



Assignments in 1st semester course „OOP with Java“ as small projects

Michael Ritzschke

Humboldt-Universität zu Berlin
Department of Computer Science
Software Engineering

10th Workshop “Software Engineering Education and Reverse Engineering”
Ivanjica, Serbia , 5 September – 12 September 2010

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

Contents

- Programming lab “OOP with Java” (Overview)
- Details: Structure, Requirements
- Example: Sudoku checker and Sudoku solver
- Shell scripts allows partially automatical evaluation
- Statistic WS2009/2010 and summary

Programming lab "OOP with Java" in WS 0910 (Overview)

- Bachelor students (94 registrations)
- 12 assignments, handling time 2 weeks
- Submission the solutions with our special management system "goya", upload packed file
- Success: 50 % of the reachable points

Programming lab website

Grundlagen der Programmierung

Aufgaben



Wintersemester 2009/2010

Mono-Bachelor Informatik
und Mathematik

Kombi-Bachelor mit
Kernfach Informatik

Aufgaben

Betreuer

Software

Scheine

Bitte achten Sie immer darauf, dass Sie Ihre Lösungen bei [Goya](#) unter der richtigen Aufgabennummer einsenden!

Aufgabe Nr.	Bearbeitungszeitraum	Punkte	Thema
0	19.10.2009 - 02.11.2009	0	Einstieg
1	26.10.2009 - 09.11.2009	10	Texte und Mails
2	02.11.2009 - 16.11.2009	20	Arbeiten im Unix-Filesystem
3	09.11.2009 - 23.11.2009	10	Homepage
4	16.11.2009 - 30.11.2009	10	Mein erstes Java-Programm
5	23.11.2009 - 07.12.2009	10	Primfaktoren
6	30.11.2009 - 14.12.2009	20	Große Zahlen in Feldern
7	07.12.2009 - 21.12.2009	20	Kleingeld
8	04.01.2010 - 18.01.2010	20	Mastermind
9	11.01.2010 - 25.01.2010	20	Im Supermarkt
10	18.01.2010 - 01.02.2010	30	Sudoku-Checker
11	25.01.2010 - 08.02.2010	30	Sudoku-Solver

The Bachelor students java assignments (overview WS0910)

Mein erstes Java-Programm

Which weekday has a given date (formula from Zeller)

Primfaktoren

Prime factorization

Große Zahlen in Feldern

Big numbers in arrays, different mathematical operations

Kleingeld

How many combinations of coins can realize a given value

Mastermind

Well known game

Im Supermarkt

Model of a supermarket

Sudoku-Checker

Test correctness of two Sudoku kinds: with diagonal line or without

Sudoku-Solver

Solves 9 x 9 Sudokus, if possible



```
$ java testSudokuChecker < s6
Sudoku
+-----+-----+
| 8      |         | 6      |
| 3 9    |         | 1 5    |
|         | 4 9 7  |         |
+-----+-----+
|         | 5 | 2 |         | 3 |         |
| 2 6    |   |   |         | 7 1 |
|         | 4 | 7 |         | 6    |
+-----+-----+
|         | 6 1 8 |         | | | |
| 2 1    |   |   |         | 9 6 |
| 6      |   |   |         |     | 4 |
+-----+-----+
ok
```

6	9	3	7	8	4	5	1	2
4	8	7	5	1	2	9	3	6
1	2	5	9	6	3	8	7	4
9	3	2	6	5	1	4	8	7
5	6	8	2	4	7	3	9	1
7	4	1	3	9	8	6	2	5
3	1	9	4	7	5	2	6	8
8	5	6	1	2	9	7	4	3
2	7	4	8	3	6	1	5	9

Other interesting little java projects

Management for a calendar of birthdays

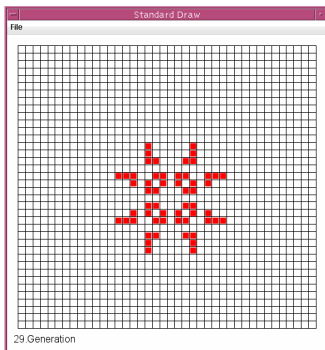
The game of Life

Computation of special primes, e.g. weakly primes, gaps of primes, twin primes ...

