Design Principles

no principle is a law, and all principles should be used when and where they are helpful.

SINGLE RESPONSIBILITY PRINCIPLE

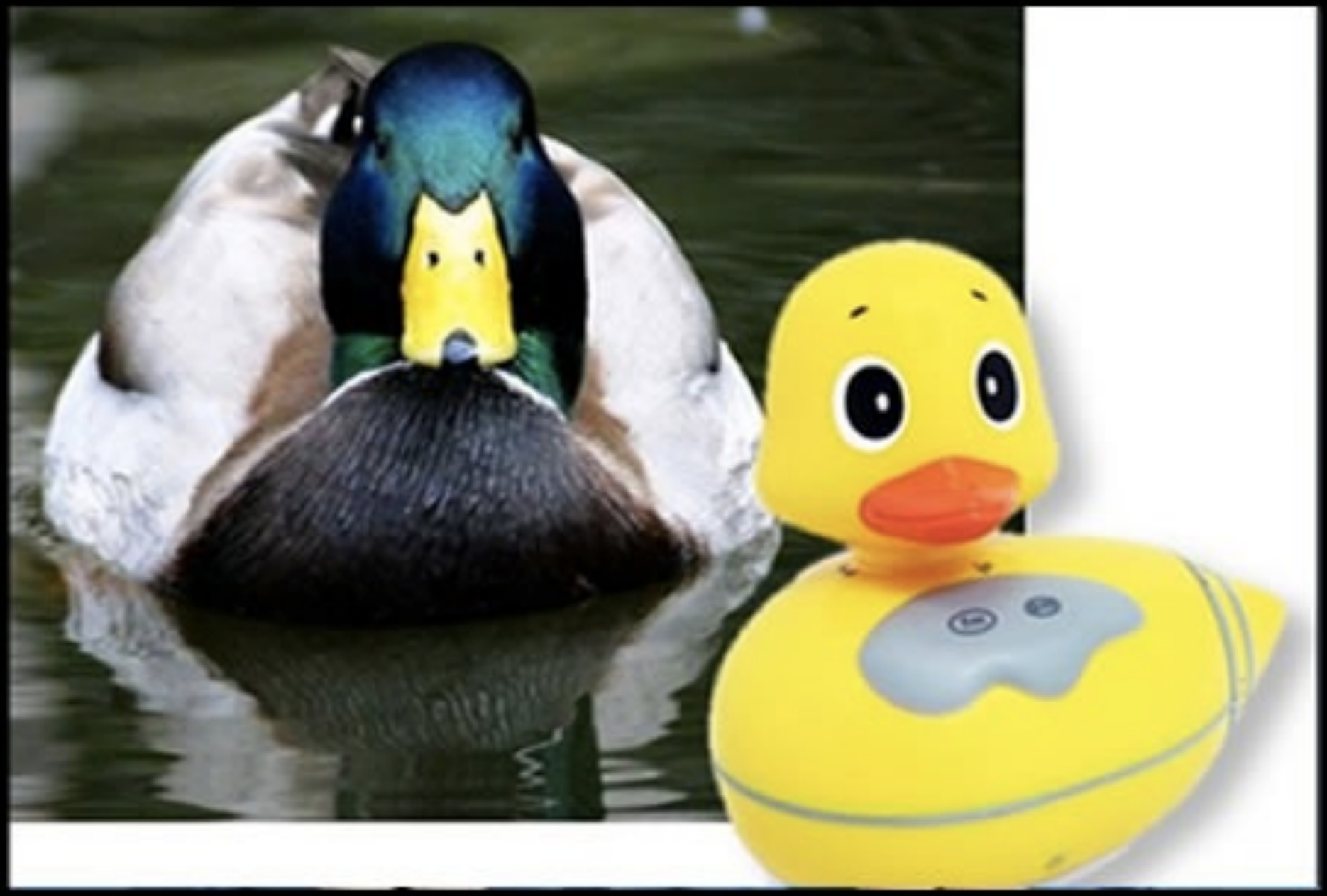
Every object should have a single responsibility, and all its services should be narrowly aligned with that responsibility.

If the application is not changing in ways that cause the two responsibilities to change at different times, there is no need to separate them.

Open-Closed Principle

Open-chest surgery isn't needed when putting on a coat.

Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification.



Liskov Substitution Principle

If it looks like a duck and quacks like a duck but needs batteries, you probably have the wrong abstraction.

In class hierarchies, it should be possible to treat a specialized object as if it were a base class object.

Interface Segregation

Tailor interfaces to individual clients' needs.

Clients should not be forced to depend upon interfaces that they don't use.

Dependency Inversion Principle

Would you solder a lamp directly
to the electrical wiring in a wall?

Abstractions should not depend on details.
Details should depend on abstractions

DON'T REPEAT YOURSELF

Repetition is the root of all software evil

Don't Repeat Yourself (DRY) / Duplication is Evil (DIE) / Single Source of Truth is a principle aimed at reducing repetition of information of all kinds.

You Aren't Gonna Need It

Don't waste resources on what you *might* need.

Solve tomorrows problem tomorrow,
Good for now Design

Boy Scout Rule

Leave your code better than you found it.

