UML Class Diagrams

Table of contents

UML (Unified Modelling Language) diagrams are the standard, language-agnostic way of visually describing the classes in a system and the relationships between them — useful for planning a design *before* writing code, and for communicating that design to others.

Class boxes

A class box has three parts:

```
ClassName
----------------------
attributes
----------------------
methods()
```

Each attribute/method is prefixed with a **visibility** marker:

Symbol	Visibility
`-`	private
`+`	public
`#`	protected

Relationship types

Relationship	Line style	Meaning
Inheritance	solid line, hollow (open) triangle arrowhead pointing to the parent	"is-a" — subclass inherits from superclass
Association	plain solid line	one class uses/references another, without ownership
Aggregation	solid line, hollow (open) diamond at the "whole" end	weak "has-a" — the part *can* exist independently of the whole
Composition	solid line, filled (solid) diamond at the "whole" end	strong "has-a" — the part *cannot* exist independently of the whole

Multiplicity

Multiplicity annotations on association/aggregation/composition lines describe how many objects on each end participate in the relationship:

Notation	Meaning
`0..1`	zero to one
`n`	an exact number
`0..*`	zero to many
`1..*`	one to many
`m..n`	a specific range

Why use UML?

- Plan a class hierarchy/object graph before writing any code.
- Communicate a design to teammates without needing to read source code.
- Document the intended structure and relationships for future maintenance.
- Class diagrams are the most commonly used UML diagram type in object-oriented design, but UML also includes other diagram types (e.g. sequence diagrams, use-case diagrams) not covered here.