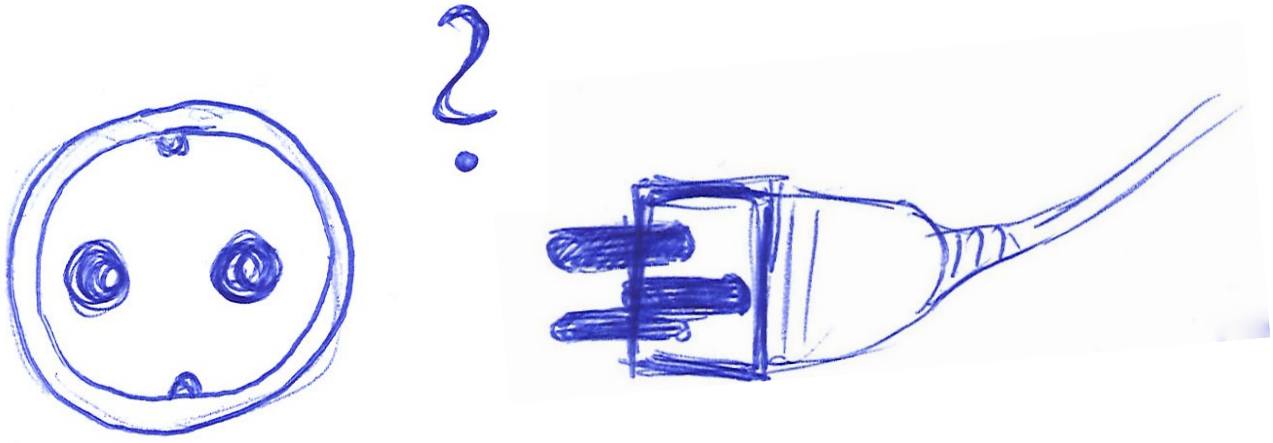
Type Casting

openHPI-Java-Team

Hasso-Plattner-Institut

Was tun - wenn mein Datentyp gerade nicht passt?

- Parameter und Rückgabewert haben vordefinierte Datentypen
- Manchmal liegt gerade ein anderer Typ vor als der den man braucht
- → temporäre Umwandlung (Type Casting)
- Funktioniert nur mit **kompatiblen** Typen



Implizites Type Casting

```
1 int x = 2;
2 double y = x;
3 // y speichert nun den Wert 2.0
4 // gleiches Ergebnis bei double y = 2;
```

- Nur möglich, wenn keine Information verloren geht

  → 2 wird zu 2.0
- Geschieht implizit ohne weitere Eingriffe durch den Entwickler

  → Keine spezielle Syntax nötig

Explizites Type Casting
Numerische Typen (1/2)

```
1 double y = 2.5;
2 int x = y;          <-- FEHLER
3 int x = (int) y;
4 // y wird abgeschnitten nach dem Komma
5 // x enthält nun also 2
```

- Compiler erzeugt Fehler bei **potentiellem Informationsverlust**
- Type Casting kann erzwungen werden
  - Zieltyp in runden Klammern vor den Bezeichner schreiben
  - Hallo Compiler! Ich übernehme hier mal selbst die Verantwortung!
- Funktioniert nur zwischen kompatiblen Datentypen

Explizites Type Casting klassischer Anwendungsfall (2/2)

```
1 double z = 1 / 2.0;
2
3 int x = 1;
4 int y = 2;
5
6 z = x / y;
7 z = x / (double) y;
```

z == 0.5

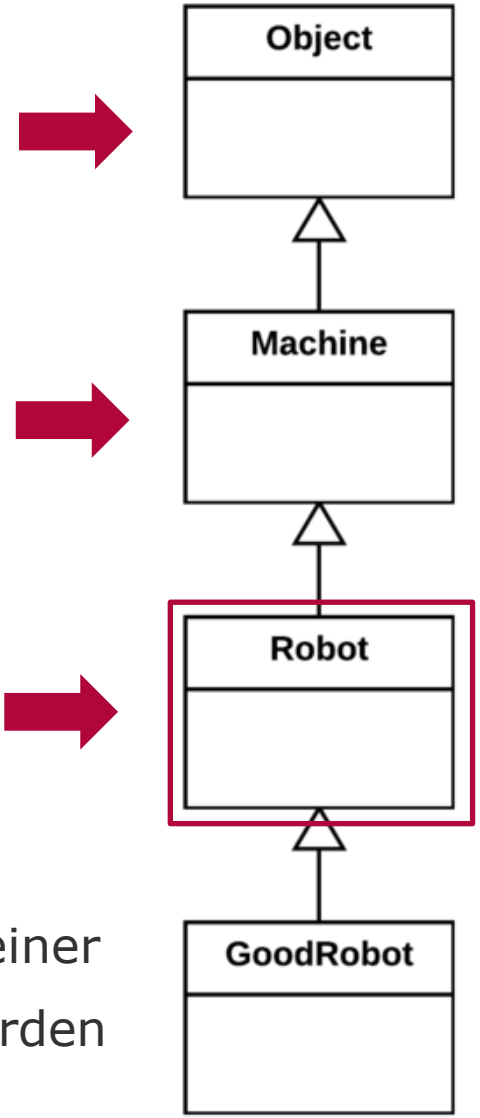
x/y == 0 ➔ z == 0.0

z == 0.5

- Division ergibt für Ganzzahlen keine Nachkommastellen
- → alles was zwischen 0 und 1 ist wird als 0.0 ausgegeben
- Erst wenn Zähler oder Nenner eine Kommazahl ist, wird Ergebnis mit Nachkommastellen gespeichert

Explizites Type Casting Objektdatentypen

```
1 Object robbi = new Robot();
2
3 Machine m = (Machine) robbi;
4
5 Robot richard = (Robot) m;
6
7 Robot randy = (Robot) robbi;
8
9 GoodRobot rolf = (GoodRobot) robbi;
10
```



- Jedes Objekt kann in jeden Datentyp innerhalb seiner (**linearen**) Vererbungshierachie umgewandelt werden

Explizites Typecasting
Objektdatentypen (2/2)

```
1 Robot aRobot = new Robot();
2
3
4 String s = (String) aRobot;
5 /* wirft ClassCastException, da für das Object
6 aRobot nur die Eigenschaften von Robot
7 vorhanden sind und nicht von String */
```

FEHLER

Parsing
String → Zahl

```java
double x = Double.parseDouble("1.234");
int y = Integer.parseInt("1234");
boolean b = Boolean.parseBoolean("true");
// nutzt bereits implementierte Methoden
```

- Statt Casting
- Bereitgestellte Methoden durch Java Wrapper-Klassen
- Wenn Zahl als String bekannt (z.B. "1.234")
- Funktioniert nicht für Strings, die nicht-numerisch sind
- → `int z = Integer.parseInt("Ich fliege");` wirft FEHLER
- → NumberFormatException