Some calculations for credits, e.g. interest rates, duration ...

The game Peg Solitaire

Management of foods in a fridge



A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

Contents

- Programming lab “OOP with Java” (Overview)
- Details: Structure, Requirements
- Example: Sudoku checker and Sudoku solver
- Shell scripts allows partially automatical evaluation
- Statistic WS2009/2010 and summary

The assignment structure

- Summary

10th assignment: *Sudoku*

数独

Part 1: The checker

Handling time: 18.01.10 - 01.02.10, 10.00 Uhr

Points: 30 points

Goal: Please implement in Java a Sudoku checker, i.e. a class for testing correctness of 9x9-Sudokus

Requirements: Knowledge from the lecture about objects and classes in Java.

Exercise: You implement a class **Sudoku**, which is able to build and to write a 9x9 Sudoku and a class **Checker**, which is able to check the correctness

FAQ: At the end of the website you find [answers to often ask questions](#) and (later) [concretizations of the assignment](#). Please note them.

Submission: Send your classes **Sudoku.java** and **Checker.java** whith [Goya](#) as your solution. Use exactly the given filenames and regard case sensitivity

- Job description

Job description

We think, you know Sudokus :-), otherwise read <http://de.wikipedia.org/wiki/Sudoku>, please.

Your job in the 2 last assignments is to implement a Sudoku-Solver, which is able to solve any Sudoku or to find out, that it is impossible. The first step (Sudoku-Checker): Find out the correctness of a given 9x9 Sudoku, that means at first: **Sudoku.read()** and **Sudoku.write()**:

```

.743..9..
...1...4.
6...2..1
2...7.8..
86...45..
..1.....
7.6.....
3...62..
.....
                
```

Sudoku

7	4	3		9
		1		4
6		2		1
2		7		8
8	6		4	5
	1			
7	6			
3		6	2	

The requirements (relevant to all solutions)

ATTENTION

Follow our rules, please:

1. The **compiler must accept your program**. If it does not accept, you get without any control **0 points** !
2. The **generated results** of your program must have **exactly the same format as** the given **reference result** (including the line structure and the whitespaces). Often the results are automatically compared.
3. Prepare all classes and methods with a **short annotation** - this is **obligatory**.

FAQ

1. **Q:** Questions?
A: yet no!

This was the 10th assignment! Godspeed!

Last modification 15.1.2010
Please send your questions to: [Ahrens](#)

Different kinds of assignments

- Only the expected result is given
- Additionally students get a frame with the declarations of classes and methods
- Students get a description like javadoc result

Package **Class** Use Tree Deprecated Index Help
 PREV CLASS NEXT CLASS
 SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)
[FRAMES](#) [NO FRAMES](#)
 DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

Class StringHash

java.lang.Object
 extended by **StringHash**

```
public class StringHash
extends java.lang.Object
```

Die Klasse StringHash traegt Strings in eine geschlossenen Hashtabelle ein. Die vorgegebenen Hash-Implementation besitzt zwei Parameter:

Constructor Summary

[StringHash](#)(int size, int factor)
 Konstruktor mit den Parametern Tabellenlaenge und Faktor (der Hashfunktion).

Method Summary

int	colls () Anzahl der bisher aufgetretenen Kollisionen.
private int	hash (java.lang.String key) Hashfunktion berechnet einen Wert w mit $0 \leq w \leq \text{size}-1$ also einen Index des arrays (oder Vektors) table unter Nutzung der Laenge des gegebenen key und der Werte der Zeichen des key.
double	load () Berechnet die Auslastung.
private boolean	lookup (java.lang.String key) Testet, ob ein key bereits in der Tabelle eingetragen ist.
static void	main (java.lang.String[] s) Beispiel fuer die Nutzung der Klasse: Aufruf des Konstruktors, Eintragen von Strings, Ausgabe der Hashtabelle. Erwartete Ausgaben: Siehe Kommentar zu printHashtable().

