Wertsemantik in C++

Stephan Neuhaus

```
public class A { public int x; }

public class B {
  private void f(A a, int n) {
    a.x = 2;
    n = 3;
  }

  public static void main(String[] args) {
    A a = new A(); a.x = 1;
    int n; n = 2;
    f(a, n);
    // a.x = ?; n = ?
  }
}
```

Ziel

Wie erstelle ich eine Klasse in C++, die wie `int` oder `std::string` verwendet werden kann?

```
int main() {
  Person a;
  Address* x = new Address("Stuhlsatzenhausweg", "Saarbruecken");
  Person b = Person("Andreas", "Zeller", x);
  Person c = b;

  a = c = b;

  delete x;
  return 0;
}
```

Demo: Übergabe

Konzepte

- Call-by-value
- Call-by-reference (pointer)
- Pointer vs. Reference
- Address-of Operator
- Const Pointer/Reference

Demo: Pointer

Konzepte

- Aliasing
- Dereferencing

Demo: Kopieren

Konzepte

- Copy-Konstruktor
- Kopie vs. Referenz

Demo: Person

Konzepte

- Default Constructor
- Copy Constructor
- Assignment Operator
- Destructor
- Orthodox Canonical Class Form (OCCF)
- Ownership

Regeln

- Für Objekte mit Wertsemantik: OCCF
- Dynamische Ressourcen immer freigeben
- Eigentümerschaft (ownership) beachten