YASP

Yet Another SOLID Presentation

Thomas Rothenbacher - 04.03.2024

What are the SOLID principles

- Single Responsibility (SRP)
- Open/Closed (OCP)
- Liskov Substitution (LSP)
- Interface Segregation (ISP)
- Dependency Inversion (DIP)

Each class should have only one reason to change

Single Responsibility Principle

Single Responsibility Principle

SRP

- Examples
  - Bad: A class that logs, validates, and performs calculations
  - Good: Separate logging, validation and calculation in different classes

Single Responsibility Principle

SRP

- Benefits
  - High cohesion and self explanatory
  - Reduce code conflict
  - Introducing new features becomes easier
- Your class shouldn’t be doing only one thing. Everything it does should be very closely related.

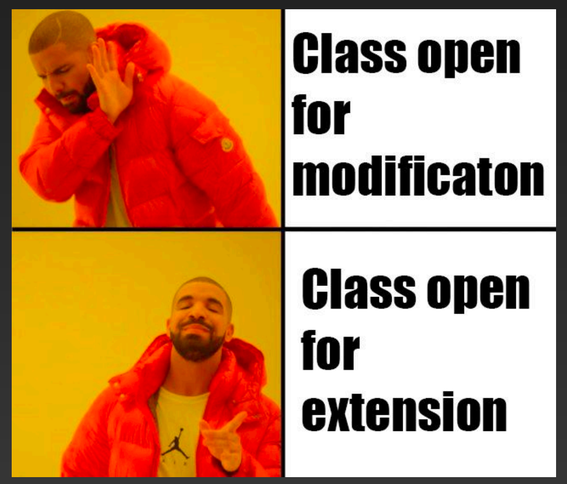
Software entities should be open for extension but closed for modification

Open/Closed Principle

Open/Closed Principle

OCP

- Examples
  - Bad: Directly modifying existing code instead of extending it.
  - Good: Use inheritance and interfaces



Objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program

Liskov Substitution Principle

Liskov Substitution Principle

LSP

No client should be forced to depend on methods it does not use

Interface Segregation Principle

Interface Segregation Principle

ISP

```
/**
 * Created by thomasr on 16.05.15.
 */
public class Simulation implements MouseMotionListener, MouseListener, KeyListener {
```

Interface Segregation Principle

ISP

```
111
112         @Override
113         public void mousePressed(final MouseEvent e) {
114             if (!go) {
115                 int mx = e.getX() / Cell.size;
116                 int my = e.getY() / Cell.size;
117                 cells[mx][my].setAlive(true);
118
119             }
120         }
121
122         @Override
123         public void mouseReleased(final MouseEvent e) {
124
125         }
126
127         @Override
128         public void mouseEntered(final MouseEvent e) {
129
130         }
131
132         @Override
133         public void mouseExited(final MouseEvent e) {
134
135         }
136
137         @Override
138         public void mouseDragged(final MouseEvent e) {
139
140         }
141
```

High-level classes should not depend on low-level modules. Instead, both should depend on abstraction

Dependency Inversion Principle

Dependency Inversion Principle

DIP

Conclusion

- SRP keeps classes focused and maintainable
- OCP allows for flexibility and extensibility
- LSP ensures interchangeable use of subclasses
- ISP prevents unnecessary dependencies
- DIP promotes decoupling for better reusability and testability

Thank you!

Any Questions?

Contact and Sources

- https://github.com/trothenbacher
- https://www.xing.com/profile/Thomas_Rothenbacher

- Sources:
  - https://blog.ndepend.com/defense-solid-principles/
  - https://www.coengoedegebure.com/solid-design-principles/
  - https://chat.openai.com/