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

Contents

- Programming lab “OOP with Java” (Overview)
- Details: Structure, Requirements
- Example: Sudoku checker and Sudoku solver
- Shell scripts allows partially automatical evaluation
- Statistic WS2009/2010 and summary

Example: Sudoku checker

- Students get the frames of 3 classes
- TODO: Constructors (data structure); functions read, write, check

```
class Sudoku {
    public enum Variant { normal, withDiagonalLines };
    public Variant v;
    // what is furthermore necessary ...
    public Sudoku(Variant v) { /* TODO */ }
    public void read() { /* TODO */ }
    public void write() { /* TODO */ }
}
```

```
class testSudokuChecker {
    public static void main (String args []) {
        Sudoku.Variant v = Sudoku.Variant.normal;
        if (args.length > 0 && args[0].equals("-x"))
            v = Sudoku.Variant.withDiagonalLines;
        Sudoku s = new Sudoku(v);

        s.read();
        s.write();
        Checker checker = new Checker(s);
        if (!checker.check()) {
            System.out.print("not ");
        }
        System.out.println("ok");
    }
}
```

```
class Checker {
    public Checker(Sudoku s) { /* TODO */ }
    public boolean check() { /* TODO */ }
}
```

```
.743..9..
...1...4.
6....2..1
2...7.8..
86...45..
..1.....
7.6.....
3....62..
.....
```

```
$ java testSudokuChecker < s4
Sudoku
+-----+-----+-----+
|   |   |   |   |
|  1 |  2 |  3 |   |
|   |   |   |   |
+-----+-----+-----+
|   |   |   |   |
|  4 |  5 |  6 |   |
|   |   |   |   |
+-----+-----+-----+
|   |   |   |   |
|  7 |  8 |  9 |   |
|   |   |   |   |
+-----+-----+-----+
ok
```

Solution: Internal data structure

```
/**
 * Represents a sudoku that shall be checked or solved.
 *
 * @author P. S., Matrikel-Nr: 533567
 */
class Sudoku {
    public enum Variant { normal, withDiagonalLines };
    public Variant v;
    public int[][] sudoku; // 2-dimensional field for storing the numbers in the sudoku

    /**
     * Creates a new sudoku of a given variant.
     *
     * @param v
     *             variant
     */
    public Sudoku(Variant v) {
        sudoku = new int[9][9];
        this.v = v;
    }
}
```

Solution: Function read()

```
/**
 * Reads data linewise from the standard input and if the input is correct,
 * the characters are written into the 2-dimensional sudoku-field. The digits
 * 1 to 9 are written into the field, as they are. All other characters are
 * written as 0.
 * If the input isn't correct, an error message is printed and the program
 * terminates.
 */
public void read() {
    try {
        for (int i=0; i < 9; i++) {
            String tmp = Keyboard.input.readLine();
            if (tmp.length() < 9) {
                System.err.println("input error");
                System.exit(1);
            }
            for (int j=0; j < 9; j++) {
                if ("123456789".indexOf(tmp.substring(j,j+1)) == -1) sudoku[i][j] = 0;
                else sudoku[i][j] = Integer.parseInt(tmp.substring(j,j+1));
            }
        }
    } catch (Exception e) {
        System.err.println("input error");
        System.exit(1);
    }
}
```

Solution: Function write()

```

/**
 * Prints the sudoku on the standard output. At first the variant of the sudoku
 * is printed, followed by a framed representation of the sudoku.
 */
public void write() {
    System.out.print("Sudoku");
    if (v.toString().equals("withDiagonalLines")) System.out.println("withDiagonalLines");
    else System.out.println();
    System.out.println("+-----+-----+-----+");
    for (int i=0; i < 9; i++) {
        System.out.print("| ");
        for (int j=0; j < 9; j++) {
            if (sudoku[i][j] != 0) System.out.print(sudoku[i][j]+" ");
            else System.out.print("  ");
            if (j % 3 == 2) System.out.print("| ");
        }
        System.out.println();
        if (i % 3 == 2) System.out.println("+-----+-----+-----+");
    }
}

```

Sudoku								
	7	4		3			9	
				1			4	
	6				2			1
+-----+-----+-----+								
	2				7		8	
	8	6					4	5
				1				
+-----+-----+-----+								
	7	6						
	3				6		2	
+-----+-----+-----+								

Function check(): Calls checkLines and others ...

