CSSE2002 Diagram & Notation Conventions

Table of contents

How this note was built . . . 1
1. The page frame every figure sits in . . . 2
2. Typeface roles . . . 3
3. What colour means . . . 3
4. Class-hierarchy diagrams — the dominant style . . . 4
5. UML-style class boxes (lecture 02 only) . . . 5
6. Dependency / interface diagrams — the only figures with a legend . . . 5
7. The pre/postcondition "funnel" figure . . . 6
8. Coupling and cohesion figures . . . 6
9. The coupling-levels ladder . . . 7
10. Annotating code . . . 7
11. Code blocks . . . 8
12. Specification / Javadoc notation . . . 8
13. Arrays and stacks . . . 9
14. Black-box / glass-box testing icon . . . 9
15. Attribution and textbook lineage . . . 9
16. Naming and capitalisation . . . 10
Inconsistencies . . . 10
Not established by the source . . . 12

The house style for every figure drawn in CSSE2002 — inferred from the lecture decks themselves, because **this course ships no written notation guide**.

How this note was built

The anchor manifest for csse2002 lists **12 lecture decks, 656 pages, and an empty `guides` array**. There is no style guide, no drawing standard, no "how to draw a class diagram" handout. Every convention below is *inferred from the drawings*, not quoted from a rule. Where a convention rests on a single figure, the entry says so.

Pages actually read

I read **210 of the 656 pages** as rendered images, sampled to go wide across the semester and to land on pages that carry drawings rather than bullets:

Deck	Pages read	of
L01a Course Overview	18–27	62
L01b Java Basics, Collections & Strings	31–42	77
L02 Object Oriented Programming I	1–20, 31–45	54
L03 OOP II & Specification	14–31	62
L04 Exceptions & Specification	6–36	49
L05 Testing	3–14	42
L06 Refactoring, Cohesion & SOLID I	15–32	52
L07 SOLID Principles II	14–30	39
L08 Java I/O	3–16	50
L09 Generics in Java	19–33	46
L11 Lambdas, Streams & Events	20–32	34
L13 Course Summary & Exam Prep	26–40	89

I did **not** read: L01a 1–17 & 28–62, L01b 1–30 & 43–77, L02 21–30 & 46–54, L03 1–13 & 32–62, L04 1–5 & 37–49, L05 15–42, L06 1–14 & 33–52, L07 1–13 & 31–39, L08 1–2 & 17–50, L09 1–18 & 34–46, L11 1–19 & 33–34, L13 1–25 & 41–89. A few citations below (textbook mentions, "Further Reading" slides) come from **extracted text on pages I did not view as images**; those are marked *(text only)*. Font names come from `pdffonts`; hex colours were sampled from rendered pixels, not from any stated palette.

Citations use the form **L07 p29** = the deck for lecture 07, PDF page 29.

---

1. The page frame every figure sits in

Template A — the course template. White ground. Slide title in UQ purple **#4E247F**, set in **Tw Cen MT** (a geometric sans), left-aligned at roughly x = 7% and y = 15%. An optional second title line sits directly beneath in smaller magenta **#9D0792 / #A02B93** Tw Cen MT. Body prose is **Bell MT** (a serif) in near-black **#171717**, round bullets, up to four indent levels with decreasing size. UQ crest bottom-left; solid purple **CSSE 2002** badge bottom-right; page number immediately left of the badge in small grey serif.

Cited: L02 p2, p14; L04 p12; L06 p20; L07 p14; L08 p5; L09 p33; L13 p39. **Holds across 11 of the 12 decks, weeks 1–13.**

Template B — the guest deck. L11 alone uses a completely different template: **Raleway Bold** black titles, **Source Sans Pro** black body, hyphen bullets, teal underlined links, occasional full-bleed purple block, and **no crest, no badge, no page number**. *Cited:* L11 p20, p23, p24, p28, p31. Seen only in L11 — see Inconsistencies.

