Bounds & Wildcards

Carsten Gips (HSBI)

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

Bounds: Einschränken der generischen Typen

```
public class Cps<E extends Number> {
    // Obere Schranke: E muss Number oder Subklasse sein
    // => Zugriff auf Methoden aus Number moeglich
}
Cps<Double> a;
Cps<Number> b;
Cps<String> c;  // Fehler!!!
```

- Schlüsselwort `extends` gilt hier auch für Interfaces
- Mehrere Interfaces: nach `extends` Klasse oder Interface, danach mit "`&`" getrennt die restlichen Interfaces:

```
class Cps<E extends KlasseOderInterface & I1 & I2 & I3> {}
```

Wildcards: Dieser Typ ist mir nicht so wichtig

Wildcard mit "`?`" => steht für unbestimmten Typ

```
public class Wuppie {
    public void m1(List<?> a) { ... }
    public void m2(List<? extends Number> b) { ... }
}
```

- `m1`: `List` beliebig parametrisierbar
  => In `m1` für Objekte in Liste `a` nur Methoden von `Object` nutzbar!
- `m2`: `List` muss mit `Number` oder Subklasse parametrisiert werden.
  => Dadurch für Objekte in Liste `b` alle Methoden von `Number` nutzbar ...

Bloch (2018): Nur für Parameter und nicht für Rückgabewerte nutzen!

Hands-On: Ausgabe für generische Listen

Ausgabe für Listen gesucht, die sowohl Elemente der Klasse `A` als auch Elemente der Klasse `B` enthalten

```
class A { void printInfo() { System.out.println("A"); } }
class B extends A { void printInfo() { System.out.println("B"); } }

public class X {
    public static void main(String[] args) {
        List<A> x = new ArrayList<A>();
        x.add(new A());  x.add(new B());
        printInfo(x);    // Klassenmethode in X, gesucht
        List<B> y = new ArrayList<B>();
        y.add(new B());  y.add(new B());
        printInfo(y);    // Klassenmethode in X, gesucht
    }
}
```

Konsole wildcards.v3.X

Wrap-Up

- Ein Wildcard (`?`) als Typ-Parameter steht für einen beliebigen Typ
  - Ist in Klasse oder Methode dann aber nicht mehr zugreifbar
- Mit Bounds kann man Typ-Parameter nach oben oder nach unten einschränken (im Sinne einer Vererbungshierarchie)
  - `extends`: Der Typ-Parameter muss eine Unterklasse eines bestimmten Typen sein
  - `super`: Der Typ-Parameter muss eine Oberklasse eines bestimmten Typen sein

LICENSE

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.