Typisierung

Effiziente Programmierung

Thomas Schnieders

Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2018-04-26

Gliederung (Agenda)

1 Einleitung

2 Typsystem

3 Typisierung

4 Zusammenfassung

5 Literatur

Einleitung

- Was sind Typen
- Was bedeutet Typsicherheit
- Typsicherheit durch Typumwandlung (implizit und explizit)
- Schwache und starke Typisierung
- Dynamische und statische Typisierung

Typsystem

- Der Typ definiert Wertebereich und zugehörige Operationen (für ein Objekt)
- Wertebereich in Java

```
1  Integer a = 3; // klappt, da 3 im Wertebereich
2  Integer b = 2147483648; // Fehler: out of Range
```

- Operationen

```
1  Integer a = 3;
2  int b = a.intValue(); // klappt, da Methode fuer
       ↪ Integer zulaessig
3  int c = a.length(); // Fehler: Methode kann auf a
       ↪ nicht angewendet werden
```

- Der Typ definiert Wertebereich und zugehörige Operationen (für ein Objekt)
- Ein Objekt ist ein Speicherbereich, in dem der Wert eines angegeben Typ abgelegt wurde
- Ein Wert ist eine Folge von Bits im Speicher, der entsprechend seines Typs interpretiert wird
- Eine Variable ist ein benanntes Objekt (Zuweisung bei der Deklaration) [Str10]

Programmiersprachen können zwischen unterschiedlichen Typgruppen unterschieden.
In Java zwei Arten von Typen:

- Primitivtypen (einfache, eingebaut):
  `byte`, `short`, `int`, `long`, `float`, `double`, `boolean`. `char`
- Referenztypen:
  Damit lassen sich Objektverweise auf Zeichenketten, Datenstrukturen realisieren. [Ull18]

In C++ Unterteilung in Grundtypen (`int`, `short`, `bool`, `char`, usw.) und die daraus ableitbaren oder zusammengesetzten Typen

Vorteil:

- Compiler unterscheidet zwischen primitiven Typen und Referenztypen
- Bytecode kann dadurch ebenfalls zwischen den Typen unterscheiden
- Laufzeitumgebung kann den Programmcode schneller ausführen

Typsicherheit

- Typsicherheit: Ausmaß, in dem eine Programmiersprache Typfehler verhindert
- Typfehler äußert sich durch unerwünschtes oder fehlerhaftes Programmverhalten
- Keine Typverletzungen: Datentypen werden gemäß ihren Definitionen verwendet
- Typsicherheit herzustellen ist je nach Sprache dann Aufgabe des Compilers (Statisch Typisiert) oder des Interpreters (Dynamisch Typisiert).
- Sprachunterstützung durch implizite Typumwandlung.

Implizite Typumwandlung

- Datentyp wird in einen anderen Datentypen umgewandelt
- Erscheint nicht im Quelltext
- `int` zu `long` in Java

```
1  int a = 3;
2  double b = a; // b hat jetzt den Wert 3.0
```

- arithmetische Typumwandlung Java

```
1  int a = 3;
2  double b = 4.7;
3  double c = a + b;  // b hat jetzt den Wert 7.7
```

- double zu int C++ (Achtung: Informationsverlust)

```
1  double a = 3.3;
2  int b = a; // b hat jetzt den Wert 3
```

- Typumwandlung im Funktionsaufruf in Java

```
1  public void func(long a){ ... } // irgendeine
       ↪ Funktion
2  int b = 1000;
3  // ...
4  func(b);     // der Aufruf
```

Explizite Typumwandlung

- Möglichkeit der expliziten Typumwandlung mittels Cast-Operator
- **Syntax**: `(typ)` Argument
- Unärer Operator mit hoher Präzedenz
- Der Wert des Ausdrucks wird in den angegebenen Typ konvertiert

Explizite Typzuweisung in Java

```
1  int i = 1;
2  short  s = (short) i;
```

```
1  int i = 1000000000;
2  short s = (short) i;
```

- In beiden Zuweisungen findet weder der Compiler noch die Laufzeitumgebung einen Fehler
- Umwandlung von `int` (4 byte) nach `short` (2 byte)[Gü17]
- Die höherwertigen 2 byte werden abgeschnitten
- Dadurch nimmt `s` im 2. Beispiel den Wert -13.824 (Interpretation im 2er-Komplement)
- Typerweiterung: `short` zu `int`
- Typeinschränkung: `int` zu `short`

- Typumwandlung ist auch mit Referenztypen möglich
- z.B. im Rahmen von Vererbung oder der Implementation von Interfaces
- Beispiel: explizite Umwandlung in Java

```
1  Object o1 = new String("test");
2  String s = (String) o1;
```

Umwandlung gelingt, da das von o1 referenzierte Objekt ein String ist [Str16]

Typisierung

Einleitung:

- Unterteilung:
  - Stark oder schwach typisiert
  - Dynamisch oder statisch typisiert
- Java ist stark und statisch typisiert
- C++ ist schwach und statisch typisiert
- Python ist stark und dynamisch typisiert