Section dividers. Purple Tw Cen MT heading alone, placed low-left or low-centre, everything else blank. *Cited:* L02 p34, p43, p44; L06 p16, p19, p26; L09 p20, p23, p32; L13 p31, p32, p38. **Throughout.**

Deck title slides. Centred stack: deck title in purple Tw Cen MT → "Programming in the Large (CSSE2002)" in purple Bell MT → "Dr. Thilina Halloluwa" in black Bell MT → "Week NN – Part NN" in black Bell MT. *Cited:* L02 p1; L04 p19; L06 p15. **Consistent wherever a title slide appears.**

2. Typeface roles

Role	Face	Evidence
Slide titles, section headings	Tw Cen MT, purple	everywhere in Template A
Prose, captions, labels on the lecturer's own diagrams	Bell MT (serif)	L03 p27; L06 p21; L07 p16
Java source; Java identifiers inline in prose; text inside class/dependency boxes	**Inconsolata** (hand-set) or **Courier New** (pasted)	L02 p32, p36; L07 p27; L08 p6
Borrowed / stock figures	Arial or similar sans	L02 p15; L06 p17; L08 p3; L09 p27

Inline Java identifiers in serif prose are always switched to monospace: `byte`, `char`, `BufferedInputStream`, `java.io.InputStream`, `instanceof`, `? extends T`. *Cited:* L03 p29; L08 p15; L09 p24, p30. **Very consistent.**

3. What colour means

Colour	Sampled hex	Meaning	Evidence	Consistency
UQ purple	`#4E247F`	chrome, slide titles; nothing semantic inside a figure	everywhere	throughout

Colour	Sampled hex	Meaning	Evidence	Consistency
Teal	`#156082`	**every relationship line, connector and pointer the lecturer draws**	L03 p27; L04 p7; L06 p18, p20, p21; L08 p7, p11, p14; L09 p25, p29; L13 p27	strongest colour convention in the course — weeks 3–13
Magenta	`#9D0782` – `#A02B93`	subtitle line; a term being defined; a verdict on an example	L02 p35; L04 p27, p30; L05 p5; L08 p5, p8; L09 p21, p27, p30	very consistent
Pure red	`#FF0000`	a hard rule to memorise; the outline of a code-highlight box; deadlines	L01a p24; L02 p38, p40; L03 p28; L04 p14; L07 p15, p20; L08 p16	consistent for "stop and look"
Green	`#3B7D23` (Javadoc), bright green (chart)	Javadoc/comment text in code; passing tests in the one chart	L04 p27; L05 p5	chart usage is a single instance
Near-black	`#171717`	prose, box outlines, arrows in *borrowed* figures	L05 p6, p8; L07 p26	throughout

Nothing anywhere in any deck states what a colour means. Teal-for-relations and magenta-for-defined-terms are inferred from repetition alone.

Cube outline colours in the coupling/cohesion figures (purple `#7030A0`, red `#FF0000`, green `#196B24`, cyan `#00B0F0`, orange) carry **module identity only** — the same colour means "same module", not any property of it. *Cited:* L06 p18, p20, p27; L07 p24.

4. Class-hierarchy diagrams — the dominant style

Style H1 (dominant). Node = rectangle, thin border, white fill, class name centred in **Bell MT serif** (not monospace). Superclass sits **above**. Connector is a **teal orthogonal polyline**: one vertical stub rising into the parent's bottom edge carrying a **single solid teal arrowhead**, a horizontal teal bus, and vertical drop-legs to each child. **One arrowhead per parent, not per child.** No edge labels, no multiplicities, no stereotypes.

Cited: **L04 p7** (`Throwable → Error, Exception → RuntimeException, IOException`), **L08 p7** (`OutputStream` tree), **L08 p11** (`InputStream` tree). Three figures, two decks, weeks 4 and 8. This is the closest thing the course has to a settled hierarchy notation — but see Inconsistencies for two rival styles.

5. UML-style class boxes (lecture 02 only)

Three compartments separated by thin horizontal rules: **name / attributes / operations**. Thin black border, white fill, **monospace** throughout. Name centred and bold-ish; attribute and operation lines left-aligned.

- Visibility prefix: `-` private, `+` public. No `#` or `~` appears anywhere.
- Attribute line: `- name : String`, `-hourlyRate: double`.
- Operation line: `+getName():String`, `+logHours(hours:int) : void`. Parameters written `name:Type`, return type after the closing paren's colon.
- **Constructors live in the operations compartment**, written `+Employee (name:String, address:String)` with no return type.
- Generalisation: **solid black line, unfilled triangle arrowhead at the superclass**, orthogonal routing, superclass above.

