2009 General Scoring Guidelines

Apply the question-specific rubric first. The question-specific rubric *always* takes precedence.

Refer to the error categorization below for cases not already covered by the question-specific rubric.

Points can only be deducted if the error occurs in a part which has earned credit via the question-specific rubric.

A particular error is **penalized only once** in a question, even if it occurs on different parts of that question.

If a **minor error** (½ point) is the **only instance**, **one of two**, or occurs **two or more times**, then it **must be penalized as shown**.

A **minor error** that occurs **exactly once** when the same concept is **correct two or more times** is **not penalized** (regarded as an oversight).

Non-penalized Errors

spelling/case discrepancies*

local variable not declared if other variables are declared in some part

use keyword as identifier

`[]` vs. `()` vs. `<>`

= instead of == (and vice versa)

`length`/`size` confusion for array, `String`, and `ArrayList`, with or without `()`

`private` qualifier on local variable

extraneous code with no side-effect; *e.g., precondition check*

common mathematical symbols for operators (× • ÷ ≤ ≥ <> ≠)

missing { } where indentation clearly conveys intent and {} used elsewhere

missing ; where indentation clearly conveys intent and ; used elsewhere

default constructor called without parens; *e.g.*, `new Fish;`

missing `()` on parameter-less method call

missing `( )` around `if`/`while` conditions

missing `public` on class or constructor header

extraneous `[]` when referencing entire array

extraneous size in array declaration, *e.g.*, `int[size] nums = new int[size];`

Minor Errors (½ point)

confused identifier *(e.g.,* `len` *for* `length` *or* `left()` *for* `getLeft()` *)*

no local variables declared

missing `new` in constructor call

modifying a constant (`final`)

use `equals` or `compareTo` method on primitives, *e.g.*, `int x; …x.equals(val)`

array/collection access confusion (`[] get`)

assignment dyslexia, *e.g.*, `x + 3 = y;` *for* `y = x + 3;`

`super(method())` instead of `super.method()`

formal parameter syntax (with type) in method call, *e.g.*, `a = method(int x)`

missing { } where indentation clearly conveys intent; {} not used elsewhere

missing ; where indentation clearly conveys intent and ; not used elsewhere

missing `public` from method header when required

`"false"`/`"true"` or 0/1 for boolean values

`"null"` for `null`

Major Errors (1 point)

extraneous code which causes side-effect; *e.g., information written to output*

use interface or class name instead of variable identifier; *e.g.*, `Simulation.step()` *instead of* `sim.step()`

use `this` within static method

`aMethod(obj)` instead of `obj.aMethod()`

use private data or method when not accessible

destruction of data structure *(e.g., using root reference to a* `TreeNode` *for traversal of the tree)*

use class name in place of `super` either in constructor or in method call

`void` method (or constructor) returns a value

**Note: Spelling and case discrepancies for identifiers fall under the "non-penalized" category only if the correction can be unambiguously inferred from context. For example, "Queu" instead of "Queue" . Note, however, that if a student declares "Fish fish;", then uses Fish.move() instead of fish.move(), the context does not allow for the reader to assume the object instead of the class.*