

This exam scores code against specific behaviours: the loop bound, the return, the signature. Everything below is a rule the readers apply.

How the readers score free-response code

✗ Writing an outline of the method in comments.

✓ **Write compilable Java. Structure must clearly convey intent.**

The guidelines award points where the structure conveys intent – a comment describing intent is not code.

SG 8 years

✗ Printing the result where the signature says return.

✓ **Return the value. If the method is void, print; otherwise do not.**

A method that prints instead of returning does not do what the signature promises, whatever the logic inside.

`return` `print`

CED 1.7

✗ Changing the method signature to something easier.

✓ **Use the header exactly as given, with the same parameter types and order.**

The header is part of the question, and a different one cannot be called by the tests the readers imagine.

CED 3.4

✗ Declaring a new array and copying nothing into it.

✓ **Copy the elements across; do not remove or overwrite the original.**

The guidelines withhold the point from responses that remove or overwrite data instead of copying it.

SG2023

✗ Adding a semicolon or a separator at the end of a built string.

✓ **Build it so the final element carries no trailing separator.**

The guidelines state responses do not earn the point if they end the constructed output with a separator.

SG2023

■ A response can still earn later points after an earlier error. Write every method, even when one has defeated you.

SG 3 years

2 · Selection and Iteration

✗ Looping to '`<= length`'.

✓ **Valid indices run from 0 to '`length - 1`', so the condition is '`< length`'.**

The off-by-one throws an out-of-bounds exception, and the guidelines withhold the point for bounds errors.

`0` `length-1`

SG 2 years

✗ Stopping the loop before the last element.

✓ **Check that the final element is visited – the guidelines require ALL elements to be accessed.**

Missing the last element is invisible in most test data and fails the criterion outright.

SG2023

✗ Reversing the loop bounds when traversing a 2-D array.

✓ **Rows outer, columns inner – and keep the same order in every nested loop.**

The guidelines list reversing the loops among the responses that do not earn the point.

SG2023

✗ Removing from an ArrayList inside a forward loop.

✓ **Loop backwards, or adjust the index after each removal.**

Each removal shifts the remaining elements, so a forward loop skips one every time.

CED 4.4

1 · Using Objects and Methods

✗ Comparing Strings with ‘==’.

✓ **Use ‘equals()’ – ‘==’ compares references, not contents.**

The comparison silently returns false for equal strings built separately.

equals

CED 1.5

✗ Writing ‘array.length()’ or ‘string.length’.

✓ **An array has ‘length’; a String has ‘length()’.**

The two differ by a pair of brackets and neither compiles as the other.

length, length()

CED 1.4

✗ Assuming integer division gives a decimal.

✓ **‘7 / 2’ is 3. Cast to double when you need 3.5.**

The truncation is silent and propagates through every later calculation.

CED 1.3

✗ Calling an instance method as though it were static.

✓ **Create the object first, then call the method on it.**

The distinction is course content and the call does not compile.

CED 1.6

3 · Class Creation

✗ Assigning constructor parameters to local variables instead of the instance fields.

✓ **Assign to ‘this.field’, so the object keeps the value.**

The object is then constructed with default values and every later method returns the wrong thing.

CED 3.3

✗ Making every field public to save writing accessors.

✓ **Keep the fields private and provide the accessors the question names.**

Encapsulation is assessed content in this unit.

private

CED 3.6

✗ Calling a superclass method with the wrong name or arguments.

✓ **Use ‘super.method()’ with the signature the parent declares.**

The guidelines allow other points to survive an incorrect ‘super’ call, but that point is lost.

super

SG2023

4 · Data Collections

✗ Declaring an ArrayList of primitives.

✓ **Use the wrapper type: ‘ArrayList<Integer>’, not ‘ArrayList<int>’.**

A collection holds objects, and the primitive form does not compile.

CED 4.2

✗ Comparing 2-D array dimensions in the wrong order.

✓ **‘arr.length’ is the number of rows; ‘arr[0].length’ is the number of columns.**

Swapping them produces bounds errors on any non-square array.

arr.length

CED 4.5

■ Trace your method by hand on the example the question gives before writing the next one. Most lost points are logic that was never tested against the sample.

CED 2.6

✗ Using ‘get()’ and ‘add()’ on an array.

✓ **Arrays use ‘[i]’; only an ArrayList uses the methods.**

The two collections have different interfaces and neither compiles as the other.

,ArrayList

CED 4.3

✗ Adding to a list while looping forward over it.

✓ **Build a second list, or loop backwards.**

Adding shifts the indices ahead of the cursor, which can loop forever.

CED 4.4

✗ Assuming an enhanced for loop can modify the elements.

✓ **It gives a copy for primitives; use an indexed loop to write back.**

The assignment inside the loop is silently discarded.

for

CED 2.7

2 · Selection and Iteration

✗ Writing ‘if (x = 5)’ in a condition.

✓ **‘==’ compares; ‘=’ assigns.**

In Java this does not compile for ints, but it is the habit that produces the wrong condition.

==, =

CED 2.2

✗ Chaining ‘else if’ where each condition should be tested.

✓ **Decide whether the cases are exclusive; if not, use separate ‘if’ statements.**

An ‘else if’ chain stops at the first match, which is right only for exclusive cases.

else if

CED 2.3

✗ Building a ‘while’ loop whose variable never changes.

✓ **Update the loop variable inside the body, before the next test.**

The infinite loop times out and earns nothing for the logic that was otherwise correct.

while

CED 2.5

✗ Testing a boolean with ‘if (flag == true)’.

✓ **Write ‘if (flag)’.** It is shorter and cannot be mistyped as an assignment.

Structure that conveys intent is what the guidelines reward.

CED 2.2

3 · Class Creation

✗ Overloading a method and expecting it to override.

✓ **Overriding needs the SAME signature; a different parameter list overloads instead.**

The wrong method is then called at runtime, with no compile error.

CED 3.5

✗ Writing a ‘toString’ that prints instead of returning.

✓ **‘toString’ returns a String; the caller decides whether to print it.**

The same return-versus-print rule that catches candidates in every other method.

toString

CED 3.7

■ Read every method header the question supplies before writing any of them. Later parts usually call earlier ones, and the signatures tell you how.

CED 3.4