Software Architecture

Zhenyan Ji
zhyji@bjtu.edu.cn

Content

- Symptoms of Poor Design (Design Smells)
- SRP – The Single-Responsibility Principle

Symptoms of Poor Design (Design Smells)

- Odors of rotting software:
  - Rigidity
  - Fragility
  - Immobility
  - Viscosity
  - Needless Complexity
  - Needless repetition
  - Opacity

Smells - Rigidity

- *The system is hard to change because every change forces many other changes to other parts of the system.*
- A design is rigid if a single change causes a cascade of subsequent changes in dependent modules.
- The more modules that must be changed, the more rigid the design.
- Root causes are **bad modularization** and **high coupling.**

Coupling

- **Coupling** describes how dependant one object is on another object (that it uses). Objects that are loosely coupled can be changed quite radically without impacting each other. The slightest change to objects that are tightly coupled can cause a host of problems.

Coupling

- Problems with high coupling:
  - Change in one module forces changes in other modules
  - Modules are difficult to understand in isolation
  - Modules are difficult to reuse or test because dependent modules must be included.
- The goal is **Low Coupling**

Example

```
class A {
int x;
...
}
class B extends A {
void b() {
x = 5;
}
}
```

Coupling

- The emphasis on **classes** and **inheritance** can result in some types of undesirable coupling.
- Erich Gamma (GoF): “Favor object composition over class inheritance. ”
- Erich Gamma (GoF): “Program to an interface, not an implementation”

Smells - Fragility

- *Changes cause the system to break in places that have no conceptual relationship to the part that was changed.*
- Is closely related to rigidity (same root causes)
- Managers (and now also developers) will fear change
- Fragility tends to get worse, and the software gets impossible to maintain

Smells - Immobility

- *It is hard to disentangle the system into components that can be reused in other systems.*
- May happen because modules are not designed for reuse, e.g. when modules depends on infrastructure or when modules are too specialized.
- related to **low cohesion**
- The consequence is that software is rewritten instead of reused.

Smells - Viscosity

- *Doing things right is harder than doing things wrong.*
- Two forms: **Viscosity of the design** or **viscosity of the environment**.
- If making changes that preserves the design is harder to do then doing “hacks”, the viscosity of the design is high.
- Viscosity of environment comes about when the development environment is slow and inefficient.

Smells – Needless Complexity

- *The design contains infrastructure that adds no direct benefit.*
- This frequently happen when developers anticipate changes to the requirements, and put in facilities for those potential changes.
- The design will carry the weight of all the unused design elements, and possibly make other changes difficult.

Smells – Needless repetition

- *The design contains repeating structures that could be unified under a single abstraction.*
- A result of cut and paste
- All duplication is bad!
- Be aware of semi-duplication, code that is almost the same. Is even worse to fix.
- Makes the software difficult to maintain

Smells - Opacity

- *The code is hard to read and understand. It does not express its intent well.*
- Not following a coding standard
- Bad or inconsistent naming
- Bad or lacking commenting
- Modules too big
- Some kind of code review should be done to avoid opaque code.

What Stimulates the Software to Rot?

- Requirements always change!
- Poor design!
- Short-term-thinking!

Design Principles

- Software design principles represent a set of guidelines that helps us to avoid having a bad design.
- the principles of OO design:
  - SRP - Single Responsibility Principle
  - OCP - Open-Closed Principle
  - LSP - Liskov Substitution Principle
  - ISP - Interface-Segregation Principle
  - DIP - Dependency-Inversion Principle

SRP - Single Responsibility Principle

- Principle: A class should have only one reason to change.
- Responsibility = ”a reason to change”
- If a class has more than one responsibility, then the responsibilities become coupled.
- The cohesion is low if the module does several things
- Cohesion should be high

Cohesion

- Cohesion: a measure of how well the lines of source code within a module work together to provide a specific piece of functionality. In object-oriented programming, the degree to which a method implements a single function. Methods that implement a single function are described as having high cohesion.