- Checks all lines, rows, 3x3 blocks and diagonal lines

```
/**
 * Checks the lines of a sudoku.
 * Therefore it checks for each element in a line, whether it has the same value as an
 * subsequent element.
 *
 * @return true, if the lines are correct, false otherwise.
 */
public boolean checkLines() {
    for (int i=0; i<9; i++) {
        for (int j=0; j<9; j++) {
            if (s.sudoku[i][j] != 0) {
                for (int k=j+1; k<9; k++) {
                    if (s.sudoku[i][j] == s.sudoku[i][k]) return false;
                }
            }
        }
    }
    return true;
}
```


The (hopefully successful) test of the written program with some given different input files



```
0000000000
010020030
0000000000
0000000000
040050060
0000000000
0000000000
070080090
0000000000
```

```
adler ritzschk 36 ( ../10/testsudoku ) > java testSudokuChecker < s4 -x
SudokuwithDiagonalLines
+-----+-----+-----+
| 1 | 2 | 3 |
+-----+-----+-----+
| 4 | 5 | 6 |
+-----+-----+-----+
| 7 | 8 | 9 |
+-----+-----+-----+
ok
```

```
8      6
39     15
      497
      52 3
26     71
      47 6
      618
21     96
6      4
```

```
adler ritzschk 37 ( ../10/testsudoku ) > java testSudokuChecker < s6 -x
SudokuwithDiagonalLines
+-----+-----+-----+
| 8 |   | 6 |
| 3 9 |   | 1 5 |
|   | 4 9 7 |   |
+-----+-----+-----+
| 5 | 2 | 3 | |
| 2 6 |   | 7 1 |
|   | 4 | 7 | 6 |
+-----+-----+-----+
|   | 6 1 8 |   | |
| 2 1 |   | 9 6 |
| 6 |   |   | 4 |
+-----+-----+-----+
not ok
```

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

Next assignment: Sudoku solver

- Possible solution algorithm: Backtracking
- Variable fields and constant fields
- Fill step by step all variable fields by increasing numbers and check after increasing if the sudoku is still correct
- If correct, do the same for the next field, if not, clear the field and go back to the last correct field and continue with the last correct field

Backtracking needs time ...

```

adler ritzschk 89 ( .../11/86880 ) > java testSudokuSolver < s7
Sudoku
+-----+-----+-----+
|   |   |   | 8 3 |
|   | 4 |   |   |
| 3 | 1 |   | 9  |
+-----+-----+-----+
|   | 6 |   | 9  |
| 2 4 8 | 9  | 3  |
|   |   | 4 | 7 2 |
+-----+-----+-----+
| 6 |   | 8 |   |
| 5 8 2 | 9  |   | 4  |
|   | 1 |   |   |
+-----+-----+-----+
Loesung:
Sudoku
+-----+-----+-----+
| 6 1 4 | 2 7 9 | 8 3 5 |
| 9 2 5 | 4 8 3 | 1 6 7 |
| 8 3 7 | 6 1 5 | 2 4 9 |
+-----+-----+-----+
| 1 7 6 | 5 3 2 | 4 9 8 |
| 2 4 8 | 9 6 7 | 3 5 1 |
| 3 5 9 | 8 4 1 | 6 7 2 |
+-----+-----+-----+
| 4 6 1 | 7 5 8 | 9 2 3 |
| 5 8 2 | 3 9 6 | 7 1 4 |
| 7 9 3 | 1 2 4 | 5 8 6 |
+-----+-----+-----+
18.77u 0.09s 0:18.83 100.1%
  
