GRASP (General Responsibility Assignment Software Patterns)

A set of very practical guidelines that help answer one of the most common object-oriented design questions: "Which class should be responsible for this responsibility?"

Here are the 9 original GRASP patterns (most people actively use 5–7 of them):

	Pattern Name	Core Idea / When to use it	Very practical heuristic / slogan	Common opposite (bad) smell
1	Information Expert	Assign responsibility to the class that has the necessary information	"That class already knows that information — let it do it"	Anemic Domain Model + Service classes doing everything
2	Creator	Who should create instances of Class A?	Class B should create A if B contains, aggregates, records, closely uses, or has the initializing data for A	God factories everywhere, everything created in services
3	Controller	Who should handle a system event / use case?	First object beyond UI that takes responsibility for the operation (usually façade-like object)	Fat UI controllers, everything in God Service class
4	Low Coupling	Assign responsibilities so classes depend on as few other classes as possible	"Keep dependencies minimal and stable"	Classes knowing about 10+ other concrete classes
5	High Cohesion	Keep related responsibilities together in one class	"Stuff that changes together, belongs together"	Classes with 2–3 completely unrelated methods
6	Polymorphism	When behavior varies by type — use polymorphism instead of conditionals	"Let the subclass decide how to behave"	Big switch/if-else on type checking
7	Protected Variations (PV)	Identify points of predicted variation and protect them	"Encapsulate what varies" — very close to OCP	Code full of if (type == "X") then...
8	Indirection	Assign responsibility to an intermediate object to avoid direct coupling	"Don't talk to strangers" → introduce a middleman	Direct dependency between two parts that change often
9	Pure Fabrication	Create artificial classes that aren't in the domain model when needed	"It's okay to invent classes that don't represent real-world concepts"	Forcing domain objects to do technical infrastructure

Larman, C. (2005). *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development*. Prentice Hall PTR.