## Motivation

### In this context, a responsibility is considered to be one reason to change. This principle states that if we have 2 reasons to change for a class, we have to split the functionality in two classes. Each class will handle only one responsibility and if in the future we need to make one change we are going to make it in the class which handles it. When we need to make a change in a class having more responsibilities the change might affect the other functionality related to the other responsibility of the class.

### **The Single Responsibility Principle** is a simple and intuitive principle, but in practice it is sometimes hard to get it right.

## Intent

### **Each software module or class or function method should have only one reason to change**

**How does this principle help us to build better software?** Let's see a few of its benefits:

1. **Testing** – A class with one responsibility will have far fewer test cases.
2. **Lower coupling** – Less functionality in a single class will have fewer dependencies.
3. **Organization** – Smaller, well-organized classes are easier to search than monolithic ones.

User Class

Own responsibility

Create User

Extra responsibility

Add Role

Add Org

Org Class

Role Class

User Class

Add Org

Add Role

Create User