5. AuD Tafelübung T-C3

Simon Ruderich

24. November 2010

this

this

- nur in Instanzmethoden verfügbar
- nicht in statischen Methoden

Beispiel

```
public class Test {
    int a = 5;

    public void instanzmethode() {
        this.a = 5; // funktioniert
    }
    public static void staticmethode() {
        this.a = 5; // Compiler-Fehler!
    }
}
```

Klassen/Interfaces

- müssen `public` deklariert werden (falls nicht anders angegeben)!

TestInterface.java

```
public interface TestInterface {
    // ...
}
```

TestKlasse.java

```
public class TestKlasse implements TestInterface {
    // ...
}
class Intern { /* falls noetig */ ... }
```

Stack

Stack

- Speicher für (lokale) Variablen
- wächst bei jedem Funktionsaufruf
- neuere Definition von Variablen überdecken ältere
- Speicher wird automatisch freigegeben

Beispiel

```
public void test(int a) { // a, b, c auf dem Stack
    int b = a + 3;
    int c = 0;
    // ...
}
x.test(5);
// a, b und c existieren nicht mehr!
```

Heap (in Java)

Heap

- Speicher für Objekte (=Instanzen/Exemplare)
- wird automatisch angelegt und freigegeben
- kein `free()` wie in C nötig/möglich

Beispiel

```
public Integer test(int a) {
    Integer b = new Integer(a);
    // ...
    return b;
}
Integer c = x.test(5);
// a, b existieren nicht mehr
// c und das erzeugte Integer existieren weiterhin!
```

Garbage-Collector

Fragen

- Wird der Speicher nicht schnell voll?
- Wie werden die Objekte wieder freigegeben?

Lösung

- Javas Garbage-Collector (abgekürzt: GC)
- entfernt automatisch nicht benötigte Objekte
- = sobald es keine Referenzen mehr auf das Objekt gibt

Javas Garbage-Collector

Javas Garbage-Collector (GC)

- entfernt automatisch nicht benötigte Objekte
- läuft in einem eigenen Thread im Hintergrund
- verwendet (grob) Mark-and-Sweep Algorithmus
  (jüngere Objekte werden dabei häufiger geprüft)
- es ist *nicht* voraussagbar, wann der GC läuft
- man kann es *nicht* erzwingen, ist nur eine Bitte an die JVM
  (auch nicht mit `System.gc()`)

Javas Garbage-Collector

Gültigkeitsbereich

- (lokale) Variablen sind nur in ihrem Block (`{ .. }`) gültig
- sobald sie diesen verlassen werden sie ungültig
- sobald die letzte Variable auf ein Objekt ungültig wird, kann der GC dieses Objekt entfernen
- solange noch mindestens *eine* Variable auf ein Objekt zeigt wird es nicht entfernt

Javas Garbage-Collector

Beispiel

```
Person p = new Person("Max");
Person q = p;

p = null;
// Welche Objekte koennen vom GC entfernt werden?

p = new Person("Andrea");
// Welche Objekte koennen vom GC entfernt werden?

q = null;
// Welche Objekte koennen vom GC entfernt werden?

p = null;
// Welche Objekte koennen vom GC entfernt werden?
```

Javas Garbage-Collector

Beispiel

```
Person p = new Person("Max");
Person q = p;
{
    Person r = new Person("Andrea");
    q = null;
}
p = null;

// Welche Objekte koennen ab wann entfernt werden?
```

Codierregeln

Codierregeln

- Richtlinien bezüglich Formatierung, Kommentierung, ...
- sind (nur) für den Programmierer wichtig
- helfen teilweise Fehler zu vermeiden

Die wichtigsten Regeln

- pro Zeile eine Deklaration - möglichst mit Initialisierung
- alle Variablen am Anfang der Datei bzw. Klasse deklarieren
- Einrückungen verwenden (4 Spaces oder 1 Tab pro Level)
- öffnende und schließende Klammern richtig eingerückt
- eine geschweifte Klammer ({ }) pro Zeile
- Mnemonische Namen verwenden - Name gibt Funktion an
- ...

Codierregeln

Der Klassiker

```
// ...
double erg = 0;
for(int i = 0; i < 42; i++)
    System.out.println("Doing " + i);
    erg += DoSomethingVeryImportant(i);
return erg;
```

Codierregeln

Negativ-Beispiel

```
if (Math.sqrt(l)%1==0){
        e = new int [q][q];
        } else { if (q * (q+1) >= l) {
            e = new int [q][q +1];
              int rest = ((q + 1)* q - l);
              e [q-1] = new int [q + 1 - rest];
          } else { e = new int [q + 1] [q + 1];
}}}
```

Codierregeln

Wichtig

- ab jetzt können Verstöße gegen die Codierregeln mit Punkteabzug bestraft werden!
- natürlich nur für schwerwiegende Sachen (siehe letzte Folie)

Snake

Farben

Für Snake werden Farben zum Einfärben der Blöcke benötigt.
Zu finden in `java.awt.Color`.

Fragen

Fragen?