Hidden Classes

JVMS 5.4.4 Access Control

...

- R is either `protected` or has default access ...
- R is `private` and is declared by a class or interface C that belongs to the same nest as D, according to the nestmate test below.

If R is not accessible to D, then **access control throws an `IllegalAccessError`.**

- ~~If R is `public`, `protected`, or has default access, then access control throws an `IllegalAccessError`.~~
- ~~If R is `private`, then the nestmate test failed, and access control fails for the same reason.~~

Otherwise, access control succeeds.

A *nest* is a set of classes and interfaces that allow mutual access to their `private` members. One of the classes or interfaces is the *nest host*. It enumerates the classes and interfaces which belong to the nest, using the `NestMembers` attribute (§4.7.29). Each of them in turn designates it as the nest host, using the `NestHost` attribute (§4.7.28). A class or interface which lacks a `NestHost` attribute belongs to the nest hosted by itself; if it also lacks a `NestMembers` attribute, this nest is a singleton consisting only of the class or interface itself. **The nest host for a given class or interface (that is, the nest to which the class or interface belongs) is *determined* by the Java Virtual Machine as part of access control, rather than when the class or interface is loaded. Certain methods of the Java SE Platform API may determine the nest host for a given class or interface prior to access control, in which case access control respects the prior determination.**

To determine whether a class or interface C belongs to the same nest as a class or interface D, the *nestmate test* is applied. C and D belong to the same nest if and only if the nestmate test succeeds. The nestmate test is as follows:

- If C and D are the same class or interface, then the nestmate test succeeds.
- Otherwise, the following steps are performed, in order:

    1. **Let H be the nest host of D, if the nest host of D has previously been determined.**
       If the nest host of D **has not previously been determined, then it is determined using the algorithm below, yielding H.** ~~The nest host of D, H, is determined (below). If an exception is thrown, then the nestmate test fails for the same reason.~~
    2. **Let H' be the nest host of C, if the nest host of C has previously been determined.**
       If the nest host of C **has not previously been determined, then it is determined using the algorithm below, yielding H'.** ~~The nest host of C, H', is determined (below). If an exception is thrown, then the nestmate test fails for the same reason.~~
    3. H and H' are compared. If H and H' are the same class or interface, then the nestmate test succeeds. Otherwise, the nestmate test fails~~ by throwing an `IllegalAccessError`~~.

The nest host of a class or interface `M` is determined as follows:

- If `M` lacks a `NestHost` attribute, then `M` is its own nest host.
- Otherwise, `M` has a `NestHost` attribute, and its `host_class_index` item is used as an index into the run-time constant pool of `M`. The symbolic reference at that index is resolved to a class or interface H (§5.4.3.1). **Then:**

  ~~During resolution of this symbolic reference, any of the exceptions pertaining to class or interface resolution can be thrown. Otherwise, resolution of H succeeds.~~

  ~~If any of the following is true, an `IncompatibleClassChangeError` is thrown:~~

    If resolution of this symbolic reference fails, then M is its own nest host. Any exception thrown as a result of failure of class or interface resolution is not rethrown.

    Otherwise, if resolution succeeds but any of the following is true, then M is its own nest host:

    - H is not in the same run-time package as `M`.
    - H lacks a `NestMembers` attribute.
    - H has a `NestMembers` attribute, but there is no entry in its `classes` array that refers to a class or interface with the name `N`, where `N` is the name of `M`.

    Otherwise, H is the nest host of `M`.