Cited: L02 p12, p13 (two-compartment variant), **L02 p36** (the full hierarchy — the canonical figure), L02 p37, p38, p40.

This notation appears in lecture 02 and nowhere else I read. L09 p21 shows a small corner decoration in a similar spirit, but it is a borrowed graphic in a different palette (pale blue-grey fill, sans labels, `+ name : string` with a lowercase type). Do not treat L09 p21 as course style.

6. Dependency / interface diagrams — the only figures with a legend

These are the only diagrams in the course that **draw their own key on the slide**, so this is the most reliable convention in the note.

- **Node** = plain rectangle, thin black border, white fill, monospace label that **includes the Java keyword**: `class A`, `class B`, `class Z`, `interface AI`, `interface BSpec`.
- **Members** are listed inside the same box, one per line, as `+ a()` — with a space after the `+`, and **no compartment rule** separating them from the name.
- **Legend, drawn at the right of the slide:**
  - solid line + solid arrowhead → **"Uses"**
  - dashed line + open arrowhead → **"Implements"**

- **Direction:** an arrow always points **from the depender to the thing depended upon**. `class A` `interface AI` = A uses AI. `class Z` — the dashed implements arrow runs **from the implementing class to the interface** (L07 p27, p29, p30).
- **Layout:** clients left, abstractions centre, concrete classes right (L07 p26, p27); or dependents at the bottom with the abstraction above (L07 p29, p30). Flow is left→right for "uses" and bottom→top for "implements".
- An arrow may terminate on an individual **member row** inside a box, not only on the box edge (L07 p26).

Cited: L07 p26 (legend shows "Uses" only), **L07 p27, p29, p30** (full legend). Four figures, one deck, week 7.

Note the divergence from standard UML: UML draws realisation as a *dashed line with a hollow triangle* and dependency as a *dashed open arrow*. This course draws implements as a dashed **open arrowhead** and uses as a **solid** line with a solid head. Follow the course, not UML — but be aware you are diverging.

7. The pre/postcondition "funnel" figure

A distinctive, course-specific idiom for reasoning about the Liskov Substitution Principle:

- A method is a small **square box, thin black outline, monospace name** (`A.f`, `B.f`).
- **Above** it, a cone that narrows downward into the box. Its wide upper ellipse is the set of inputs the method accepts — the **precondition**.
- **Below** it, a cone that widens downward out of the box. Its wide lower ellipse is the set of outcomes it may produce — the **postcondition**.
- **Width = weakness.** A *wider* top mouth = a *weaker* (more permissive) precondition. A *wider* bottom mouth = a *weaker* (less committal) postcondition. So a legal subclass override has a top mouth **at least as wide** and a bottom mouth **no wider** than its parent's.
- Labels `preconditions` and `postconditions` in **lowercase serif** at the far left of the slide, vertically aligned with the two mouths.
- Entirely **black line art, unfilled, no colour**.
- To compare parent and child, the child's cones are **overlaid on the parent's** in the same frame in thinner/greyer stroke (L07 p17).

Cited: **L07 p16** (A.f vs B.f side by side), **L07 p17** (overlaid), L07 p18 (reused as a thumbnail). Three pages, one deck.

8. Coupling and cohesion figures

- **A module is an isometric wireframe cube** — outline only, no fill, hidden edges dashed. One colour per module, identity only.

- **Coupling is a teal double-headed arrow** between cubes. High coupling = many crossing arrows; low coupling = each cube attached by one short teal double arrow to a **horizontal grey bar** standing for a shared interface.
- **Cohesion** uses the same cubes flattened. Low cohesion = colours scattered; high cohesion = same-coloured cubes grouped inside a rectangle **outlined in that same colour**.
- **Comparison layout: the bad case on the LEFT, the good case on the RIGHT**, each labelled with one serif word beneath (`High/Low`, `Low/High`).

Cited: L06 p18 (tangled), **L06 p20** (High vs Low coupling), **L06 p27** (Low vs High cohesion); reused verbatim in **L07 p24**. Two decks, weeks 6–7.

