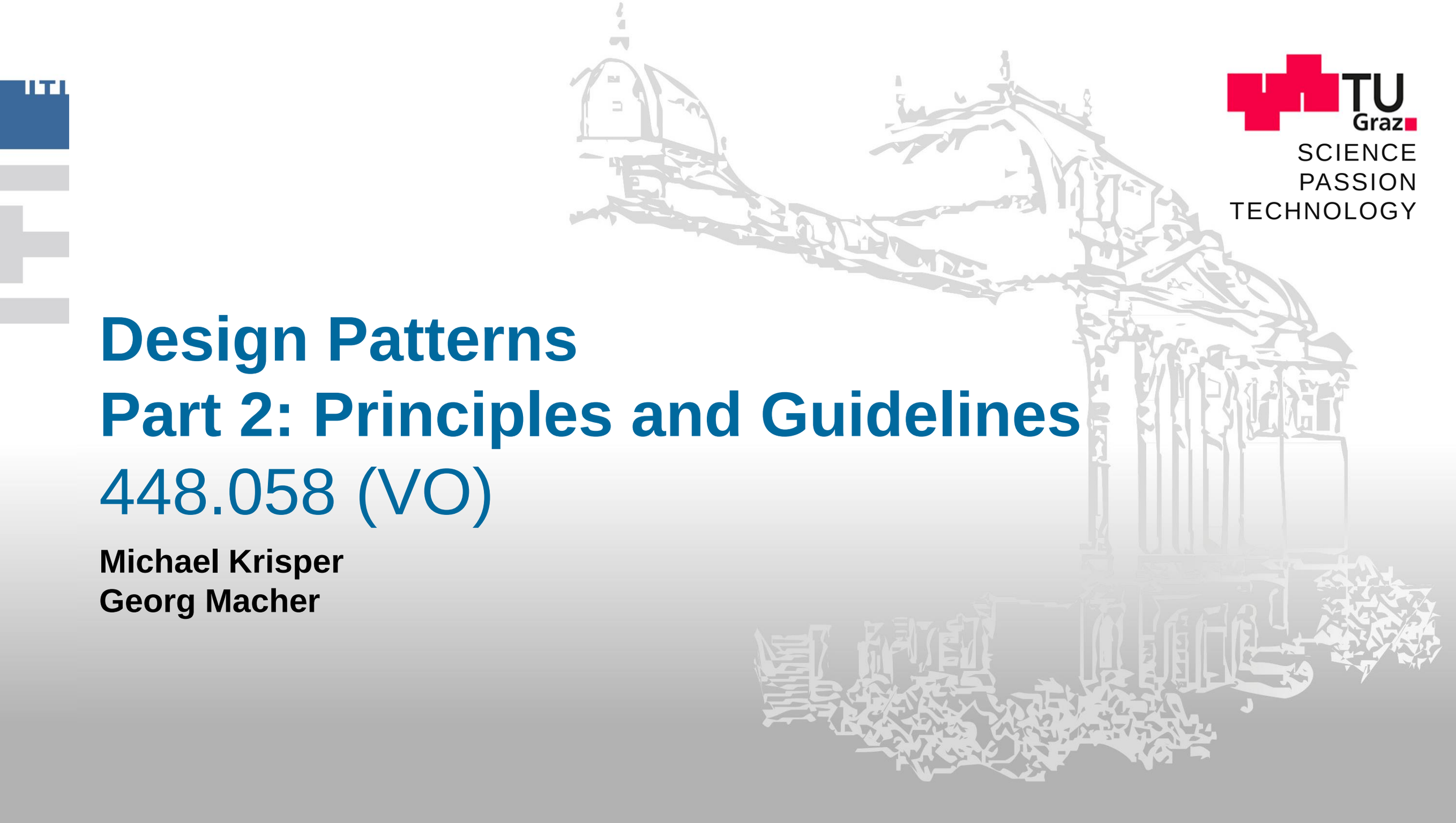


Design Patterns
Part 2: Principles and Guidelines
448.058 (VO)

Michael Krisper
Georg Macher

Principles and Guidelines

- SOLID
  - Single Responsibility
  - Open Closed Principle
  - Liskov Substitution
  - Interface Segregation
  - Dependency Inversion
- Principles of Good Programming / Clean Code
  - Decomposition
  - Abstraction
  - Decoupling
  - Usability & Simplicity
- Examples for some well known Patterns: Iterator, Observer, Layers