Cohesion

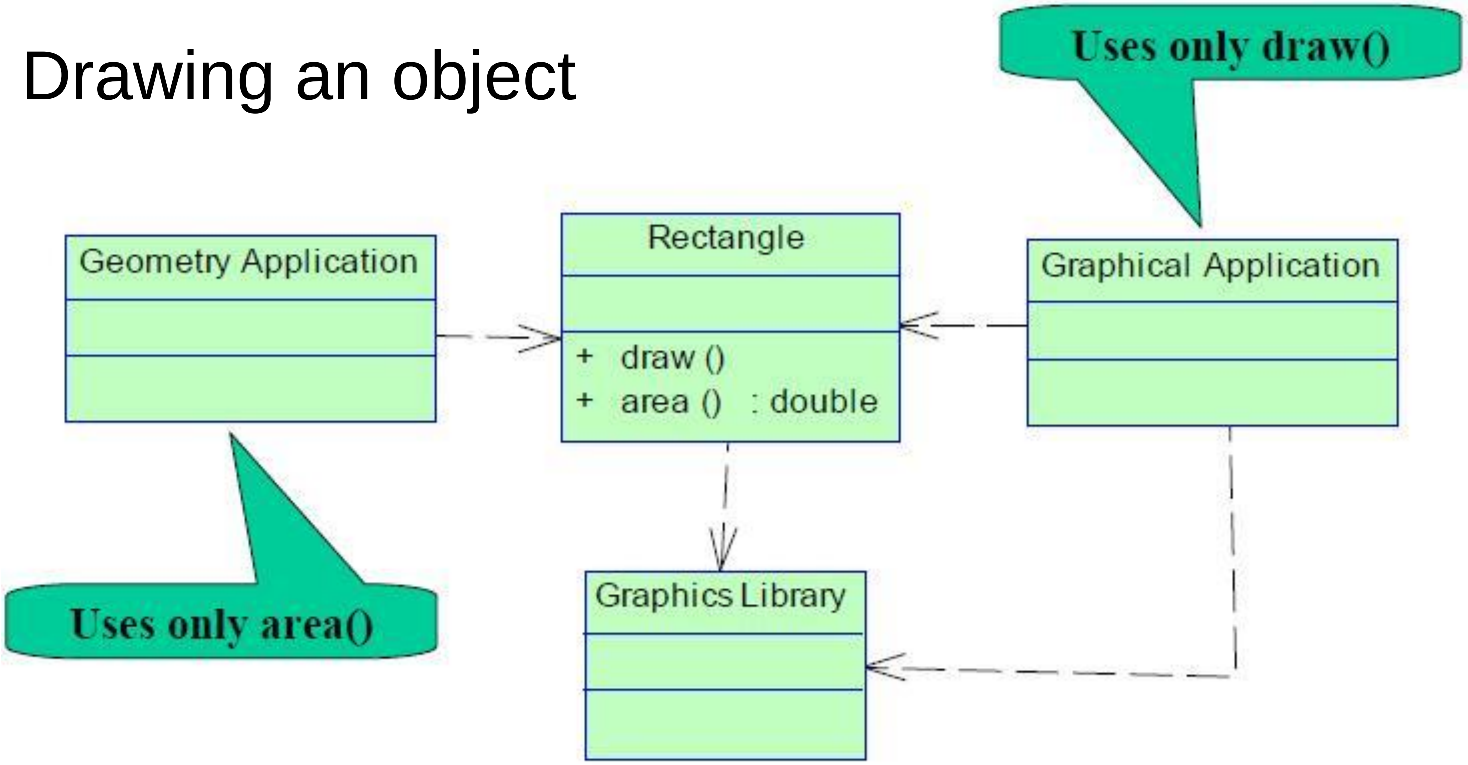
- Cohesion is decreased if:
  - The responsibilities (methods) of a class have little in common.
  - Methods carry out many varied activities, often using unrelated sets of data.

Cohesion

- Disadvantages of low cohesion (or "weak cohesion") are:
  - Increased difficulty in understanding modules.
  - Increased difficulty in maintaining a system.
  - Increased difficulty in reusing a module because most applications won’t need the random set of operations provided by a module.

