" [JVMSpec 1999, Chapter 4, "The `class` File Format"]. This term refers to the tension between Lilliput and Blefuscu (regarding whether to open soft-boiled eggs from the large or the small end) in Jonathan Swift's satirical novel *Gulliver's Travels*; it was first applied to the question of byte-ordering by Danny Cohen [Cohen 1981].

canonicalization
Reducing the input to its equivalent simplest known form.

class variable
A class variable is a field declared using the keyword static within a class declaration or with or without the keyword static within an interface declaration. A class variable is created when its class or interface is prepared and is initialized to a default value. The class variable effectively ceases to exist when its class or interface is unloaded [JLS 2005].

condition predicate
An expression constructed from the state variables of a class that must be true for a thread to continue execution. The thread pauses execution, via `Object.wait()`, `Thread.sleep()`, or some other mechanism, and is resumed later, presumably when the requirement is true and when it is notified [Goetz 2006a].

conflicting accesses
Two accesses to (reads of or writes to) the same variable provided that at least one of the accesses is a write [JLS 2005].

data race
Conflicting accesses of the same variable that are not ordered by a happens-before relationship [JLS 2005].

deadlock
Two or more threads are said to have deadlocked when both block waiting for each other's locks. Neither thread can make any progress.

denial-of-service attack
Also *DoS attack*. An attempt to make a computer resource unavailable to its intended users.

error tolerance
The ability of a system or component to continue normal operation despite the presence of erroneous inputs [IEEE Std 610.12 1990].

exploit [Seacord 2005]
A piece of software or a technique that takes advantage of a security vulnerability to violate an explicit or implicit security policy.

fail fast [Gray 1985]
Pertaining to software that either functions correctly or detects the fault, signals failure, and stops operating.

fail safe [IEEE Std 610.12 1990]
Pertaining to a system or component that automatically places itself in a safe operating mode in the event of a failure—for example, a traffic light that reverts to blinking red in all directions when normal operation fails.

fail soft [IEEE Std 610.12 1990]
Pertaining to a system or component that continues to provide partial operational capability in the event of certain failures—for example, a traffic light that continues to alternate between red and green if the yellow light fails.

fault tolerance [IEEE Std 610.12 1990]
The ability of a system or component to continue normal operation despite the presence of hardware or software faults.

fields [JLS 2014]
The class variables and instance variables of classes, and constants of interfaces.

happens-before order
"Two actions can be ordered by a happens-before relationship. If one action happens-before another, then the first is visible to and ordered before the second. ... It should be noted that the presence of a happens-before relationship between two actions does not necessarily imply that they have to take place in that order in an implementation. If the reordering produces results consistent with a legal execution, it is not illegal. ... More specifically, if two actions share a happens-before relationship, they do not necessarily have to appear to have happened in that order to any code with which they do not share a happens-before relationship. Writes in one thread that are in a data race with reads in another thread may, for example, appear to occur out of order to those reads" [JLS 2005].

heap memory
"Memory that can be shared between threads is called shared memory or heap memory. All instance fields, static fields and array elements are stored in heap memory. ... Local variables (§14.4), formal method parameters (§8.4.1) or exception handler parameters are never shared between threads and are unaffected by the memory model" [JLS 2005].

hide
One class field hides a field in a superclass if they have the same identifier. The hidden field is not accessible from the class. Likewise, a class method hides a method in a superclass if they have the same identifier but incompatible signatures. The hidden method is not accessible from the class. See [JLS 2005] §8.4.8.2 for the formal definition. Contrast with override.

immutable
When applied to an object, this means that its state cannot be changed after being initialized. An object is immutable if:

- Its state cannot be modified after construction;
- All its fields are final; and
- It is properly constructed (the `this` reference does not escape during construction).

It is technically possible to have an immutable object without all fields being final. `String` is such a class but this relies on delicate reasoning about benign data races that requires a deep understanding of the Java Memory Model. (For the curious: `String` lazily computes the hash code the first time `hashCode` is called and caches it in a nonfinal field, but this works only because that field can take on only one nondefault value that is the same every time it is computed because it is derived deterministically from immutable state) [Goetz 2006a].