9. The coupling-levels ladder

Seven identical rectangles — thin black border, white fill, serif label — stacked vertically with even gaps. Ordered **most-coupled at the top** (`Content`) down to **least at the bottom** (`None`). A single long thin **teal arrow points downward** to the right of the stack. Its endpoints are captioned in magenta serif with a parenthetical gloss: "Tightly Coupled (Needy)" at the top, "Loosely Coupled (Independent)" at the bottom.

Cited: **L06 p21**; reused verbatim **L07 p25**. Two decks.

10. Annotating code

- **Highlight a token:** a thin **red `#FF0000` rectangle** drawn around it. *Cited:* L02 p38, p40; L03 p28; L04 p14; L08 p16. Dominant across four decks.
- **Point at a token:** a thin **teal straight or diagonal line with a small solid arrowhead**, running between the token and a **serif caption placed outside the code block, unboxed**. *Cited:* L02 p16; L03 p24, p28; L08 p14; L09 p25, p29; L13 p27. Weeks 2–13 — very consistent.
- **Show a replacement: strike through** the token being removed and put the replacement beside it in red. *Cited:* L02 p4; L07 p20; L13 p27.
- **Mark an error inline:** a red squiggly/dashed underline, IntelliJ-style. *Cited:* L09 p22.
- **Aside / rule-to-remember:** a short sentence centred beneath the content in serif — sometimes inside a thin rectangle, sometimes bare. *Cited (boxed black):* L02 p31; L07 p24; L08 p5; L09 p26. *Cited (boxed teal):* L03 p21. *Cited (unboxed):* L03 p23; L06 p31; L08 p12, p15; L07 p14, p25. Inconsistent — see below.
- **Yellow highlighter** behind a heading inside a worked-example box. *Cited:* L07 p19 (single instance).
- **Verdict on an example**, placed in the right margin in magenta serif: "Bad Example", "Better / Still Bad ", "Does not satisfy specification!", "Satisfies specification!". *Cited:* L04 p26, p27, p28, p29, p30, p31. Consistent within L04.

11. Code blocks

Java is shown as **light-theme IntelliJ IDEA output**, confirmed by the sampled hex values: keywords `#0033B3` blue, fields/constants `#871094` magenta, strings and Javadoc `#3B7D23` green, comments grey italic, annotations (`@Override`, `@Test`, `@Before`) olive/mustard, numeric literals blue. *Cited:* L02 p4, p16; L04 p22; L06 p24; L09 p22.

Some slides show real IntelliJ inlay hints (`3 usages`, `accountNumber:`) and even an editor error popup, i.e. they are screenshots rather than set text. *Cited:* L02 p5, p6, p9, p20.

Container: either bare on white, or inside a **thin black rectangle**. Two files at once are shown as **two overlapping black rectangles**, z-order indicating which is in front. *Cited:* L02 p39, p41, p42; L03 p16; L04 p13, p18; L07 p18.

A **trace** puts line numbers in the left margin in monospace and expected results in a right-aligned column of `//` comments. *Cited:* L01b p40, p41.

12. Specification / Javadoc notation

The course's specification "algebra" lives inside Javadoc comments, not in a separate formalism.

- Javadoc opens `/**` and closes `*/`, sits immediately above the thing documented, and tags begin with `@`. *Cited:* L04 p21.
- **The tag table** (L04 p23) is set as a two-column list with no rules: tag name in monospace, its argument placeholder in *italic serif*, gloss in upright serif.

Tag	Argument	Meaning (as printed)
`@param`	*varname …*	Describe a parameter for a method
`@return`	…	Describe the return value
`@throws`	*ExceptionType …*	Describe when a particular exception is thrown
`@requires`	*Precondition*	Specify the assumptions for a method to execute properly
`@ensures`	*Postcondition*	Specify the effects of executing the method
`@author`	*authorname*	Specify the author of the class
`@copyright`	…	Provide details on who owns copyright

- **`\result`** denotes the returned value inside a postcondition. *Cited:* L07 p18, p19, p20.

- Conditions are written in **Java boolean syntax**, not mathematical notation: `row != null && col != null && 0 < val && val < 100`. No , , appears anywhere I read. *Cited:* L07 p18; L13 p36.
- `{param}` braces denote a parameter's value inside prose in a tag: `@return {first} followed by {second}`. *Cited:* L13 p36 (single instance).