```

easy
and
difficult
Sudoku

```

+-----+-----+-----+
|   |   |   | 1  |
| 4 | 2 |   |   |
+-----+-----+-----+
|   |   | 5 | 4 7 |
| 8 |   | 9 | 3  |
| 1 |   |   |   |
+-----+-----+-----+
| 3 | 4 |   | 2  |
| 5 | 1 |   |   |
|   | 8 6 |   |   |
+-----+-----+-----+
Loesung:
Sudoku
+-----+-----+-----+
| 6 9 3 | 7 8 4 | 5 1 2 |
| 4 8 7 | 5 1 2 | 9 3 6 |
| 1 2 5 | 9 6 3 | 8 7 4 |
+-----+-----+-----+
| 9 3 2 | 6 5 1 | 4 8 7 |
| 5 6 8 | 2 4 7 | 3 9 1 |
| 7 4 1 | 3 9 8 | 6 2 5 |
+-----+-----+-----+
| 3 1 9 | 4 7 5 | 2 6 8 |
| 8 5 6 | 1 2 9 | 7 4 3 |
| 2 7 4 | 8 3 6 | 1 5 9 |
+-----+-----+-----+
794.37u 0.32s 13:14.50 100.0%
  
```

Results of a competition

The winners of the Sudoku competition are known now.

We measured the time required for solving of 2 x 27 Sudokus (with and without diagonal lines).

Congratulation to 10 additional points!

Tobias Schall	0:12
Denis Erfurt	1:02
Tom Bierschenk	1:22
Malte Schmidt	1:56
David Hadizadeh	4:40
Anna-Lisa Deussing	11:22
Paul Scherer	13:37
Jan Lelis	13:55
Andreas Schuldt	18:30
Sebastian Klemke	27:11

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

Contents

- Programming lab “OOP with Java” (Overview)
- Details: Structure, Requirements
- Example: Sudoku checker and Sudoku solver
- Shell scripts allows partially automatical evaluation
- Statistic WS2009/2010 and summary

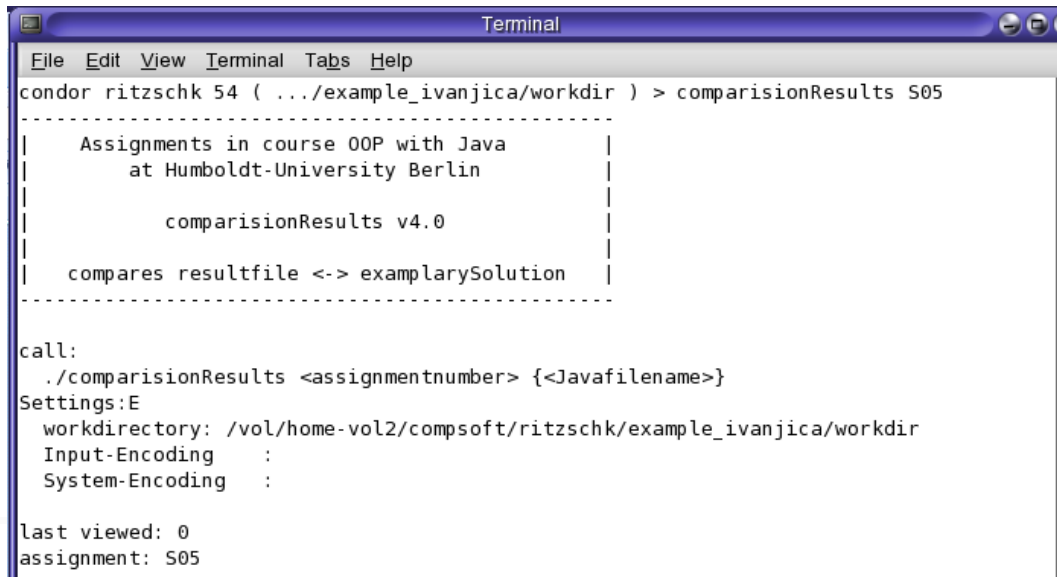
Shell script revise ...