Starke und schwache Typisierung

Keine genaue Definition, aber verschiedenen Merkmale zur Stärke der Typisierung vorhanden:

- Nichtvorhandensein irgendwelcher Konvertierungsmethoden und fehlende Möglichkeit, das Typsystem zu umgehen
- Nur explizite Typumwandlungen möglich
- Typüberprüfung zur Übersetzungs- statt zur Laufzeit
- Implizite Typumwandlung nur zwischen ähnlichen Typen
- Generelle Unterscheidung zwischen Typen[Wik18]

Allgemein: stark typisierte Sprachen prüfen die Typisierung strenger zur Compilezeit

- Stark Typisiert sind z.B. Java oder Python
- Beispiel in Java

```
1  Object o1 = new String("test");
2  Object o2 = new Integer(1);
3  String s1 =  o1; // Type mismatch
4  String s2 = (String) o1;
5  s1 = (String) o2; // Laufzeitfehler
```

- Schwach typisiert sind z.B. C, C++, oder Assembler
- In C++ können z.B. Zeiger oder Zahlen implizit in boolsche Ausdrücke umgewandelt werden.
- Beispiel in C++

```
1  MeineKlasse* meinObjekt = meineKlasse();
2  String* text = new String("test");
3  text = (String*) meinObjekt;
```

Fehler wird nicht zur Laufzeit entdeckt, sondern erst wenn die Variable `text` im Programm aufgerufen wird.

Statische Typisierung

- Typ von Variablen und Parameter im Quelltext deklariert
- Variablen sind somit einem festen Typen zugeordnet
- Einschränkung in der Zuweisung von Objekten
- Überprüfung zur Compilezeit
- Statisch in Java

```
1  String foo = "pokipsi";
2  char[] charArray = foo.toCharArray();
3  foo = new int[3];  //Fehler: Type mismatch
```

Vorteile:

- Überprüfung des Wertebereichs und der zulässigen Operationen zur Compilezeit übernommen, sodass kompilierte Anwendung besser optimiert werden können
- Programmcode ist lesbarer, da klar ist, welche Aufgabe eine Variable hat
- Fehlererkennung durch den Compiler [Str16, S. 96]

Dynamische Typisierung

- Variablen sind im Quellcode keine deklarierenden Typen zugeordnet
- Variable kann Referenzen auf beliebige Objekte aufnehmen
- Prüfung, ob Funktionen auf konkrete Objekte zulässig sind: dynamisch zur Laufzeit

- Dynamisch: Variablenzuweisung in Python

```
1  a = 1        # a ist eine ganze Zahl
2  a = 2.0      # a ist jetzt eine Gleitkommazahl
3  a.upper()    # Scheitert: a ist keine Zeichenkette
4  a           # gibt den Wert von a aus
5  -> 2.0
6  a = "jetzt ist a ein String"
7  a += 1       # Scheitert: Inhalt von a ist String
8  a.upper()    # Gibt den neuen String aus
9  -> 'JETZT IST A EIN STRING'
```

- Funktionen in Python

```
1  def add(x,y):
2      return x + y
```

Vorteile:

- größere Flexibilität
- Variablen, Parameter und Ergebnisse von Funktionen benötigen keine deklarierenden Typen
- Funktionen können mit einer größeren Vielzahl von Objekten verwendet werden
- Notwendigkeit der explizite Typumwandlung entfällt

Nachteile:

- Je nach Zustand des Programms können Funktionen Objekte unterschiedlichen Types zurückgeben. $\rightarrow$ Probleme beim Programmieren
- In statisch typisierten Sprachen liefert eine Methode immer einen festen Typen zurück.

Zusammenfassung

Was haben wir bis hierhin gelernt?

- Typen:
  - Haben Wertebereich und zulässige Operationen
  - Können umgewandelt werden
- Typsicherheit
  - Muss sichergestellt sein, um lauffähige Software entwickeln zu können
  - durch Typumwandlung: implizit (Sprachunterstützung) und explizit (durch Programmierer)
- Typisierung
  - Stark oder schwach
    - anhand von Merkmalen zur Typumwandlung
  - Dynamisch oder statisch
    - statisch: Typen im Quelltext angeben
    - dynamisch: Typen nicht im Quelltext angegeben

Literatur

[Gü17] Kai Günster. *Einführung in Java*. Rheinwerk Computing, Bonn, 2017.

[Str10] Bjarne Stroustrup. *Einführung in die Programmierung mit C++*. Pearson Studium, München, 2010.

[Str16] Bernhard Lahres; Gregor Raýman; Stefan Strich. *Objektorientierte Programmierung*. Rheinwerk Computing, Bonn, 2016.

[Ull18] Christian Ullenboom. *Java ist auch eine Insel*. Rheinwerk Computing, Bonn, 2018.

[Wik18] Wikipedia. Starke typisierung, 2018. [Online; accessed 16-April-2018].