13. Arrays and stacks

Array. A horizontal row of equal-width square cells sharing thin black borders; values centred; **no index row**. An empty cell is a hole. A **grey ghosted digit** is a value being shifted out. `?` is an unspecified value. A **purple curved arrow** drawn above the row = one shift-left step. *Cited:* L01b p31–p37 (a seven-frame build). A second, smaller variant puts the array's name in monospace immediately to the left of the row and sets the values in serif: L04 p27.

Stack. *Cited:* L01b p39 — an **open-topped U-shaped container** in bright purple `#6B088F` (no top edge), entries as purple-filled rounded rectangles with white bold numerals, **growing upward** so the top of stack is at the top. A grey curved arrow **into** the container = push; a grey curved arrow **out and away** = pop. Each frame is captioned in monospace with the operation (`empty()`, `push(1)`, `pop()`), and frames read **left→right as time**.

14. Black-box / glass-box testing icon

A **grey-filled isometric cube, unlabelled**, with a **black horizontal arrow entering on the left** labelled `input` in serif above the line, and a matching arrow **leaving on the right** labelled `output`. Flow is strictly left→right. The glass-box variant is the same cube rendered translucent with the source code visible inside it. *Cited:* **L05 p6** (both cubes together), L05 p8 (black-box cube reused as a corner marker).

15. Attribution and textbook lineage

No figure anywhere I read carries a "Figure from ..." caption, and no single textbook governs the drawing style. The course cites books for *content*, never for *notation*.

Placement convention: a source line sits at the **top of the slide, right-aligned, small serif, no box and no rule**.

- "Effective Java (2017) - Item 75: Include failure-capture information in detail messages" — **L04 p8**
- "Java: The Complete Reference, Eleventh Edition- Herbert Schild" — **L08 p4**
- "*Effective Java*, Item 31: parameterized types are invariant; `List<Type1>` is neither a subtype nor supertype of `List<Type2>`" — **L09 p28**
- "Effective Java (2017, Addison-Wesley) - Item 9" — **L08 p20** *(text only)*

- Bare URLs in the same position: L01b p38; L05 p5, p9; L08 p9, p10; L09 p26.

Borrowed figures carry an adjacent link instead: "Source : Javatpoint" in small serif at the figure's lower right (**L08 p3**); a bare hyperlink reading just "source" centred beneath the figure (**L06 p28**).

A book cover can stand in for a citation: L03 p15 shows the *Effective Java* **2nd edition** cover as the whole visual for the slide.

Books named in prose *(text only — pages not viewed as images)*: Martin Fowler, *Refactoring: Improving the Design of Existing Code* (2nd edition, 2019) and Robert Martin, *Clean Code* at **L06 p14**; Robert C. Martin, *Design Principles and Design Patterns* at **L06 p35 & p50, L07 p3, L07 p38**; *Effective Java* items at **L03 p18, p22** and **L04 p47**.

16. Naming and capitalisation

- Java identifiers follow standard Java: `PascalCase` types, `camelCase` methods and fields, `SCREAMING_SNAKE` constants, always monospace. **Consistent everywhere.**
- Running example names: `Person`, `Employee`/`ContractEmployee`/ `FullTimeEmployee`, `Animal`/`Dog`/`Cat`, `BankAccount`/`AccountHolder`, `Foo`, `Printer`, and for abstract structural examples `class A`/`B`/`C`, `class Z`, `interface AI`/`BI`/`CI`, `interface BSpec`.
- **Interfaces in structural diagrams are named either `<Letter>I`** (`AI`, `BI`, `CI` — L07 p27) **or `<Class>Spec`** (`BSpec` — L07 p29, p30). Both in the same deck.
- Method-signature shorthand in prose uses UML order — `getType():String`, `addListener(String type, Consumer<SimpleEvent> action): action` (L11 p28) — while diagram boxes use the same order (L02 p36). Consistent.
- A right arrow `→` in serif means "resolves to" / "dispatches to": `kapow(Object) → locked in`. *Cited:* L13 p26, p28.
- ASCII `->` between monospace type names means "widens to" in the casting chains: `byte -> short -> char -> int -> long -> float -> double`. *Cited:* L03 p26.

