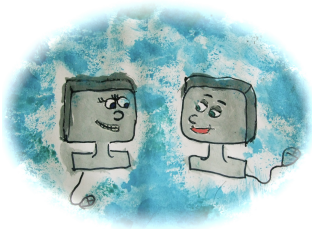
Design Patterns and Refactoring

Introduction

Oliver Haase

HTWG Konstanz



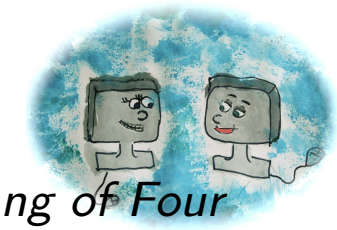
Literature

Design Patterns

- Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software* [1], Addison–Wesley, 1994, ISBN 78-0201633610.
- Elisabeth Freeman et al. *Head First Design Patterns*, O'Reilly, 2004, ISBN 978-0596007126.

Refactoring

- Martin Fowler et al. *Refactoring: Improving the Design of Existing Code*, Addison–Wesley, 1999, ISBN 978-0201485677.



[1] The bible for design patterns. The authors are widely knows as the *Gang of Four (GoF)*, their patterns as *GoF patterns*.

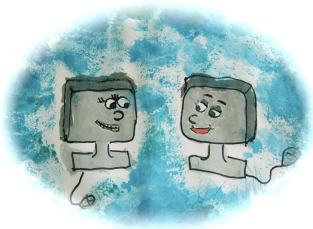
Design Patterns — Definition

Design Pattern:

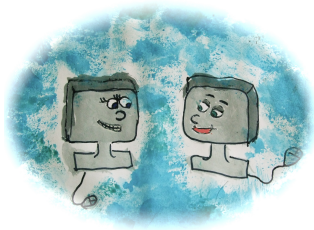
> *"Each [design] pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem" — Christopher Alexander, 1977.*

Christopher Alexander:

- born 1936 in Vienna
- studied mathematics and architecture (Cambridge)
- received PhD in architecture (Harvard)
- professor for architecture (UoC, Berkeley)
- known as the founder of design patterns



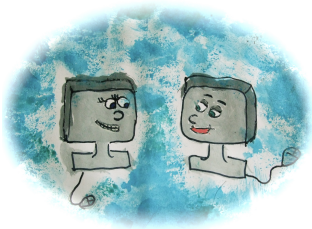
Design Patterns — Goals



Design Patterns — Goals

- fast to develop, clean solution through solution reuse
- simplified maintenance through clean structures and deja vu effects
- improved extensibility through anticipatory design
- (partly) opportunity for code generation or language support (e.g. singleton pattern)
- (partly) opportunity for class libraries (e.g. observer pattern)

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand." — Martin Fowler.



Design Patterns — Principles of Reusable Design

According to GoF, there are two fundamental principles of reusable software design (that also underly most if not all patterns):

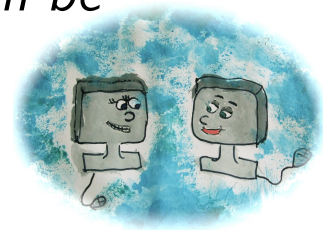
Principle of reusable software design

Program against an interface, not an implementation.

2nd principle of reusable software design

Prefer composition over inheritance.

"Computer Science is the discipline that believes all problems can be solved with one more layer of indirection." — Dennis DeBruler.

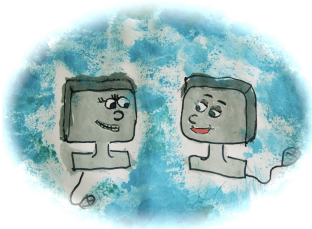


Design Patterns — Principles of Reusable Design

Key motivation behind both design principles:

Holy Grail of reusable software design

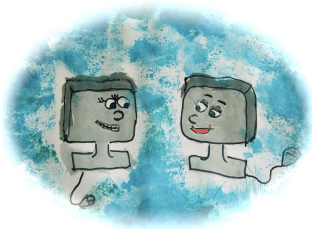
Avoid code dependencies to the largest extent possible.



Design Patterns — Categorization

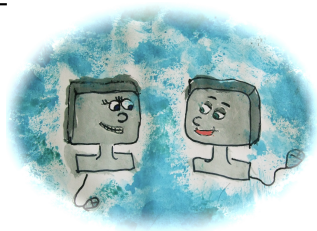
GoF categorize design patterns based on two criteria:

1. *Purpose*
   - **creational**: patterns for object creation
   - **structural**: patterns that compose classes and objects
   - **behavioral**: patterns that distribute responsibility for behavior across classes and objects, and make them interact
2. *Scope*
   - **class based**: based on relationships between classes, mainly inheritance
   - **object based**: based on relationships between objects



Design Patterns — Categorization

	creational	**structural**	**behavioral**
class based	factory method	adapter (class based)	interpreter template method
object based	abstract factory builder prototype singleton	adapter (object based) bridge decorator facade flyweight composite proxy	command observer visitor iterator memento strategy mediator state chain of responsibility



What's Next?

"The way to get started is to quit talking and begin doing" — Walt Disney.