- Reads the goya-files with the uploaded programs
- Generates directories with the extracted solutions
- Copies the testing program and the expected pattern solution "solution.out"
- Compiles the files and starts the testing program

```
./assignmentS05:  
numbers          results          solutions          solutions.zip  
  
./assignmentS05/results:  
E05482 E18456 E28506 E30785 E40881  
  
./assignmentS05/solutions:  
05482          18456          28506          30785          40881          unzipReport  
  
./assignmentS05/solutions/05482:  
TestQuader2.java quader.java          solution.out          test.out
```

Shell script comparisionResults ...

- Compares the produced result Exxxx with the expected exemplary output
- Uses the command diff (or a tool, for instance kDiff3, look <http://kdiff3.sourceforge.net/>)

A screenshot of a terminal window titled 'Terminal'. The prompt is 'condor ritzschk 54 (../example_ivanjica/workdir) > comparisionResults S05'. The output shows a dashed box containing the text: 'Assignments in course OOP with Java at Humboldt-University Berlin', 'comparisionResults v4.0', and 'compares resultfile <-> exemplarySolution'. Below the box, it says 'call: ./comparisionResults <assignmentnumber> {<Javafilename>}' and 'Settings:E workdirectory: /vol/home-vol2/compssoft/ritzschk/example_ivanjica/workdir Input-Encoding : System-Encoding :'. At the bottom, it shows 'last viewed: 0' and 'assignment: S05'.

```
Terminal
File Edit View Terminal Tabs Help
condor ritzschk 54 ( ../example_ivanjica/workdir ) > comparisionResults S05
-----
|   Assignments in course OOP with Java   |
|   at Humboldt-University Berlin       |
|                                     |
|   comparisionResults v4.0             |
|                                     |
|   compares resultfile <-> exemplarySolution |
|                                     |
|-----|
call:
./comparisionResults <assignmentnumber> {<Javafilename>}
Settings:E
workdirectory: /vol/home-vol2/compssoft/ritzschk/example_ivanjica/workdir
Input-Encoding :
System-Encoding :

last viewed: 0
assignment: S05
```

Shell script plagiarismSearch ...

- compares the java files and – if we want – the result files, also with the command diff

```
Terminal
File Edit View Terminal Tabs Help

condor ritzschk 103 ( ../example_ivanjica/workdir ) > plagiarismSearch S05 Quader.java
-----
|   Assignments in course OOP with Java   |
|         at Humboldt-University Berlin   |
|                                     |
|           plagiarismSearch v4.0         |
|                                     |
-----

call:
./plagiarismSearch <assignmentnumber> <file1> [file2 [file3 ...]] [--disable-compareResult]
settings
  workdirectory: /vol/home-vol2/compssoft/ritzschk/example_ivanjica/workdir
  Input-Encoding   :
  System-Encoding  :

assignment: S05
file(s): Quader.java
compare results?:
ja
```


5 examples of student solutions (assignment "Quader.java")

Number	contents
05842	filename incorrect
18456	o.k.
28506	1 calculation incorrect
30785	1 method without body
40881	o.k., but plagiarism

- Download from goya: [result_assignments5.zip](#)

Revise: Build the resultFiles Exxxx...

```
control solution Nr. 05482 ...
* Wed Aug 11 13:52:02 CEST 2010

-----
automatic control
-compile the code
  ### error: missing the file "Quader.java"
-----
-test the code
-----
* content of the directory:
.          TestQuader2.java  solution.out
..         quader.java      test.out

-----
control solution Nr. 18456 ...
* Wed Aug 11 13:52:03 CEST 2010

-----
automatic control
-compile the code
  o.k. ("Quader.java")
-----
-test the code
-----
* content of the directory:
.          Quader.class      TestQuader2.class  solution.out
..         Quader.java      TestQuader2.java   test.out
```

```
control solution Nr. 30785 ...
* Wed Aug 11 13:52:06 CEST 2010

-----
automatic control
-compile the code
  Quader.java:30: ';' expected
                    public void verschiebe(int dx, int dy, int dz)
                                                    ^
1 error
  ### error: compiling of "Quader.java" impossible
-----
-test the code
-----
* content of the directory:
.          Quader.java      solution.out
..         TestQuader2.java test.out
```