Inconsistencies

Where the course does the same thing more than one way. **A decision is needed on each of these before drawing a new figure — I am recording both, not choosing.**

1. **Three different ways to draw inheritance.**
    - *H1* — black-outlined box, **serif** class name, teal orthogonal connector, plain solid teal arrowhead into the parent, no edge label: **L08 p7, L08 p11**.

- *H2* — UML: black-outlined **three-compartment** box, monospace name, black orthogonal connector, **unfilled triangle** arrowhead: **L02 p36, L02 p40**.
- *H3* — black-outlined box, **serif** name, **teal diagonal** (not orthogonal) lines, plain arrowhead, and every edge **labelled "Is a" / "Is an"** in serif: **L03 p27**. All three put the superclass above. They disagree on arrowhead shape, on routing, and on whether edges are labelled. **Decision needed.**

2. **Hierarchy box border colour.** Purple `#4E247F` at **L04 p7** versus black `#171717` at **L08 p7 and L08 p11** — same figure type, four weeks apart.

3. **Class name typeface inside a box.** Serif at **L04 p7, L08 p7, L08 p11**; monospace at **L02 p36, L07 p27, L07 p29**.

4. **Where the class name goes.** Inside a dedicated name compartment at **L02 p36**; floating **above** an unlabelled two-compartment box as "Person Class" at **L02 p12 and L02 p13** — within one deck.

5. **Specification tag spelling — three variants.**

   - `@requires` / `@ensures` — **L04 p23** (the tag table itself), **L13 p36** (which additionally misspells it `@requiers`).
   - `@require` / `@ensure` (no `s`) — **L07 p18, p19, p20**, used consistently across that whole deck's worked example.
   - `@returns` instead of `@return` — **L05 p8**. **Decision needed.** The tag table (L04 p23) and the exam-prep deck (L13 p36) agree on the `-s` forms; the SOLID deck (L07) does not.

6. **Spacing inside UML member lines**, within one figure (**L02 p36**): `- name : String` (spaces both sides of the colon) in the `Employee` box, but `-hourlyRate: double` (no leading space, none before the colon) in `ContractEmployee`. Likewise `+getName():String` versus `+logHours(hours:int) : void` versus `+printDetails() : String`.

7. **Code-highlight rectangle colour.** Red at **L02 p38, L03 p28, L04 p14, L08 p16**; **purple** at **L02 p16 and L02 p17**; plain black at **L08 p14**.

8. **The "translates to / produces" block arrow.** Solid **teal** block arrow pointing **right** at **L02 p13** (UML box → code); hollow black-outlined block arrow pointing **down** at **L05 p10** (class → test class); solid **black** block arrow pointing **right** at **L06 p17** (monolith → modular). Same semantic role, three renderings.

9. **Aside / rule boxes.** Thin **black** rectangle at L02 p31, L07 p24, L08 p5, L09 p26; thin **teal** rectangle at **L03 p21**; **no box at all** at L03 p23, L06 p31, L07 p14, L08 p12, L08 p15 — all carrying the same kind of one-line rule-to-remember.

10. **Two stack renderings inside one deck. L01b p39**: purple open-topped container, purple filled blocks, grey push/pop arrows, monospace operation captions. **L01b p40 &**

p41: white cells with black borders and a heavy bottom rule, quoted string values in bold sans, variable name in monospace *below* the stack, no container and no arrows.

