6/3/2015

SOLID programming

Sebastian J. Wirkert

Willkommen im DKFZ!

dkfz.
DEUTSCHES
KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

50 Jahre – Forschen für
ein Leben ohne Krebs

SOLID

- Acronym for five object oriented design principles
- Invented by Robert C. Martin (Uncle Bob)

- Wikipedia:
- The principles, when applied together, intend to make it more likely that a programmer will create a system that is easy to maintain and extend over time. The principles of SOLID are guidelines that can be applied while working on software to remove code smells by causing the programmer to refactor the software's source code until it is both legible and extensible. It is part of an overall strategy of agile and adaptive programming.

S

- Single responsibility principle

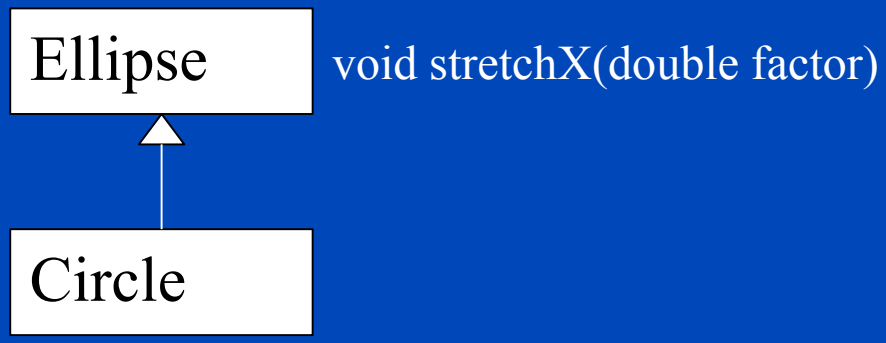
- a class should have only a single responsibility
- Responsibility: reason for change

- Bad example: a module that compiles and prints a report.

O

- Open/Close principle
- software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification
- Good example: interface classes (shall not be modified, but can be derived and extended).

- Liskov substiution principle
- Let Φ(x) be a property provable about objects x of type T. Then Φ(y) should be true for objects y of type S where S is a subtype of T.
- Bad Example: Circle-ellipse problem



- Interface-segregation principle

- The interface-segregation principle (ISP) states that no client should be forced to depend on methods it does not use. ISP splits interfaces which are very large into smaller and more specific ones so that clients will only have to know about the methods that are of interest to them.
- ISP is intended to keep a system decoupled and thus easier to refactor, change, and redeploy

D

- Dependency-Inversion-Principle

- A. High-level modules should not depend on low level modules. Both should depend on abstractions.
- B. Abstractions should not depend upon details. Details should depend upon abstractions.

```java
public class Lampe {
   private boolean leuchtet = false;

   public void anschalten() {
      leuchtet = true;
   }

   public void ausschalten() {
      leuchtet = false;
   }
}

public class Schalter {
   private Lampe lampe;
   private boolean gedrueckt;

   public Schalter(Lampe lampe) {
      this.lampe = lampe;
   }

   public void drueckeSchalter() {
      gedrueckt = !gedrueckt;
      if(gedrueckt) {
         lampe.anschalten();
      } else {
         lampe.ausschalten();
      }
   }
}
```

Schalter ⟶ Lampe

```java
public interface SchalterKlient {
   public void geheAn();
   public void geheAus();
}

public class Schalter {
   private SchalterKlient klient;
   private boolean gedrueckt;

   public Schalter(SchalterKlient klient) {
      this.klient= klient;
   }

   public void drueckeSchalter() {
      gedrueckt = !gedrueckt;
      if(gedrueckt) {
         klient.geheAn();
      } else {
         klient.geheAus();
      }
   }
}

public class LampeDIP implements SchalterKlient{
   private boolean leuchtet = false;

   public void geheAn() {
      leuchtet = true;
   }

   public void geheAus() {
      leuchtet = false;
   }
}
```

Schalter ⟶ SchalterKlient

△

LampeDIP

Auf Wiedersehen im DKFZ!

dkfz.
DEUTSCHES KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

50 Jahre – Forschen für ein Leben ohne Krebs