Karlsruher Institut für Technologie

5. Tutorium zu Programmieren

Wiederholung, final und static

Christof Urbaczek | 23.11.2012

VERIFIKATION TRIFFT ALGORITHMIK AM INSTITUT FÜR THEORETISCHE INFORMATIK

```
public boolean attachCellToPlayer(Player player, GameCell cell, int value) throws IllegalArgumentExc
    if (cell == null || player == null || value <= DataCell.NOT_SET) {
        throw new IllegalArgumentException("invalid argument given.");
    }
    MultiplayerPlayerSlot involvedPlayer = (MultiplayerPlayerSlot) getPlayerSlo
    GameCell gameCell = getGameCellByIndex(cell.getIndex(), this.sudoku.getFiel
    if (!gameCell.isOwnerPending()) throw new IllegalStateException();

    boolean result = false;
    if (!this.isAborted()) {
        if (value == gameCell.getSolution()) {
            result = gameCell.attachToPlayer(involvedPlayer);
            involvedPlayer.getScore().increment(POSITIVE_INCREMENT);
            onChange(gameCell);
            if (successfullySolved(this.sudoku)) {
                onSuccessfullyFinish(player);
            }
        }
    }
    return result;
```

Übersicht

Wiederholung

Frage 1

Frage 1

Was ist ein Compiler?

Frage 2

Frage 2

Wie kompiliere und starte ich ein Java-Programm in der Konsole?

Frage 3

Frage 3

Was versteht man unter OOP?

Frage 4

Frage 4

Was sind "code conventions"?

Frage 5

Frage 5

Was ist der Unterschied zwischen primitiven Datentypen und Referenztypen?

Frage 6

Frage 6

Was geschieht in diesem Codeausschnitt?

```
int a = 4;
```

Frage 7

Frage 7

Was ist die main-Methode?

Frage 8

Frage 8

Wozu dient der Konstruktur einer Klasse?

Frage 9

Frage 9

Was ist eine Methodensignatur?

Frage 10

Frage 10

Was ist eine statische Methode?

Frage 11

Frage 11

Was ist ein *typecast*?

Frage 12

Frage 12

Was geschieht in diesem Codeausschnitt?

```
(byte)200
```

Frage 12

Was geschieht in diesem Codeausschnitt?

```
(byte)200
```

Ausgabe: -56

Wieso?

Frage 13

Frage 13

Was bedeutet *void* und wo findet es Verwendung?

Frage 14

Frage 14

Der Vergleich welcher Variablen ergibt *true*?

```
int a = 20;

int b = 10;

int c = 20;

Car d = new Car("Polo");

Car e = new Car("Golf");

Car f = new Car("Polo");
```

Frage 15

Frage 15

Was geschieht in diesem Codeausschnitt?

```
int[] a = new int[5]
```

Frage 16

Frage 16

Was ist hier falsch?

```
puplic void main ( Char[] x ) {}
```

Frage 17

Frage 17

Kompiliert folgender Code?

```
1  int x = 42;
2
3  if (x = 42) {
4      System.out.println("Das Leben");
5  } else {
6      System.out.println("Das Universum");
7  }
```

Frage 18

Frage 18

Was wird ausgegeben?

```
1  boolean a = true;
2  boolean b = false;
3  boolean c = true;
4
5  int d = 0;
6  boolean e = false;
7
8  if (a && b && c) System.out.println("Alpha");
9  if (d) System.out.println("Beta");
10 if (a||b && b||e) System.out.println("Gamma");
```

Frage 19

Frage 19

Was wird ausgegeben?

```
1  int a = 1;
2  if ( a/10 == 0.1) {
3      System.out.println("wahr");
4  } else {
5      System.out.println("falsch");
6  }
```

Frage 20

Frage 20

Was wird ausgegeben?

```
1  double a = 10.0;
2  double b =  9.9;
3
4  if ( a - b == 0.1) {
5      System.out.println("wahr");
6  } else {
7      System.out.println("falsch");
8  }
```