SOLID Principles (in OOP)

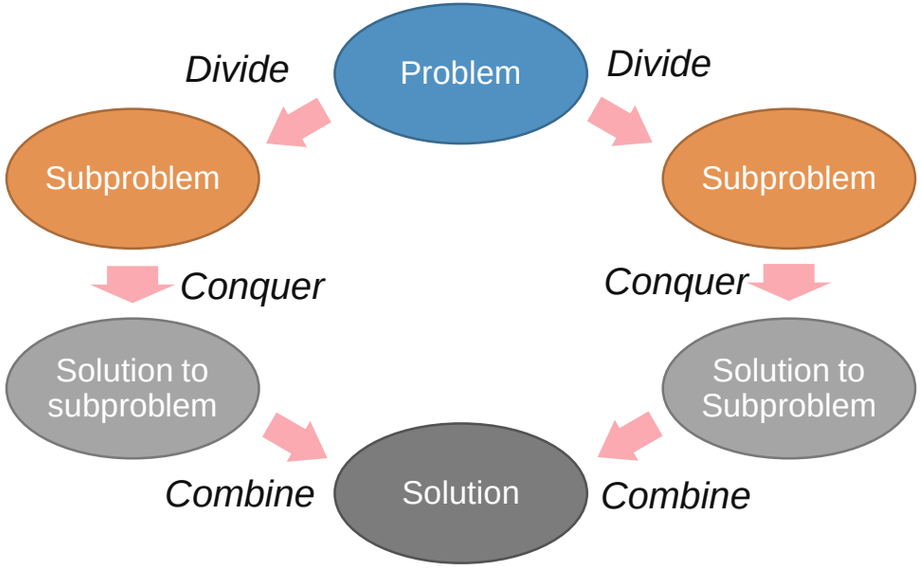
- ***S*ingle Responsibility**: A class should have one, and only one, reason to change.
- ***O*pen Closed**: You should be able to extend a class's behavior, without modifying it.
- ***L*iskov Substitution**: Derived classes must be substitutable for their base classes.
- ***I*nterface Segregation**: Make fine grained interfaces that are client specific.
- ***D*ependency Inversion**: Depend on abstractions, not on concrete implementations.

Principles of Good Programming

- **Decomposition**
  make a problem manageable
  decompose it into sub-problems
- **Abstraction**
  wrap around a problem
  abstract away the details
- **Decoupling**
  reduce dependencies, late binding
  shift binding time to “later”
- **Usability & Simplicity**
  make things easy to use right, hard to use wrong
  adhere to expectations, make usage intuitive

Decomposition

- Split up a problem until it gets manageable
- Divide and Conquer
- Separation of Concerns
- Orthogonality (Separation of Concepts)
- Single responsibility
- Curly's Law (do just one thing and stick to that)



[Movie: City Slickers (1991)]

Abstraction

- Hide implementation details
- Wrap another layer around a problem.
- Liskov substitution
  Substitute Parent-Classes by Sub-Classes
- Fundamental theorem of software engineering:
  "We can solve any problem by introducing an extra level of indirection.“
  (David Wheeler)

Decoupling

- Minimise coupling / Maximize cohesion
- Separation of Concerns
- Shift Binding time to “later”
- Composition over inheritance
- Inversion of control
  - Hollywood principle:
    “*Don’t call us, we call you!”*

- Open close - encapsulate what changes
- Embrace change
- Law of Demeter: Only use *“direct”* dependencies

Usability & Simplicity

- YAGNI – You ain’t gonna need it!
- DRY – Don’t repeat yourself!
- Principle of least astonishment
  - Don’t make me think
  - Easy to use right, hard to use wrong
  - Code for the Maintainer
  - Command / Query Separation
  - Interface segregation
- Ockham’s razor
  - Do the simplest thing possible
  - KISS – Keep it simple, stupid!
- Avoid premature optimization (Knuth, 1974)

Self-Assessment

1. What are the SOLID principles? Explain each in one sentence.

2. Which Principles govern good software solutions? Name two principles of good programming, and describe them in detail.

3. What do YAGNI, KISS and DRY mean?

4. Which design pattern could be used if you want to access every element of a collection in an uniform way?