Immutable objects are inherently thread-safe; they may be shared among multiple threads or published without synchronization, though it is usually required to declare the fields containing their references volatile to ensure visibility. An immutable object may contain mutable subobjects, provided the state of the subobjects cannot be modified after construction of the immutable object has concluded.

initialization safety
An object is considered to be completely initialized when its constructor finishes. A thread that can see a reference to an object only after that object has been completely initialized is guaranteed to see the correctly initialized values for that object's final fields [JLS 2005].

instance variable
An instance variable is a field declared within a class declaration without using the keyword `static`. If a class `T` has a field a that is an instance variable, then a new instance variable a is created and initialized to a default value as part of each newly created object of class `T` or of any class that is a subclass of `T`. The instance variable effectively ceases to exist when the object of which it is a field is no longer referenced, after any necessary finalization of the object has been completed [JLS 2005].

interruption policy
An interruption policy determines how a thread interprets an interruption request—what it does (if anything) when one is detected, what units of work are considered atomic with respect to interruption, and how quickly it reacts to interruption [Goetz 2006a].

invariant
A property that is assumed to be true at certain points during program execution but not formally specified. Invariants may be used in `assert` statements or informally specified in comments. Invariants are often used to reason about program correctness.

liveness
A property that every operation or method invocation executes to completion without interruptions, even if it goes against safety.

memoization
An optimization technique used primarily to speed up computer programs by having function calls avoid repeating the calculation of results for previously processed inputs [White 2003].

memory model
"The rules that determine how memory accesses are ordered and when they are guaranteed to be visible are known as the memory model of the Java programming language" [JPL 2006]. "A memory model describes, given a program and an execution trace of that program, whether the execution trace is a legal execution of the program" [JLS 2005].

normalization
Lossy conversion of the data to its simplest known (and anticipated) form. "When implementations keep strings in a normalized form, they can be assured that equivalent strings have a unique binary representation" [Davis 2008a].

normalization (URI)
Normalization is the process of removing unnecessary "." and ".." segments from the path component of a hierarchical URI. Each "." segment is simply removed. A ".." segment is removed only if it is preceded by a non-".." segment. Normalization has no effect on opaque URIs [API 2006].

obscure
One scoped identifier obscures another identifier in a containing scope if the two identifiers are the same, but the obscuring identifier does not shadow the obscured identifier. This can happen when the obscuring identifier is a variable while the obscured identifier is a type, for example. See [JLS 2005] §6.3.2 for more information.

obsolete reference
An obsolete reference is a reference that will never be dereferenced again [Bloch 2008].

open call
An alien method invoked outside of a synchronized region is known as an *open call* [Bloch 2008], [Lea 2000a].

out-of-band error indicator [ISO/IEC TS 17961:2013]
A library function return value used to indicate nothing but the error status.

out-of-domain value [ISO/IEC TS 17961:2013]
One of a set of values that is not in the domain of a particular operator or function.

override
One class method overrides a method in a superclass if they have compatible signatures. The overridden method is still accessible from the class via the `super` keyword. See [JLS 2005] §8.4.8.1 for the formal definition. Contrast with hide.

partial order
"An order defined for some, but not necessarily all, pairs of items. For instance, the sets {a, b} and {a, c, d} are subsets of {a, b, c, d}, but neither is a subset of the other. So a "subset of" is a partial order on sets" [Black 2004].

program order
The order that interthread actions are performed by a thread according to the intrathread semantics of the thread. "Program order [can be described] as the order of bytecodes present in the .class file, as they would execute based on control flow values" (David Holmes, JMM Mailing List).

publishing objects
Publishing an object means making it available to code outside of its current scope, such as by storing a reference to it where other code can find it, returning it from a nonprivate method, or passing it to a method in another class [Goetz 2006a].

race condition
"General races cause nondeterministic execution and are failures in programs intended to be deterministic" [Netzer 1992]. "A race condition occurs when the correctness of a computation depends on the relative timing or interleaving of multiple threads by the runtime" [Goetz 2006a].

relativization (URI)
[Relativization] is the inverse of resolution. For example, relativizing the URI `http://java.sun.com/j2se/1.3/docs/guide/index.html` against the base URI `http://java.sun.com/j2se/1.3` yields the relative URI `docs/guide/index.html` [API 2006].

reentrant [ISO/IEC/IEEE 24765:2010]
Pertaining to a software module that can be entered as part of one process while also in execution as part of another process and still achieve the desired results.

reliability [IEEE Std 610.12 1990]
The ability of a system or component to perform its required functions under stated conditions for a specified period of time.

restricted sink [ISO/IEC 9899:2011]
Operands and arguments whose domain is a subset of the domain described by their types.

robustness [IEEE Std 610.12 1990]
The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

safe publication
To publish an object safely, both the reference to the object and the object's state must be made visible to other threads at the same time. A properly constructed object can be safely published by

- Initializing an object reference from a *static* initializer.
- Storing a reference to it into a *volatile* field.
- Storing a reference to it into a *final* field.
- Storing a reference to it into a field that is properly guarded by a (*synchronized*) lock.
  [Goetz 2006a, Section 3.5 "Safe Publication"]

safety
The main goal of safety is to ensure that all objects maintain consistent states in a multithreaded environment [Lea 2000].

sanitize [ISO/IEC TS 17961:2013]
Assure by testing or replacement that a tainted or other value conforms to the constraints imposed by one or more restricted sinks into which it may flow.