Frage 21

Frage 21

Was macht diese Funktion?

```
1  int[] arr = new int[]{1,2,3,4,5};
2  int x = 1;
3  for (int a : arr) x *= a;
```

Frage 22

Frage 22

Was wird ausgegeben?

```
1   int monat = 2;
2
3   switch (monat) {
4       case 1 : System.out.println("Januar"); break;
5       case 2 : System.out.println("Februar");
6       case 3 : System.out.println("Maerz"); break;
7       //...
8       case 12 : System.out.println("Dezember"); break;
9       default : System.out.println("Unbekannter Monat");
10  }
```

Frage 23

Frage 23

Was wird ausgegeben?

```
1  //Zaehle bis 200 und beginne zu suchen
2  byte counter = 1;
3  while (counter <= 200) {
4      System.out.println(counter);
5      counter++;
6  }
7
8  System.out.println("Ich komme ... !");
```

Frage 24

Frage 24

setParams(10,10) - Was steht in a und b?

```
1   class ParamTest {
2
3       int a;    //Bedingung  0 <= a <= 9
4       int b;    //Bedingung  0 <= b <= 9
5
6       void setParams(int a, int b) {
7           //Bedingungen sicherstellen
8           if ( a > 9 ) a = 9;
9           if ( a < 0 ) a = 0;
10          if ( b > 9 ) b = 9;
11          if ( b < 0 ) b = 0;
12          //Werte speichern
13          a = a;
14          b = b;
15      }
16
17  }
```

Frage 25

Frage 25

setParams(10,10) - Was steht in a und b?

```
1   class ParamTest {
2
3       int a;    //Bedingung  0 <= a <= 9
4       int b;    //Bedingung  0 <= b <= 9
5
6       void setParams(int a, int b) {
7           //Werte speichern
8           this.a = a;
9           this.b = b;
10          //Bedingungen sicherstellen
11          if ( a > 9 ) a = 9;
12          if ( a < 0 ) a = 0;
13          if ( b > 9 ) b = 9;
14          if ( b < 0 ) b = 0;
15      }
16
17  }
```

Frage 26

Frage 26

setParams(10,10) - Was steht in a und b?

```
1   class ParamTest {
2
3       int a;    //Bedingung  0 <= a <= 9
4       int b;    //Bedingung  0 <= b <= 9
5
6       // Weitere Bedingung: b > a, sonst b auf a+1 setzen
7       void setParams(int a, int b) {
8           //Bedingungen sicherstellen
9           if ( a > 9 ) a = 9;
10          if ( a < 0 ) a = 0;
11          if ( b > 9 ) b = 9;
12          if ( b < 0 ) b = 0;
13          if ( a >= b ) b = a + 1;
14          //Wert speichern
15          this.a = a;
16          this.b = b;
17      }
18
19  }
```

Frage 26

und bei setParams(5,1)?

```
1   class ParamTest {
2
3       int a;    //Bedingung  0 <= a <= 9
4       int b;    //Bedingung  0 <= b <= 9
5
6       // Weitere Bedingung: b > a, sonst b auf a+1 setzen
7       void setParams(int a, int b) {
8           //Bedingungen sicherstellen
9           if ( a > 9 ) a = 9;
10          if ( a < 0 ) a = 0;
11          if ( b > 9 ) b = 9;
12          if ( b < 0 ) b = 0;
13          if ( a >= b ) b = a + 1;
14          //Wert speichern
15          this.a = a;
16          this.b = b;
17      }
18
19  }
```

Frage 27

Frage 27

Was wird ausgegeben?

```
1  int a = 0;
2  while ( a < 10 ); {
3      a++;
4  }
5  System.out.println(a);
```

Das static-Schlüsselwort

statische Attribute

Bisher:

- jedes Attribut ist an ein konkretes Objekt einer Klasse gebunden
  - Aufruf über Bezugsobjekt
- jedes Attribut ex. im Speicher **einmal je Objekt**

statische Attribute

Bisher:

- jedes Attribut ist an ein konkretes Objekt einer Klasse gebunden
  - Aufruf über Bezugsobjekt
- jedes Attribut ex. im Speicher **einmal je Objekt**

statische Attribute

- `static type name = ... ;`
- Attribut ist nicht mehr an konkretes Objekt gebunden
  - Aufruf über Bezugsklasse: *Klassenname.attribut*

statische Attribute

Bisher:

- jedes Attribut ist an ein konkretes Objekt einer Klasse gebunden
  - Aufruf über Bezugsobjekt
- jedes Attribut ex. im Speicher **einmal je Objekt**

statische Attribute

- `static type name = ... ;`
- Attribut ist nicht mehr an konkretes Objekt gebunden
  - Aufruf über Bezugsklasse: *Klassenname.attribut*
- jedes Attribut ex. im Speicher **genau einmal**
  - auch wenn noch kein Objekt erzeugt wurde!

statische Attribute

statische Attribute

- `static type name = ... ;`
- Attribut ist nicht mehr an konkretes Objekt gebunden
  - Aufruf über Bezugsklasse: *Klassenname.attribut*
- jedes Attribut ex. im Speicher **genau einmal**
  - auch wenn noch kein Objekt erzeugt wurde!

mögliche Einsatzgebiete

- Objektzähler/Instanzenzähler
- Alle Instanzen teilen sich einen Attributwert
- **Konstanten**

statische Methoden

Bisher:

- jedes Methode ist an ein konkretes Objekt einer Klasse gebunden
  - Aufruf über Bezugsobjekt
    - wenn nicht angegeben, implizit *this*

statische Methoden

Bisher:

- jedes Methode ist an ein konkretes Objekt einer Klasse gebunden
  - Aufruf über Bezugsobjekt
    - wenn nicht angegeben, implizit *this*

Was bedeutet das für die main-Methode unseres Programms?
Zu welchem Zeitpunkt wird sie aufgerufen und von wem?

statische Methoden

Lösung: statische Methoden

- Signatur:

```
static Rueckgabetyp name (formale Parameter)
```

- z.B.

```
static void main (String[] args)
```

- Methode ist nicht mehr an konkretes Objekt gebunden
  - Aufruf über Bezugsklasse: *Klassenname.methode(...)*
  - kein Zugriff auf nicht-statische Attribute und Methoden innerhalb einer statischen Methode
  - *this* nicht verfügbar

statische Methoden

mögliche Einsatzgebiete

- Methoden, die nicht auf den Daten eines Objekts arbeiten
  - „Arbeitswerte“ sind ausschließlich die Parameter
  - mathematische Funktionen, s. Klasse **Math**
- Methoden in Hilfsklassen
  - z.B. zum Prüfen, ob ein Objekt bestimmte Bed. erfüllt

Das final-Schlüsselwort

final-Attribute

final-Attribute

- Deklaration:

```
final type name;
```

final-Attribute

final-Attribute

- Deklaration:

```
final type name;
```

- einem als *final* markierten Attribut kann **genau einmal** ein Wert zugewiesen werden:
  - direkt bei Deklaration
    oder
  - in den Konstruktoren

final-Attribute

mögliche Einsatzgebiete

- wenn ein Attribut seinen Initialwert nicht ändern wird/soll
- **Konstanten**

final-Attribute

mögliche Einsatzgebiete

- wenn ein Attribut seinen Initialwert nicht ändern wird/soll
- **Konstanten**
  - `static final type NAME = wert;`
    - `Math.PI`
    - `Integer.MAX_VALUE`
    - `static final int DEFAULT_SIZE = 42;`
    - `static final String MESSAGE = "Bitte geben Sie einen Wert ein!";`

Abspann

Fragen?

Bis nächste Woche ...