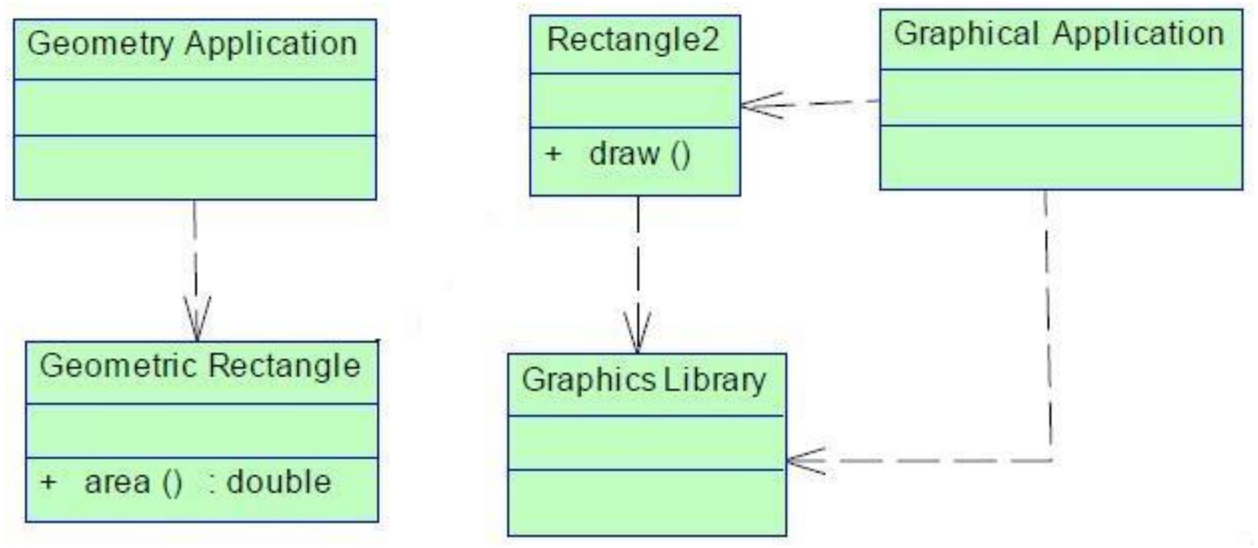
SRP Example I

- Rectangle – Class with two responsibilities
  - Geometrical calculations
  - Drawing an object



SRP Example I: better design

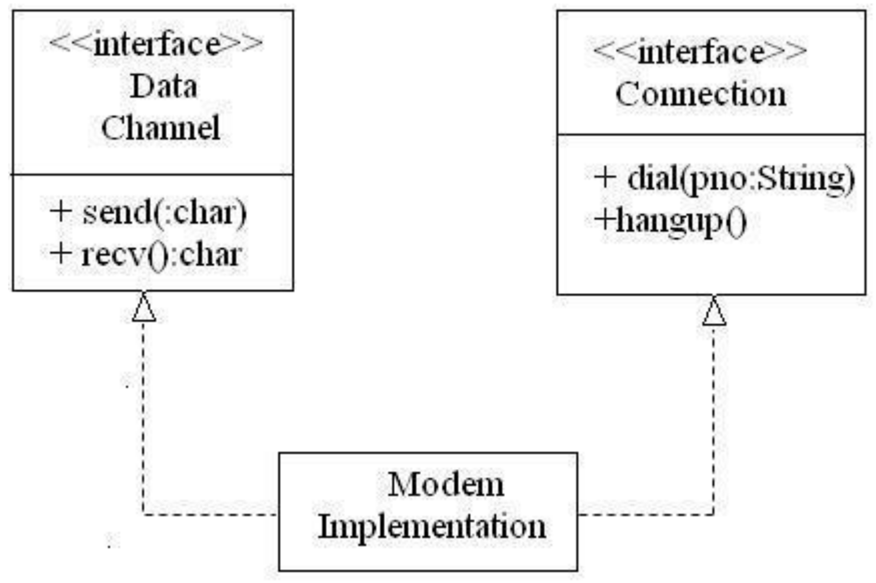
- Separate the responsibility into 2 classes
  - Geometric Rectangle – Geometrical calculations
  - Rectangle2 – Drawing an object



SRP Example II

```
interface Modem
{
   public void dial(String pno);
   public void handup();
   public void send(char c);
   public char recv();
}
```

SRP Example II



SRP - Summary

- But: Don’t separate responsibilities if it is unlikely to have independent changes. Otherwise, the codes will smell of “Needless Complexity”.

A broader perspective ...

- The principle of high cohesion can be applied at different levels. The examples we have seen so far focus primarily on class-cohesion.
- We can also talk about cohesion in methods, packages and subsystems.
- Just as an example of method-cohesion, see: http://www.javaworld.com/jw-05-1998/jw-05-techniques.html

Question

- Please give an example that violates SRP and explain why? How to modify it to conform to SRP?