Objektsammlungen

Andreas Zeller

Vgl. Kapitel 4 im BlueJ-Buch.
Bis Kapitel 6 bleiben wir weiter dicht am BlueJ-Buch, das somit die eigentliche Referenz bildet.

Programmieren 2

Andreas Zeller

Konzepte

(Wiederholung vom Dienstag)

- Abstraktion
- Modularisierung
- Klassen definieren Typen
- Klassendiagramm
- Objektdiagramm
- Objektreferenz
- Primitiver Typ
- Objekterzeugung
- Überladen
- Interner Aufruf
- Externer Aufruf
- Debugger

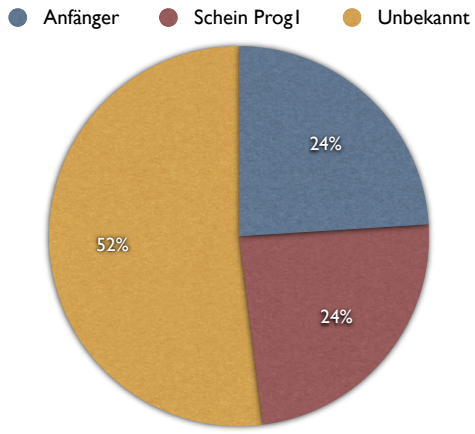
Meinungen

- Die Vorlesung ist…

zu langsam	5
zu schnell	0

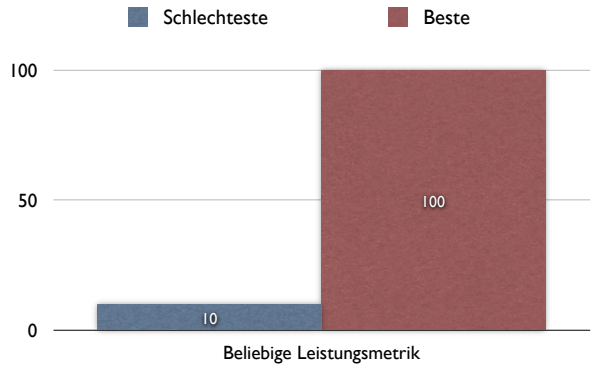
4

Hörer



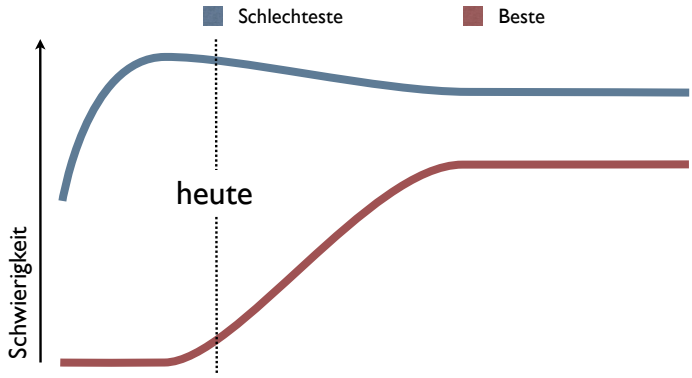
5

Programmierer



6

Die Vorlesung



7

Der kleine Unterschied

```
// 25121xx

import java.io.*;

public String getBye() {
    return "Bye!";
}

public String getHello() {
    return "Tach "+ getName()+ "!";
}
```

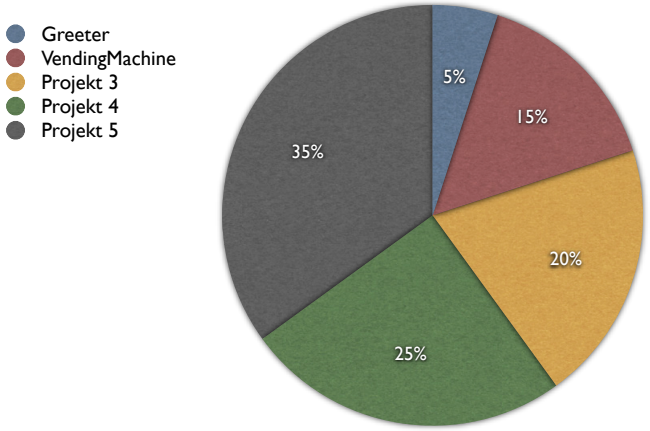
8

Der kleine Unterschied

```
// 25079xx

import java.io.*;

public String getBye() {
    return "Adieu!";
}

public String getHello() {
    return "Tach "+ getName()+ "!";
}
```

