

Learn these by heart before the exam.

Primitives, objects, and operators

Primitives: *int*, *double*, *boolean*, *char*. Everything else is a *reference type* (*String*, *arrays*, *ArrayList*, *your own classes*).

int / int ⇒ int 7/2 is 3, not 3.5. Cast first: *(double)*7/2 gives 3.5

% *x*%2 == 0 tests even; *x*%10 takes the last digit; *x*/10 removes it

Operator precedence: unary, then * / %, then + −, then relational, then &&, then ||, then assignment. Compare objects with *.equals()*, NEVER with ==. On *Strings* == compares references and silently fails.

Integer overflow wraps around. Division by zero throws for *int* but gives *Infinity* for *double*.

Selection and Iteration

Short-circuit: *a&& b* skips *b* if *a* is false. Use it to guard: if (*arr != null && arr.length > 0*). De Morgan: *!(a && b)* equals *!a || !b*; *!(a || b)* equals *!a && !b*.

for (*int i* = 0; *i* < *n*; *i*++) runs *n* times. Off-by-one is the commonest exam error: check the first and last iteration by hand.

Enhanced for (for (*int x* : *arr*)) reads only. You cannot use it to *WRITE* into the array.

while runs zero or more times; do-while runs at least once.

O(*n*²) a loop inside a loop over the same *n*; state this when asked to count statements

Class Creation

Encapsulation: *private* fields, *public* methods. Accessor (*getter*) returns; mutator (*setter*) changes. Constructor has the class name and *NO* return type. *this.field* = field disambiguates a shadowed parameter. Static belongs to the *CLASS* (one copy, shared); instance belongs to the *OBJECT*. A static method cannot touch instance fields.

Inheritance: extends. *super()* calls the parent constructor and *MUST* be the first statement.

Overriding replaces a parent method with the same signature. Overloading is the same name with *DIFFERENT* parameters.

Polymorphism: the declared type decides what compiles; the *ACTUAL* object decides which override runs at runtime.

toString() is called automatically by

System.out.println. Override *equals()* to compare by value.

Data Collections

arr.length vs **str.length()** vs **list.size()** array is a *FIELD* (no brackets), *String* is a method, *ArrayList* uses *size()*. A classic trap

Array is fixed length; *ArrayList* grows. *ArrayList* holds objects only, so use *Integer*, not *int* (*autoboxing* does it for you).

ArrayList: *add(x)*, *add(i,x)*, *get(i)*, *set(i,x)*, *remove(i)*, *size()*. *remove* *SHIFTS* everything left.

Removing while looping forward *SKIPS* elements. Loop *BACKWARDS*, or do not increment after a removal.

2-D array: *arr[row][col]*. *arr.length* is the number of *ROWS*; *arr[0].length* the number of columns.

String: *substring(a,b)* excludes *b*; *indexOf* returns -1 if absent; *compareTo* is negative/zero/positive. *Strings* are *IMMUTABLE*.

Traversals: sequential search is *O*(*n*); binary search is *O*(log *n*) but needs a *SORTED* array.

Selection and insertion sort are *O*(*n*²); merge sort is *O*(*n* log *n*) and recursive.

Recursion

Every recursion needs a *BASE CASE* and progress toward it. Without a base case: *StackOverflowError*.

Trace by hand: write each call and its return value on a separate line, then unwind from the base case.

T(*n*) = *T*(*n* − 1) + *O*(1) merge sort splits in half each time, hence *O*(*n* log *n*)

Common exceptions

NullPointerException: calling a method on *null*. *ArrayIndexOutOfBoundsException*: *index* < 0 or ≥ *length*. *StringIndexOutOfBoundsException* from *substring* with bad bounds. *ArithmeticException* from integer division by zero.

ClassCastException from casting to the wrong type. Check with *instanceof* first.

Exam technique

Free-response: write real, compilable Java. No pseudocode, no comments instead of code. Use the exact method names given.

You may call methods from earlier parts even if you

did not write them. Assume they work.
Do not change the given signature and do not add parameters. Read the preconditions: they tell you what you need not check.
Trace questions: keep a running table of every variable. Guessing the output loses more marks than slow tracing.