> NOTE
> If the value does not conform, either the path is diverted to avoid using the value or a different, known-conforming value is substituted.

security flaw [ISO/IEC TS 17961:2013]
Defect that poses a potential security risk.

security policy [Internet Society 2000]
Set of rules and practices that specify or regulate how a system or organization provides security services to protect sensitive and critical system resources.

sensitive code
Any code that performs operations forbidden to untrusted code. Also, any code that accesses sensitive data (q.v.). For example, code whose correct operation requires enhanced privileges is typically considered to be sensitive.

sensitive data
Any data that must be kept secure. Consequences of this security requirement include the following:

- Untrusted code is forbidden to access sensitive data.
- Trusted code is forbidden to leak sensitive data to untrusted code.

Examples of sensitive data include passwords and personally identifiable information.

sequential consistency
"Sequential consistency is a very strong guarantee that is made about visibility and ordering in an execution of a program. Within a sequentially consistent execution, there is a total order over all individual actions (such as reads and writes) which is consistent with the order of the program, and each individual action is atomic and is immediately visible to every thread. ... If a program is correctly synchronized, all executions of the program will appear to be sequentially consistent (§17.4.3)" [JLS 2005]. Sequential consistency implies there will be no compiler optimizations in the statements of the action. Adopting sequential consistency as the memory model and disallowing other primitives can be overly restrictive because, under this condition, the compiler is not allowed to make optimizations and reorder code [JLS 2005].

shadow
One scoped identifier shadows another identifier in a containing scope if the two identifiers are the same and they both reference variables. They may also both reference methods or types. The shadowed identifier is not accessible in the scope of the shadowing identifier. See [JLS 2005] §6.3.1 for more information. Contrast with obscure.

synchronization
The Java programming language provides multiple mechanisms for communicating between threads. The most basic of these methods is *synchronization*, which is implemented using monitors. Each object in Java is associated with a monitor, which a thread can lock or unlock. Only one thread at a time may hold a lock on a monitor. Any other threads attempting to lock that monitor are blocked until they can obtain a lock on that monitor [JLS 2005].

starvation
A condition wherein one or more threads prevent other threads from accessing a shared resource over extended periods of time. For instance, a thread that invokes a synchronized method that performs some time-consuming operation starves other threads.

tainted data
Data that either originate from an untrusted source or result from an operation whose inputs included tainted data. Tainted data can be sanitized (also *untai nted*) through suitable data validation. Note that all outputs from untrusted code must be considered to be tainted [Jovanovich 2006].

thread-safe
An object is thread-safe if it can be shared by multiple threads without the possibility of any data races. "A thread-safe object performs synchronization internally, so multiple threads can freely access it through its public interface without further synchronization" [Goetz 2006a]. Immutable classes are thread-safe by definition. Mutable classes may also be thread-safe if they are properly synchronized.

total order
An order defined for all pairs of items of a set. For instance, <= (less than or equal to) is a total order on integers; that is, for any two integers, one of them is less than or equal to the other [Black 2006].

trusted code
Code that is loaded by the primordial class loader, irrespective of whether or not it constitutes the Java API. In this text, this meaning is extended to include code that is obtained from a known entity and given permissions that untrusted code lacks. By this definition, untrusted and trusted code can coexist in the namespace of a single class loader (not necessarily the primordial class loader). In such cases, the security policy must make this distinction clear by assigning appropriate privileges to trusted code while denying those privileges to untrusted code.

untrusted code
Code of unknown origin that can potentially cause some harm when executed. Untrusted code may not always be malicious, but this is usually hard to determine automatically. Consequently, untrusted code should be run in a sandboxed environment.

untrusted data [ISO/IEC 11889-1:2009]
Data originating from outside of a trust boundary.

volatile
"A write to a volatile field (§8.3.1.4) happens-before every subsequent read of that field" [JLS 2005]. "Operations on the master copies of volatile variables on behalf of a thread are performed by the main memory in exactly the order that the thread requested" [JVMSpec 1999]. Accesses to a volatile variable are sequentially consistent, which also means that the operations are exempt from compiler optimizations. Declaring a variable volatile ensures that all threads see the most up-to-date value of the variable if any thread modifies it. Volatile guarantees atomic reads and writes of primitive values; however, it does not guarantee the atomicity of composite operations such as variable incrementation (read-modify-write sequence).

vulnerability
A set of conditions that allow an attacker to violate an explicit or implicit security policy [Seacord 2005].