CSSE2002 · PROGRAMMING IN THE LARGE

Complete Exam Revision Notes

Every examinable idea, explained plainly, with original Java 21 examples, worked test tables and a plan for your one A4 sheet.

Built for: the current Java 21 version of the course (2026 onward)

Organised by: the question types that keep coming back in the final exam

Every code example: compiled and run on Java 21. The output printed under each one is the real output

Written by: a UQ mechatronics engineering graduate who took CSSE2002 and tutored first-year engineering at UQ

Independent study notes. Not affiliated with or endorsed by The University of Queensland. All explanations and code are original. Course details (weightings, exam rules) are taken from the public course profile and can change, so check yours.

How to use these notes

The final exam is worth 55% and it has a graded hurdle, so it decides your grade more than anything else in the course. These notes are built to get you through it.

The exam at a glance

What	Detail
Weight	55% of the course
Time	120 minutes plus 10 minutes planning
Allowed in	One A4 sheet, both sides, typed or handwritten. An approved calculator.
Question types	Multiple choice, short answer, calculations and programming tasks (writing code on paper)
Coverage	The whole semester

The hurdle: your exam mark caps your grade

Final grade	3	4	5	6	7
Minimum exam mark	35%	40%	55%	65%	75%

So strong assignments can't carry a weak exam. If you are aiming for a 7, you need 75% in the exam room with only your sheet.

How each section works

How it's examined	What the question usually looks like and what the marker wants to see
Trap	The mistakes that lose marks most often
Put this on your sheet	Lines worth the space on your A4 sheet (section 15 pulls them into one plan)

Contents

1 Java essentials and type traps
2 Code reading: tracing loops and arrays
3 Classes, encapsulation and invariants
4 Specifications and contracts
5 Inheritance, polymorphism and declared vs actual type
6 Interfaces, abstract classes and records
7 The substitution principle
8 Exceptions
9 equals, hashCode and ordering
10 Collections and generics
11 File I/O and parsing
12 Testing: black box, white box, JUnit
13 Loop invariants, correctness and recursion
14 Design, good practice and refactoring
15 Your A4 sheet plan
16 Exam-day checklist

A study plan that works

1. **Read a section, then close the notes and redo its example on paper.** The exam has no compiler, so practise writing code by hand.
2. **Do every "How it's examined" task without looking.** Then check against the worked answer.
3. **Build your sheet as you go.** Copy the green boxes in your own words. Writing it is half the revision.
4. **Finish with past papers under time.** Divide the exam minutes by its total marks to get your pace per mark.

1 Java essentials and type traps

Short-answer and multiple-choice questions love the corners of Java's type system. Most of them come down to five rules.

Primitive types vs reference types

Java has eight **primitive** types that hold a value directly: `byte`, `short`, `int`, `long`, `float`, `double`, `char` and `boolean`. Everything else is a **reference** type: the variable holds an arrow to an object on the heap, not the object itself.

- **Assigning a primitive copies the value.** Assigning a reference copies the arrow, so two variables can point at one object.
- **Java always passes arguments by value.** For objects the value passed is the reference. A method can change the object you passed in, but it can't make your variable point somewhere else.
- **`==` on references asks "same object?"** Use `.equals()` to ask "same contents?"
- **Fields get default values** (0, false, null). Local variables don't: using one before assigning it is a compile error.

Arithmetic rules that catch people

- **Integer division truncates toward zero**: `7 / 2` is 3 and `-7 / 2` is -3. `%` keeps the sign of the left side.
- **The type is decided before the assignment.** `double b = 7 / 2;` gives 3.0 because the division is done in `int` first.
- **Overflow is silent.** `Integer.MAX_VALUE + 1` wraps to the most negative int.
- **`char` is a number.** `'A' + 1` is the int 66. Once a String is on the left, `+` means join.

TYPES.JAVA

```
public class Types {
    public static void main(String[] args) {
        int a = 7 / 2;          // int / int stays int
        double b = 7 / 2;       // divided first, widened after
        double c = 7 / 2.0;     // one double makes it double maths
        System.out.println(a + " " + b + " " + c);

        int big = Integer.MAX_VALUE;
        System.out.println(big + 1);              // wraps around silently

        System.out.println(-7 / 2 + " " + -7 % 2); // truncates toward zero

        String s1 = "hi";
        String s2 = new String("hi");
        System.out.println((s1 == s2) + " " + s1.equals(s2));

        char ch = 'A';
        System.out.println(ch + 1);       // char + int is an int
        System.out.println("" + ch + 1);  // String first, so it joins
    }
}
```

OUTPUT

```
3 3.0 3.5
-2147483648
-3 -1
false true
66
A1
```