ComparisonResults:



```
...
for i in `< numbers`
do
    diff results/E$i ../inputs/$assignment/solution.out
done
```

```
05482
0a1,95
> -----
> Originaltests:
> -----
> X-Koordinate = 1, Y-Koordinate = 1, Z-Koordinate = 1
> Volumen = 24
> Oberflaeche = 52
> Oberflaeche > Volumen? true
> #####
> X-Koordinate = 4, Y-Koordinate = 4, Z-Koordinate = 4
> Volumen = 24
> Oberflaeche = 52
> Oberflaeche > Volumen? true
> #####
> X-Koordinate = 5, Y-Koordinate = 5, Z-Koordinate = 5
> Volumen = 1000
> Oberflaeche = 600
> Oberflaeche > Volumen? false
> #####
```

without result
→ all 95 lines are different

```
18456
28506
6c6
< Oberflaeche = 26
...
> Oberflaeche = 52
11c11
< Oberflaeche = 26
...
> Oberflaeche = 52
16c16
< Oberflaeche = 300
...
> Oberflaeche = 600
29c29
< Oberflaeche = 3
```

line with difference
(surface too small)

PlagiarismSearch:

```
$ diff -i -e -b -w $file1 $file2 |wc -l
```

```
plagiarismSearch...  
* review 05482...  
  difference from 05482 to 18456:  
  Quader.java:  
/  
  results:  
  97  
  difference from 05482 to 28506:  
  Quader.java:  
/  
  results:  
  97  
  difference from 05482 to 30785:  
  Quader.java:  
/  
  results:  
  0  
  difference from 05482 to 40881:  
  Quader.java:  
/  
  results:  
  97
```

```
* review 18456...  
  difference from 18456 to 28506:  
  Quader.java:  
  6  
  results:  
  45  
  difference from 18456 to 30785:  
  Quader.java:  
  7  
  results:  
  1  
  difference from 18456 to 40881:  
  Quader.java:  
  0  
  results:  
  0
```

```
* review 28506...  
  difference from 28506 to 30785:  
  Quader.java:  
  13  
  results:  
  1  
  difference from 28506 to 40881:  
  Quader.java:  
  6  
  results:  
  45  
* review 30785...  
  difference from 30785 to 40881:  
  Quader.java:  
  9  
  results:  
  97
```

no difference:
potentially plagiat

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

Summary evaluation

- **Revise:** Prepares excellent the solutions for evaluation; furthermore it shows invalid classes and methods
- **ComparisionResults:** Helps to find methods with different results
- **PlagiarismSearch:** Gives an indication for plagiarism
- Finally the inspector has **always to look into the source files for evaluation the layout and the annotation of classes and methods**
- Students get: points and email with critical comments

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

Contents

- Programming lab “OOP with Java” (Overview)
- Details: Structure, Requirements
- Example: Sudoku checker and Sudoku solver
- Shell scripts allows partially automatical evaluation
- **Statistic WS2009/2010 and summary**

Statistics WS0910

assignment	1	2	3	4	5	6	7	8	9	10	11
solutions(goya)	94	83	78	63	63	58	51	52	35	40	22
points 75-100%	77	48	69	37	38	31	30	37	27	17	11
points 50-75%	5	13	2	17	10	7	10	5	3	9	4
points 25-50%	0	9	0	2	0	3	2	1	1	4	1
points 0-25%	0	2	2	6	12	17	8	9	4	10	6
average	9,3	16,2	9,5	7,5	7,1	12,4	14,9	15,5	15,7	19,1	20,0
max. points	10	20	10	10	10	20	20	20	20	30	30
average %	93,0	80,8	94,8	75,0	70,8	61,8	74,7	77,3	78,7	63,6	66,7
non-valued	11	11	5	1	3		1				

- 43 students passed the programming lab with success

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

Summary

- Some students have experience in Java programming
 - In addition to the lessons students get many executable program examples
 - So the programming lab contains little projects with interesting backgrounds (games)
 - There are mentored times in the computer rooms to clear up questions and to understand the solutions (students don't get sample solutions)
 - Comments from students: motivating homework tasks
-



Summary

Thank you for your attention!