Überladung von Methoden (Overload)

openHPI-Java-Team

Hasso-Plattner-Institut

Überladung von Methoden

```
1  int sum(int x, int y) {
2     return x + y;
3  }
4
5  int sum(int x, int y, int z) {
6     return x + y + z;
7  }
8
9  double sum(double x, double y) {
10    return x + y;
11 }
```

Überladung von Methoden

- Gleicher Name
- Unterschiedliche Parameterliste
  - Anzahl der Parameter
  - Datentypen der Parameter

Überladung von Methoden

```
1  int sum(int x, int y) {
2     return x + y;
3  }
4
5  int sum(int c, int d) {
6     return c + d;
7  }
```

Überladung von Methoden

- Keine valide Überladung
- Parameterbezeichner sind nicht entscheidend

Überladung

Hasso Plattner Institut

```
1  int sum(int x, int y) {
2     return x + y;
3  }
4
5  double sum(int x, int y) {
6     return x + y;
7  }
```

Überladung von Methoden

- Keine valide Überladung
- Rückgabetypen nicht entscheidend

Überladung von Methoden

```java
int sum(int x, int y) {
   return x + y;
}

int sum(int x, int y, int z) {
   return x + y + z;
}

double sum(double x, double y) {
   return x + y;
}
```

Was wird aufgerufen?

sum(2, 4)	➔ `int sum(int x, int y)`
sum(2.0, 4.0)	➔ `double sum(double x, double y)`
sum(2.3, 4)	➔ `double sum(double x, double y)`
sum(2.0, 4)	➔ `double sum(double x, double y)`

Überladung von Methoden

```
1  int sum(int x, int y) {
2     return x + y;
3  }
4
5  int sum(int x, int y, int z) {
6     return x + y + z;
7  }
8
9  double sum(double x, double y) {
10    return x + y;
11 }
```

Überladung von Methoden

- Es wird die am Besten übereinstimmende Methode ausgeführt

Ein Beispiel aus der Java API

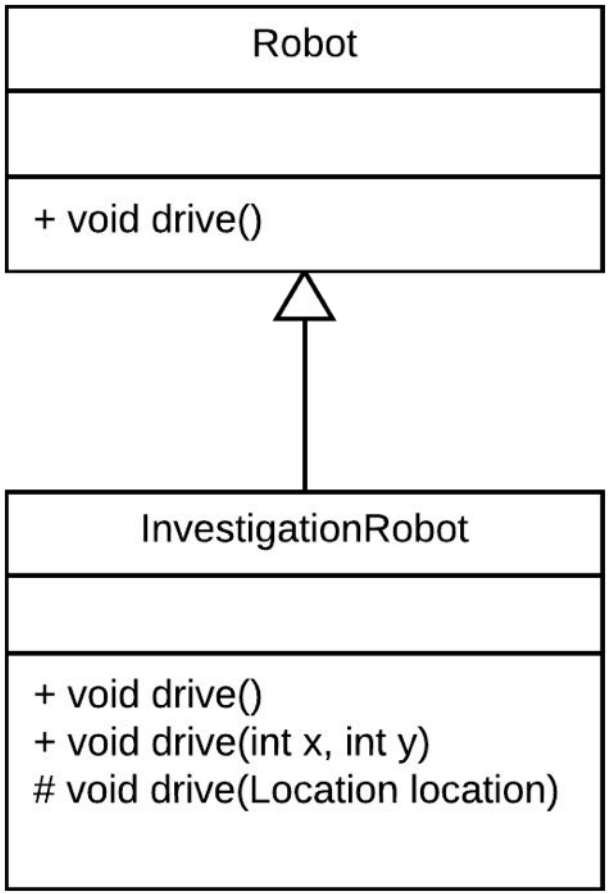
HPI Hasso Plattner Institut

```
1 public class PrintStream{
2    void print(int arg) { ...}
3    void print(String arg) { ...}
4    void print(char[] arg) { ...}
5    // ...
6 }
```

Vorteile:

- Methodenbezeichner wird identisch wiederverwendet
  - Nicht `printInt()`, `printString()` etc.
- Ausführung variiert abhängig von den übergebenen Argumenten

Overload

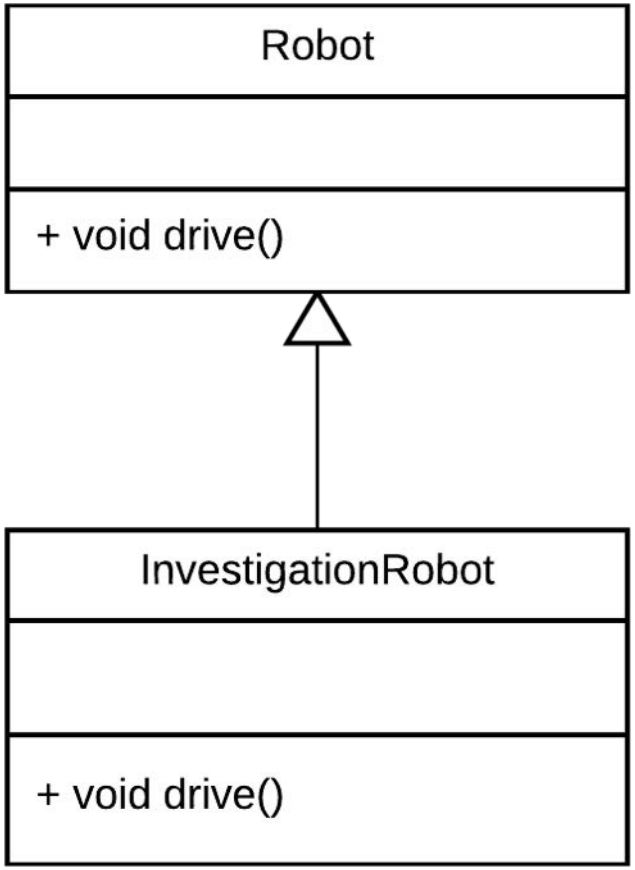


Überladen

- Verschiedene Methoden haben den gleichen Namen innerhalb einer Klasse
- Hat **nichts** mit Vererbung zu tun

- Parameterliste muss verändert werden
- Rückgabetypen können unterschiedlich sein
  - Ändern des Rückgabtyps reicht alleine nicht aus
- Sichtbarkeit darf in jede Richtung verändert werden

Override



Überschreiben

- Methode wird von der Superklasse an die Subklasse vererbt
- Subklasse ändert ihr Verhalten in Bezug auf diese Methode

- Parameter müssen gleich bleiben
- Rückgabetypen müssen kompatibel sein
- Sichtbarkeit darf nicht weiter eingeschränkt werden

Überladung von Methoden (Overload)

openHPI-Java-Team

Hasso-Plattner-Institut