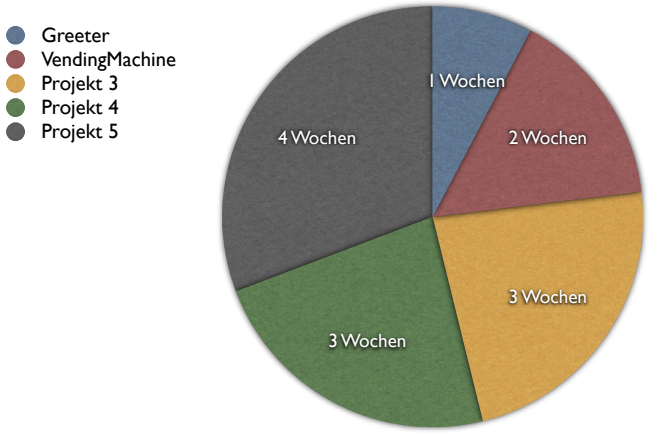
9

Benotung



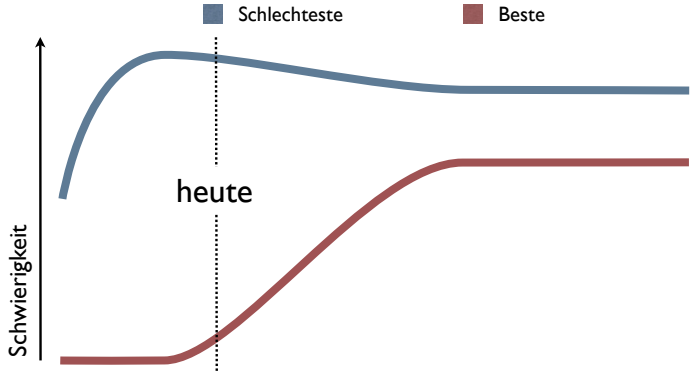
10

Dauer



11

Die Projekte



12

Objektsammlungen

Andreas Zeller

Vgl. Kapitel 4 im BlueJ-Buch.
Bis Kapitel 6 bleiben wir weiter dicht am BlueJ-Buch, das somit die eigentliche Referenz bildet.

Sammlungen in ML

Listen `[1, 2, 3]`
Tupel `(1, 2)`
Vektoren `Vector.sub(v, i)`
Mengen `set[1, 2, 3]`

Sammlungen in Java

29 Stück

AbstractCollection • AbstractList • AbstractQueue • AbstractSequentialList • AbstractSet • ArrayBlockingQueue • ArrayList • AttributeList • BeanContextServicesSupport • BeanContextSupport • ConcurrentLinkedQueue • CopyOnWriteArrayList • CopyOnWriteArraySet • DelayQueue • EnumSet • HashSet • JobStateReasons • LinkedBlockingQueue • LinkedHashSet • LinkedList • PriorityBlockingQueue • PriorityQueue • RoleList • RoleUnresolvedList • Stack • SynchronousQueue • TreeSet • Vector

Collection (Java 2 Platform SE 5.0) 04/26/2007 10:33 PM

Overview Package Class Use Tree Deprecated Index Help *Java™ 2 Platform Standard Ed. 5.0*

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

java.util

Interface Collection<E>

All Superinterfaces:
Iterable<E>

All Known Subinterfaces:
BeanContext, BeanContextServices, BlockingQueue<E>, List<E>, Queue<E>, Set<E>, SortedSet<E>

All Known Implementing Classes:
AbstractCollection, AbstractList, AbstractQueue, AbstractSequentialList, AbstractSet, ArrayBlockingQueue, ArrayList, AttributeList, BeanContextServicesSupport, BeanContextSupport, ConcurrentLinkedQueue, CopyOnWriteArrayList, CopyOnWriteArraySet, DelayQueue, EnumSet, HashSet, JobStateReasons, LinkedBlockingQueue, LinkedHashSet, LinkedList, PriorityBlockingQueue, PriorityQueue, RoleList, RoleUnresolvedList, Stack, SynchronousQueue, TreeSet, Vector

```
public interface Collection<E>
extends Iterable<E>
```

The root interface in the *collection hierarchy*. A collection represents a group of objects, known as its *elements*. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. The JDK does not provide any *direct* implementations of this interface: it provides implementations of more specific subinterfaces like `Set` and `List`. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

16

Sammlungen in Java

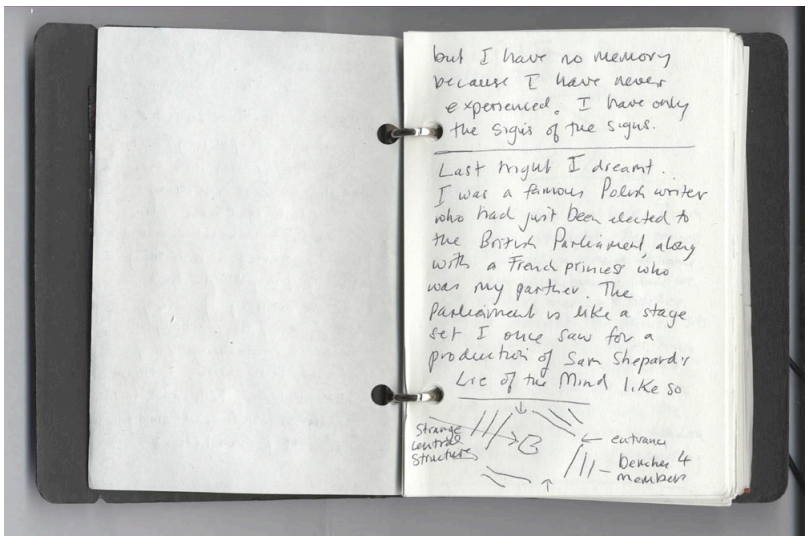
AbstractCollection • AbstractList • AbstractQueue • AbstractSequentialList • AbstractSet • ArrayBlockingQueue • ArrayList • AttributeList • BeanContextServicesSupport • BeanContextSupport • ConcurrentLinkedQueue • CopyOnWriteArrayList • CopyOnWriteArraySet • DelayQueue • EnumSet • HashSet • JobStateReasons • LinkedBlockingQueue • LinkedHashSet • LinkedList • PriorityBlockingQueue • PriorityQueue • RoleList • RoleUnresolvedList • Stack • SynchronousQueue • TreeSet • Vector

