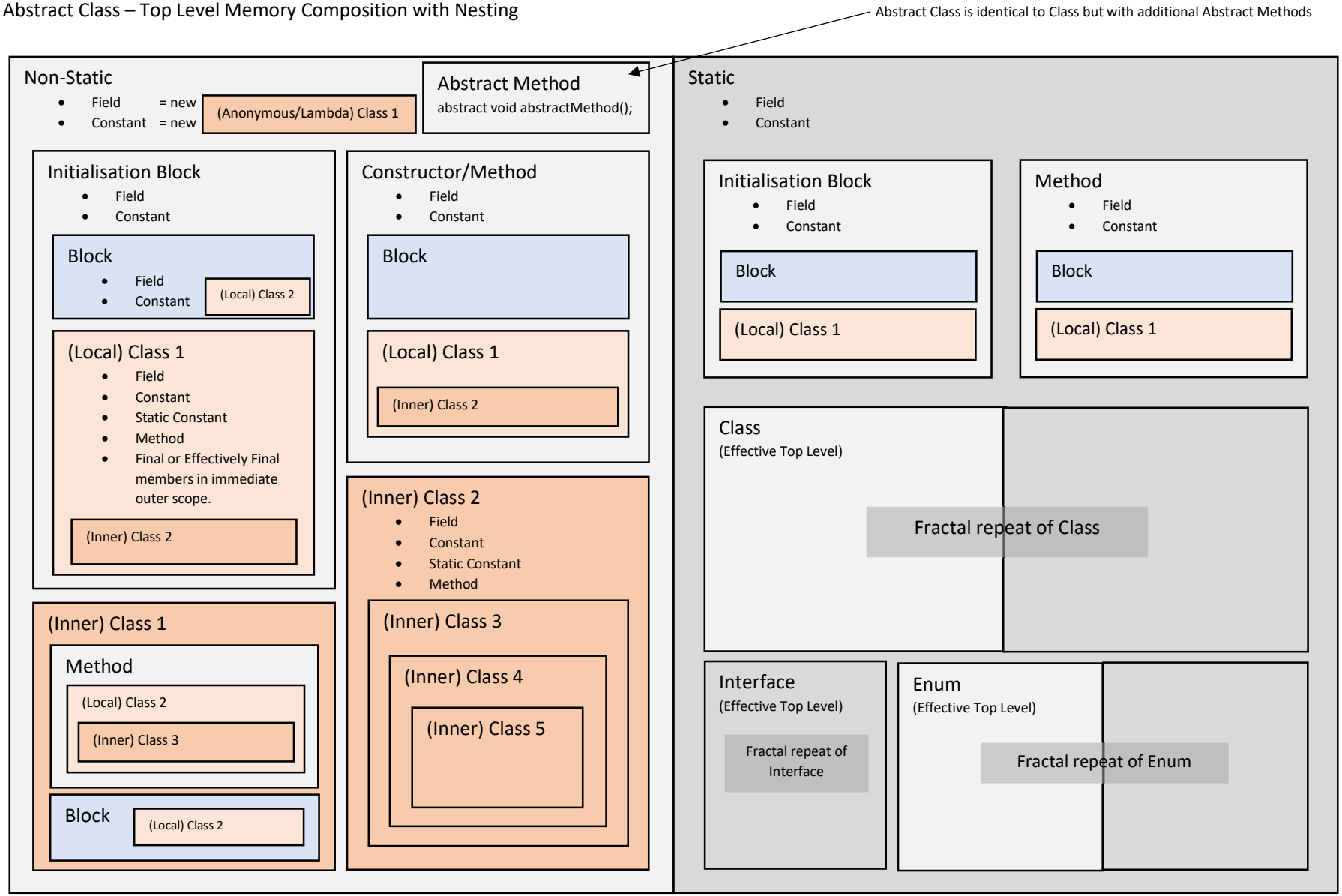
Paul Surridge

Abstract Class – Top Level Memory Composition with Nesting



Paul Surridge

Core Rules:

Please find some of the core rules of the nesting model in Java:

1. Top Level Forms are:

Class	Abstract Class
Interface	Enum

   Effective Top-Level Forms are top-level forms placed and nested within the static memory of a top level or another effective top-level form. They are identical in construct and behaviour with access to all static members back up to the file top-level form.

2. Abstract Class: In all circumstances wherever a class may be defined an abstract class may also be defined.

3. Permitted Memory: The memory composition and therefore the ability to declare and place members into static memory are as follows:

Memory Composition	Entity
Non-Static + Static:	Top-level forms and effective top-level forms
non-Static:	All other entities and nested forms (nested classes may declared static constants)
Static:	Interface

4. Permitted Members: The entities illustrate their permitted members and those with the same colour have the same permitted members.

5. Permitted Reference: The entities ability to reference members in an enclosing scope and nesting back up to the file top-level is governed by its location in top level memory:

Location*	Reference	
non-Static:	All	Reference all non-static and static members back up to top-level form**.
Static:	Static Only	Reference only static members back up to top-level form***.

   *Whether placed in non-static or static memory of the file top-level form.
   **Local class may only access final or effectively final parameters or variables within its immediate enclosing scope.
   ***Any nested form in static memory will only naturally have static members present/accessible within the nesting back up to top-level.

6. Default Placement: Interface/Enum are by default placed in static memory, therefore may only be defined within static memory, therefore may only be define within:

    1. File: Top level of a file.
    2. Nesting: Static memory of a top level or effective top-level form.

7. Location: A class is a class, a method is a method, a field is field regardless upon whether it is placed in either static or non-static memory.

Inner Class:	A class in non-static memory.
Local Class:	A class in non-static memory within a method/block.
Static Nested Class:	A class in static memory.
Method:	A method in non-static memory.
Static Method:	A method in static memory.
Field:	A field in non-static memory.
Static Field:	A field in static memory.

    There is no fundamental difference in substance nor construct, all entities are essentially the same but simply placed in either non-static or static memory of the surrounding form, which only affects their permitted reference (Rule 5).

8. Namespace: Entities (ex. blocks) have their own scope and namespace therefore local members may have the same name as other members in other scopes.

9. Shadowing All members shadow and hide members in any outer enclosing scopes (ex. Nested class, local class or blocks which do not permit shadowing).

10. Nested Classes: All nested classes (inner/local/anonymous/lambda) share the same:

    1. Permitted Members: Range and constraints on their permitted members.
    2. Static Constants: Do not have static memory but may declare their own static constants (ex. Lambda expressions).

11. File Top-Level The top-level of a file has the characteristics and behaviour of static memory, such that it is an environment where it is possible to define top-level forms just as it is only possible to define effective top-level forms in the static memory of another top-level form.