

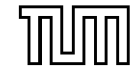
Script generated by TTT

Title: Simon: Programmiersprachen (07.12.2012)

Date: Fri Dec 07 11:05:39 CET 2012

Duration: 20:55 min

Pages: 9



Programming Languages

Multiple Inheritance

Dr. Axel Simon and Dr. Michael Petter
Winter term 2012

Outline



Inheritance Principles

- 1 Interface Inheritance
- 2 Implementation Inheritance
- 3 Liskov Substitution Principle and Shapes

C++ Object Heap Layout

- 1 Basics
- 2 Single-Inheritance
- 3 Virtual Methods

C++ Multiple Parents Heap Layout

- 1 Multiple-Inheritance
- 2 Virtual Methods
- 3 Common Parents

Discussion & Learning Outcomes

Outline



Inheritance Principles

- 1 Interface Inheritance
- 2 Implementation Inheritance
- 3 Liskov Substitution Principle and Shapes

C++ Object Heap Layout

- 1 Basics
- 2 Single-Inheritance
- 3 Virtual Methods

C++ Multiple Parents Heap Layout

- 1 Multiple-Inheritance
- 2 Virtual Methods
- 3 Common Parents

Discussion & Learning Outcomes

Excursion: Linearization

- 1 Ambiguous common parents
- 2 Principles of Linearization
- 3 Linearization algorithm

Interface vs. Implementation inheritance

The classic motivation for inheritance is implementation inheritance

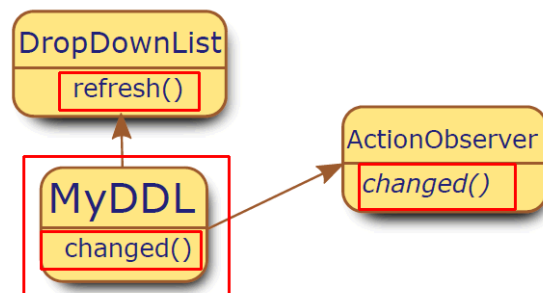
- *Code reuse*
- Child specializes parents, replacing particular methods with custom ones
- Parent acts as library of common behaviours
- Implemented in languages like C++ or Lisp

Code sharing in interface inheritance inverts this relation

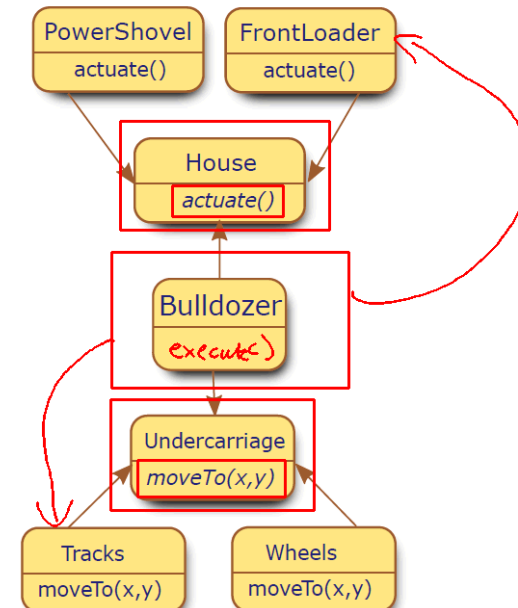
- **Behaviour contract**
- Child provides methods, with signatures predetermined by the parent
- Parent acts as generic code frame with room for customization
- Implemented in languages like **Java or C#**

“Wouldn’t it be nice to inherit from several parents?”

Interface Inheritance



Implementation inheritance





The Liskov Substitution Principle

Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.

```
class Rectangle {
    void setWidth (int w){ this.w=w; }
    void setHeight(int h){ this.h=h; }
    void getWidth ()    { return w; }
    void getHeight()    { return h; }
}
class Square extends Rectangle {
    void setWidth (int w){ this.w=w;h=w; }
    void setHeight(int h){ this.h=h;w=h; }
}
```

```
Rectangle r =
    new Square(2);
r.setWidth(3);
r.setHeight(4);
assert r.getHeight()*
       r.getWidth()==12;
```