17

Sammlungen in Java

AbstractCollection • AbstractList • AbstractQueue • AbstractSequentialList • AbstractSet • ArrayBlockingQueue • ArrayList • AttributeList • BeanContextServicesSupport • BeanContextSupport • ConcurrentLinkedQueue • CopyOnWriteArrayList • CopyOnWriteArraySet • DelayQueue • EnumSet • HashSet • JobStateReasons • LinkedBlockingQueue • LinkedHashSet • LinkedList • PriorityBlockingQueue • PriorityQueue • RoleList • RoleUnresolvedList • Stack • SynchronousQueue • TreeSet • Vector

18



Demo: Notebook

ArrayList: Array wechselnder Größe - Generische Klassen - Indizes - Items entfernen - ungültige Indizes - for-each - while - break(!)

searchNote()

```
/**
 * Search a note
 */
public String searchNote(String searchString)
{
    for (String note : notes) {
        if (note.contains(searchString)) {
            return note;
        }
    }
    return "";  // Note not found
}
```

22

Demo: Auktionssystem

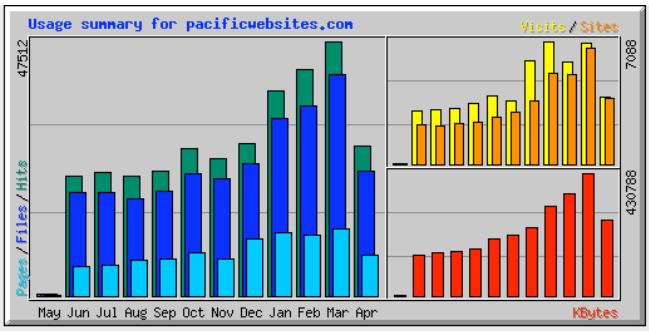
Lot - null - Auction - Anonyme Objekte

23

Sammlungen

flexible Größe	feste Größe
aus java.util.*	eingebaut
nehmen nur Objekte auf	Objekte + primitive Typen
zahlreiche Methoden	effizienter Zugang

24



Summary by Month

Month	Daily Avg				Monthly Totals					
	Hits	Files	Pages	Visits	Sites	KBytes	Visits	Pages	Files	Hits
Apr 2004	1332	1112	358	185	3770	267690	3901	7537	23364	27992
Mar 2004	1532	1335	404	224	6696	430788	6946	12549	41405	47512
Feb 2004	1457	1223	393	204	5175	358145	5922	11414	35495	42267
Jan 2004	1233	1065	377	228	5211	316035	7088	11713	33020	38237

25

Weblog.txt

Jahr • Monat • Tag • Stunde • Minute

```
2007 04 23 21 07
2007 04 23 22 13
2007 04 24 07 35
2007 04 25 14 59
2007 04 26 22 59
...
```

26

Demo: Weblogs

Felder fester Größe - for-Schleife - ++, += -

27

Arrays in C

```
#include <stdio.h>

int main()
{
    char name[20];
    int balance = 50;

    printf("Please enter your name: ");
    gets(name);
    printf("Hello, %s!\n", name);
    printf("Your balance is %i EUR.\n", balance);

    return 0;
}
```


Schleifen

for-each	`for (ElementType element: collection) {`
	`  loop body`
	`}`

while	`while (condition) {`
	`  loop body`
	`}`

for	`for (init; condition; post-body) {`
	`  loop body`
	`}`

for–each, while, for
do–while bleibt außen vor – zu fehleranfällig

Schleifen

for-each	um *alle Elemente* einer *Sammlung* zu durchlaufen
while	für eine *vorher unbekannte* Anzahl Schleifendurchläufe
for	für eine *vorher bekannte* Anzahl Schleifendurchläufe

for–each, while, for

Konzepte

- Sammlungen
  - flexibler Größe: ArrayList
  - fester Größe: Array
- Iteratoren
- null
- Schleifen
  - for each
  - while
  - for
  - break
- Überlauf