# The Single Responsibility Principle (Gruppenarbeit)

The Single Responsibility Principle (SRP) states that a class or module should have one, and only one, reason to change. This principle gives us both a definition of responsibility, and a guidelines for class size. Classes should have one responsibility—one reason to change.

SRP is one of the more important concept in OO design. It's also one of the simpler concepts to understand and adhere to. Yet oddly, SRP is often the most abused class design principle. We regularly encounter classes that do far too many things. Why?

Getting software to work and making software clean are two very different activities. Most of us have limited room in our heads, so we focus on getting our code to work more than organization and cleanliness. This is wholly appropriate. Maintaining a separation of concerns is just as important in our programming activities as it is in our programs. The problem is that too many of us think that we are done once the program works. We fail to switch to the other concern of organization and cleanliness. We move on to the next problem rather than going back and breaking the overstuffed classes into decoupled units with single responsibilities. At the same time, many developers fear that a large number of small, single-purpose classes makes it more difficult to understand the bigger picture. They are concerned that they must navigate from class to class in order to figure out how a larger piece of work gets accomplished.

However, a system with many small classes has no more moving parts than a system with a few large classes. There is just as much to learn in the system with a few large classes. So the question is: Do you want your tools organized into toolboxes with many small drawers each containing well-defined and well-labeled components? Or do you want a few drawers that you just toss everything into? Every sizable system will contain a large amount of logic and complexity. The primary goal in managing such complexity is to organize it so that a developer knows where to look to find things and need only understand the directly affected complexity at any given time. In contrast, a system with larger, multipurpose classes always hampers us by insisting we wade through lots of things we don't need to know right now. To restate the former points for emphasis: We want our systems to be composed of many small classes, not a few large ones. Each small class encapsulates a single responsibility, has a single reason to change, and collaborates with a few others to achieve the desired system behaviors.

*C., Martin Robert. Clean Code: A Handbook of Agile Software Craftsmanship, Kindle Edition.*

## Aufgaben zum Text

**Zusammenfassen:** Erklären Sie in eigenen Worten, was das Single Responsibility Principle bedeutet.

**Begründen:** Warum verstoßen viele Entwickler laut dem Text gegen dieses Prinzip, obwohl sie es eigentlich gut verstehen?

**Vergleichen:** Der Text enthält einen Vergleich mit Werkzeugkästen. Erläutern Sie, was dieser Vergleich aussagt.

**Bewerten:** Diskutieren Sie mit Ihren Nachbaren, ob Sie persönlich lieber mit vielen kleinen Klassen oder mit wenigen großen Klassen arbeiten würden - und warum.