11. **Two array renderings. L01b p31–p37**: large cells, no name label, sans/mono digits. **L04 p27**: small cells, serif digits, array name `ar` in monospace to the left of the row.
12. **Code-block borders.** Crisp thin black rectangle (L02 p39, L04 p13); **hand-drawn wobbly "sketched" rectangle** at **L05 p10 and L05 p11** only; or no border at all (L02 p32, L04 p22).
13. **Two whole slide templates.** Template A (Tw Cen MT / Bell MT / crest / badge) versus Template B in **L11** (Raleway / Source Sans Pro / no chrome). A figure drawn for L11's week will not match a figure drawn for any other week.
14. **Attribution placement.** Top-right of the slide (L04 p8, L08 p4, L09 p28) versus adjacent to the figure itself (L08 p3, L06 p28) versus a cover image standing in for the citation (L03 p15).
15. **Which edition of *Effective Java*.** 2nd edition cover shown at **L03 p15**; "(2017)" i.e. 3rd at **L04 p8** and **L08 p20**; "(3rd Edition)" spelled out at **L04 p47**; unlabelled at **L03 p18, p22** and **L09 p28**.
16. **Syntax-colour theme drifts between pasted screenshots.** `new` renders blue `#0033B3` at **L02 p16** but magenta at **L01b p42**; some blocks carry IntelliJ inlay hints (L02 p5, p6) and others do not.
17. **Sub-title-line capitalisation.** "Levels" at **L06 p21** but "levels" at **L06 p29**; "Byte Stream (Output)" (L08 p7) beside "Abstraction" (L08 p5). Slide titles likewise: "Black Box testing" (L05 p8) versus "Blackbox testing" (L13 p40); "Bounded generics" (L09 p21) versus "Bounded Generics" (L09 p20).
18. **The interface naming scheme.** `AI/BI/CI` at **L07 p27** but `BSpec` at **L07 p29 and p30** — two slides apart.
19. **Table styling.** The only formal data table I found (**L01a p18**, the semester schedule) uses a **navy** header band and pale-blue zebra rows — not the UQ purple used by everything else on the same slide deck.

Not established by the source

Things a new CSSE2002 diagram will plausibly need that the materials simply do not settle. **This is where the risk of inventing notation lives.**

- **Aggregation and composition.** No filled or hollow diamond appears in any page I read. L02 p9 has a code comment reading `// Association with AccountHolder`, but no diagram of that association is ever drawn. How to show "has-a" is unestablished.
- **Multiplicities.** No `1`, `0..1`, `1..*` or `*` ever appears on an edge.
- **Abstract classes and interfaces in a UML box.** No `«interface»` stereotype, no italicised abstract class name, no dashed realisation triangle. L07 writes the keyword into the label text instead (`interface BSpec`), but that deck's boxes have no compartments, so the two idioms cannot simply be merged.
- **Static members.** The usual UML underline is never shown.
- **Package or namespace boxes.** Never drawn, despite packages appearing in code (L02 p20).
- **Sequence diagrams, state machines, activity diagrams, flowcharts, ER diagrams.** None in any of the 210 pages I read.
- **Control-flow graphs.** Glass-box testing is taught (L05 p6) and path/branch coverage is implied, but no CFG, decision tree or coverage diagram is drawn. There is no notation for one.
- **Stack-and-heap memory diagrams.** The schedule (**L01a p18**) lists a Week 3 applied class called "Stack & Heap", but no lecture deck I sampled draws one. Boxes-and-pointers, reference vs value, and how to render `null` are all unestablished from the lecture materials.
- **Object (instance-level) diagrams.** Nothing like `p1:Person`. The nearest thing is a borrowed clip-art figure at **L02 p15** in a completely foreign style (yellow label chips, grey value cards, child cartoons) that should not be copied.
- **Generic types inside a diagram box.** `<T>`, `? extends T` and `? super T` appear only in code and in a borrowed `Number`-tree graphic (L09 p27, p31). How to render a parameterised type in a class box is unestablished.
- **Exact palette values.** No deck states a hex code. Every colour in this note was sampled from rendered pixels. Line weights, arrowhead sizes and box corner radii are likewise nowhere specified.
- **Dark mode / greyscale printing.** Every figure assumes a white ground and colour output. There is no stated fallback, and the coupling/cohesion cube figures (L06 p20, p27) become unreadable in greyscale because colour is the only thing distinguishing modules.
- **Figure numbering and captions.** Not one figure in the decks is numbered or captioned "Figure N." Captions, where they exist, are bare serif phrases beneath the art. There is no established way to cross-reference a figure.
- **Whether new hierarchy figures should be UML-correct.** This is the single biggest open question. The dominant house style (H1) uses a plain arrowhead where UML requires a hollow triangle; lecture 02 uses the triangle. Neither is declared canonical.
- **Font substitutes.** Tw Cen MT, Bell MT and Inconsolata are the embedded faces, but nothing says what to substitute when they are unavailable.