If you find a mess on the ground, you clean it up regardless of who might have made the mess. You intentionally improve the environment for the next group of campers.

Don't Call Us, We'll Call You

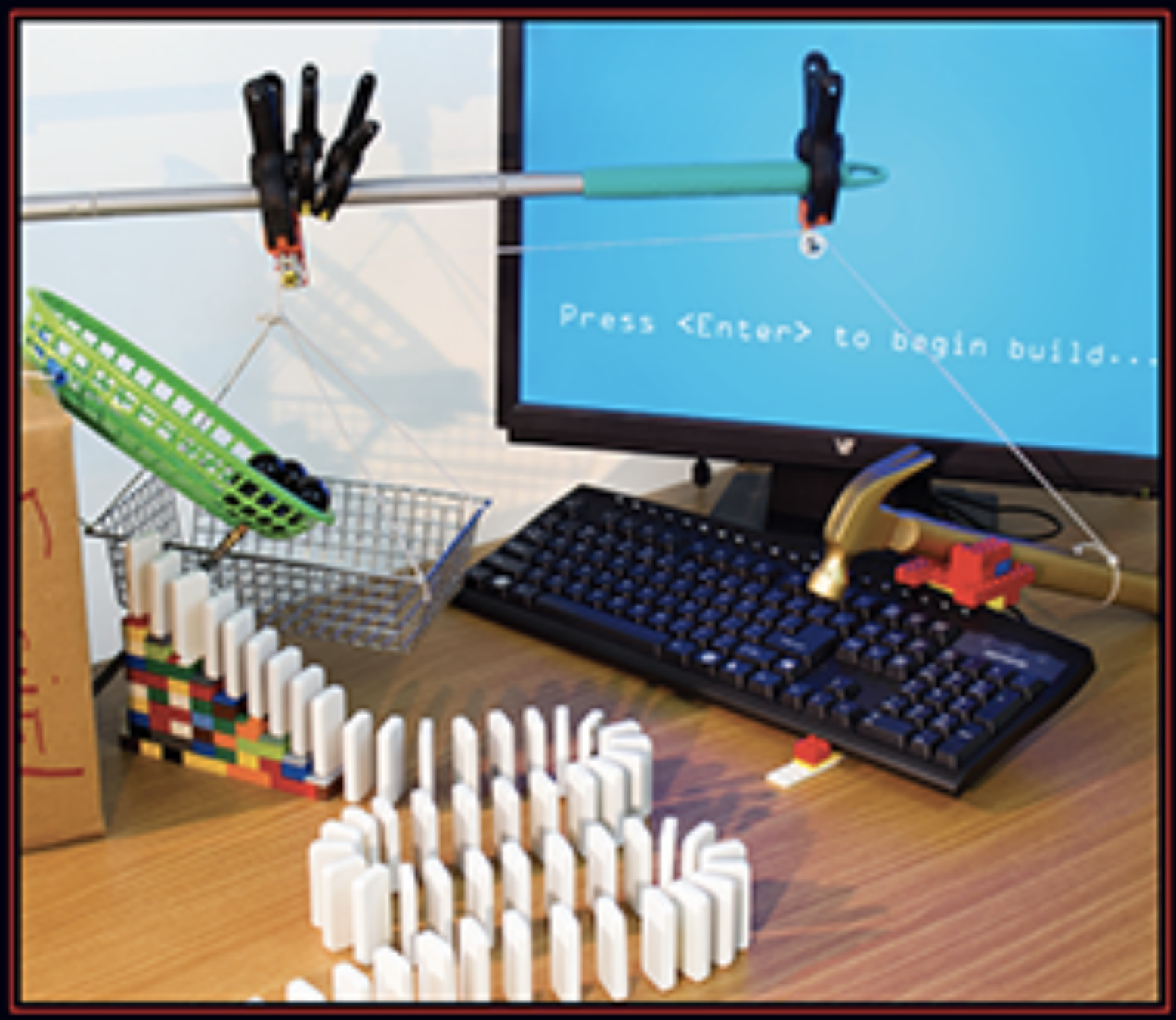
Keep your options open.

Inversion of control / Hollywood Principle inverts control(flow) as compared to traditional programming

Separation of Concerns

Don't let your plumbing code pollute your software.

avoid pollution of the domain logic with infrastructure stuff



Keep It Simple

If only there were an easier way.

Simplicity is the ultimate sophistication

Leonardo da Vinci

Favour Composition over Inheritance

Inheritance is one of the most abused concepts

Command Query Seperation Principle

Asking a Question should not change the answer

every method should either be a command that performs an action, or a query that returns data to the caller, but not both.

Design by Contract

each service provided must specify a clear contract

“If you promise to call S with the precondition satisfied, then I, in return, promise to deliver a final state in which the post-condition is satisfied.”

principle of least astonishment

the name of a function should reflect what it does

result of performing some operation should be obvious, consistent, and predictable, based upon the name of the operation

Law of Demeter

Don't talk to strangers

“Tell Don’t Ask” tell an object to do something rather than rip data out of an object to do it in client code.