SOLID Principles Cheatsheet

for real-world C# codebases & code reviews

S: Single Responsibility Principle (SRP)

- **Real Meaning:** One reason to change, not one "thing" it does.
- **Why It Matters:** Avoids "God classes" that block clean PRs & slow refactoring.
- **Personal Analogy:** "If you can't give a clean commit message for the change, it's violating SRP."
- **Code Smell:** Method/class summary has multiple `and` / `or` .
- **Actionable:** Before adding a method, ask: *"Is this a different concern?"*
- Read more on SRP
- **Short link:** *bytecrafted.dev/solid-srp*

O: Open/Closed Principle (OCP)

- **Real Meaning:** Add features by extension, not by editing old code.
- **Why It Matters:** Keeps legacy code stable; new business rules plug in cleanly.
- **Personal Analogy:** "If a new requirement means touching brittle switch statements, you're not OCP."
- **Code Smell:** Growing `switch` / `if` chains for types or behaviors.
- **Actionable:** When adding a rule, prefer new handler/class over changing the old one.
- Read more on OCP
- **Short link:** *bytecrafted.dev/solid-ocp*

L: Liskov Substitution Principle (LSP)

- **Real Meaning:** Subtypes must behave as expected, no surprises for callers.
- **Why It Matters:** Swapping implementations shouldn't break existing tests or runtime logic.
- **Personal Analogy:** "If a subclass throws where the base returns null, that's an LSP landmine."
- **Code Smell:** Derived classes override with different exceptions, parameters, or semantics.
- **Actionable:** Run parent class tests on every subclass; look for broken guarantees.
- Read more on LSP
- **Short link:** *bytecrafted.dev/solid-lsp*

I: Interface Segregation Principle (ISP)

- **Real Meaning:** Small, client-focused interfaces, never force unused methods.
- **Why It Matters:** Reduces coupling, makes mocks/tests trivial, avoids NotSupportedException landmines.
- **Personal Analogy:** "If your interface summary needs bullet points, it's already too fat."
- **Code Smell:** Implementations with empty or `throw NotSupportedException` methods.
- **Actionable:** Extract groups of related methods into separate interfaces as soon as a client skips one.
- Read more on ISP
- **Short link:** *bytecrafted.dev/solid-isp*

D: Dependency Inversion Principle (DIP)

- **Real Meaning:** Depend on abstractions, not concrete implementations, flip the usual control.
- **Why It Matters:** Makes business logic testable, swappable, and free of infrastructure glue.
- **Personal Analogy:** "If you see `new SqlRepo()` in a service, that's DIP going up in flames."
- **Code Smell:** Direct instantiation of dependencies inside business logic.
- **Actionable:** Use constructor injection for every external dependency; mock in tests, swap in production.
- Read more on DIP
- **Short link:** *bytecrafted.dev/solid-dip*

Read full series: *bytecrafted.dev/series/solid*.

Further Reading

- Violating SOLID Principles
- How does composition support SOLID?

Made by bytecrafted.dev - Real-world software design for working developers.