Object oriented analysis and modeling
Principles of OO Design

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

Design

Complexity of most software systems is in the fact that in comprises of a large amount of simple tasks.

SOLID

Recommended:

1. Bob Martin: Bob Martin SOLID Principles of Object Oriented and Agile Design

SOLID

1. Single responsibility principle
2. Open-closed principle
3. Liskov substitution principle
4. Interface segregation principle
5. Dependency inversion principle

Single responsibility principle

A class should have only a single responsibility.

This implies:

- high cohesion
- almost necessarily LCOM4$=1$ for the class.

Open-closed principle

A class/package should be open for extension, but closed for modification.

Tools:

1. Inheritance
2. Composition (preference)
   - Strategy pattern (may create circular dependencies)

How to do composition and not create a dependency?

- Dependency injection (via constructor, via method)
- Factory method

Liskov substitution principle

Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

Example:

- Square as a subclass of Rectangle is probably not a good idea.

Interface segregation principle

Many client-specific interfaces are better than one general-purpose interface.

Implications:

- Limits the impact of an interface change.
- Connascence decreased.

Dependency inversion principle

One should depend upon abstractions, not concretions.

- In procedural programming a good practice was that higher level modules depend on lower level modules.
- This creates a chain of dependencies
- This principle asks us to depend on abstractions - typically interfaces.
- High-level modules should not depend on low-level modules. Both should depend on abstractions.
- Abstractions should not depend on details. Details should depend on abstractions.
- If followed completely, there should be an interface for each potential dependence between classes, but this is not too practical.

Further maxims

1. YAGNI
2. DRY
3. Rule of 3
4. Composition over inheritance
5. Objects are about behavior, not attributes
6. Strategy pattern - We may alway treat methods like attributes
7. Design for change, not to last

Main sources

Wikipedia - SOLID