



readability-survey → CR

18.01.2023, 13:53

Code-Review zur Sauberkeit von Programmiercode

B001

Liebe Teilnehmende,

vielen Dank für Ihr Interesse an dieser Studie. Diese Untersuchung ist Teil meiner Masterarbeit an der TU Chemnitz und befasst sich mit der Sauberkeit von Programmiercode.

Alle Antworten sind **anonymisiert** und werden ausschließlich im Rahmen statistischer Auswertungen verarbeitet. Sämtliche Angaben werden vertraulich behandelt und nicht an Dritte weitergegeben. **Die Daten können nicht auf Ihre Person zurückgeführt werden.** Es ist wichtig, dass Sie ehrlich und intuitiv antworten. Sie können die Teilnahme an der Studie jederzeit und ohne die Angabe von Gründen beenden.

Wenn Sie sich bei einer Frage unsicher sind, wählen Sie einfach immer die Antwort, die am ehesten für Sie zutrifft. Es gibt keine richtigen oder falschen Antworten. Dies gilt auch, wenn Ihnen Aussagen unpassend oder als Wiederholung vorkommen.

Die Beantwortung nimmt circa 20 Minuten in Anspruch.

Weitere Informationen zur Teilnahme und zum Datenschutz folgen auf der nächsten Seite.

Vielen Dank für Ihre Teilnahme und die Unterstützung meiner Forschung.

Maurice Eichenseer

Dies ist eine anonyme Umfrage.

B003

Teilnahmeinformation

Diese Untersuchung ist Teil meiner Masterarbeit an der TU Chemnitz und befasst sich mit der Erhebung von Aspekten der Sauberkeit von Programmiercode mithilfe eines Checklist-basierten Code-Reviews. Sie werden einige Checklisten erhalten mit Fragen zur Sauberkeit von Programmiercodefragmenten in der Programmiersprache C++, die Sie im Anschluss nach dem Studium des Codes beantworten werden. Darüber hinaus werden einige demografische Angaben abgefragt. Die Beantwortung nimmt circa 20 Minuten in Anspruch, bitte stellen Sie sicher, dass Sie die Umfrage an einem ruhigen Ort und ohne Unterbrechungen durchführen können.

Bitte beantworten Sie alle Fragen so ehrlich und intuitiv wie möglich. Es gibt keine „richtigen“ oder „falschen“ Antworten. Die im Rahmen der Studie erhobenen Daten werden **anonym** erhoben und gespeichert, sowie vertraulich behandelt. Die Daten lassen **keine Rückschlüsse auf Ihre Person zu**. Verschriftlichungen der Ergebnisse der Studie werden dementsprechend ebenfalls in anonymisierter Form erfolgen.

Sie können die Teilnahme an der Studie jederzeit und ohne die Angabe von Gründen beenden.

Sie können an dieser Studie nur teilnehmen, wenn Sie **mindestens 18 Jahre alt** sind und deutsch auf Muttersprachniveau beherrschen.

Im Folgenden finden Sie die detaillierten Teilnahmeinformationen sowie Informationen zum Datenschutz:

Vielen Dank für Ihr Interesse an unserer wissenschaftlichen Untersuchung («Studie»). Im Folgenden informieren wir Sie über die Studie, damit Sie frei entscheiden können, ob Sie daran teilnehmen möchten.

a) Projektbeteiligte

Das Forschungsprojekt wird geleitet von:

Maurice Eichenseer

Masterstudent der Informatik für Geistes- und Sozialwissenschaften

E-Mail: [REDACTED]

Neben der o.g. Projektleitung sind folgende Personen an der Durchführung der Studie beteiligt:

[REDACTED] (Erstbetreuer/Leiter des Rechen- und Informationszentrums der Fakultät für Informatik (FRIZ))
E-Mail [REDACTED]

[REDACTED] (Zweitbetreuer/Wissenschaftlicher Mitarbeiter der Professur Medieninformatik)
E-Mail [REDACTED]

Bei Fragen oder Schwierigkeiten können Sie jederzeit - auch im Nachgang der Studie - Kontakt mit den Projektbeteiligten aufnehmen.

b) Hintergrund und Zielsetzung der Studie

Liebe Studieninteressierte,

vielen Dank, dass Sie sich dazu bereit erklärt haben, an dieser Erhebung teilzunehmen. Die Studie wird im Rahmen einer Masterarbeit zum Thema Clean Code an der Professur für Medieninformatik und dem Rechen- und Informationszentrum der Fakultät für Informatik der TU Chemnitz durchgeführt. Mit Ihrer Teilnahme leisten Sie einen Beitrag zur informatischen Forschung und unterstützen ein wichtiges Forschungsthema.

Ziel der Studie ist es herauszufinden, inwiefern statische Analyseprogramme einen Beitrag zum saubereren Programmieren leisten können.

Bitte beantworten Sie alle Fragen so, wie es Ihnen am passendsten erscheint. Hierbei gibt es keine „richtigen“ oder „falschen“ Antworten. Eine Nachverfolgung Ihrer Daten ist nicht möglich, da die Daten komplett anonymisiert abgespeichert werden.

Die Bearbeitung der Studie wird ca. 20 Minuten in Anspruch nehmen. Die Teilnahme an der Studie ist freiwillig. Sie haben jederzeit die Möglichkeit die Studie zu beenden ohne, dass für Sie ein Nachteil entstehen könnte.

c) Untersuchungsablauf

Ort der Untersuchung:

Die Untersuchung findet online statt.

Ablauf der Messungen:

Zunächst werden Sie nach Ihrer Einstellung zu verschiedenen Aussagen. Hier sollen Sie ihr Ausmaß an Zustimmung oder Ablehnung angeben. Abschließend werden Sie nach Angaben zu Ihrer Person gefragt. Die Daten werden vollständig anonymisiert erhoben.

Dauer der Untersuchung:

Die Gesamtdauer der Untersuchung beträgt etwa 20 Minuten, davon 15 Minuten reine Messzeit und 5 Minuten Vor- und Nachbereitungszeit sowie Pausenzeiten.

d) Ihre Vor- und Nachteile; Risiken

Vorteile: Mit Ihrer Teilnahme an dieser Studie leisten Sie einen Beitrag zur informationswissenschaftlichen Forschung. Mit Ihrer Hilfe können Aspekte der Sauberkeit von Programmiercode in Zukunft besser verstanden werden. Ein unmittelbarer Vorteil entsteht Ihnen nicht.

Nachteile: Die Untersuchung nimmt etwa 20 Minuten Ihrer Zeit in Anspruch. Mit der Messung sind keine erkennbaren Risiken verbunden.

e) Ein- und Ausschlusskriterien für die Teilnahme

An der Studie dürfen nur volljährige Personen teilnehmen, was durch eine Altersangabe kontrolliert wird. Des Weiteren ist ein sehr gutes Verständnis der deutschen Sprache notwendig.

f) Freiwilligkeit und Rücktrittsrecht

Ihre Teilnahme ist freiwillig. Sie können die Untersuchung jederzeit ohne Angaben von Gründen und ohne, dass ein Nachteil für Sie entsteht, abbrechen. Die Anrechnung der Stunden erfolgt in diesem Fall nicht.

g) Datenschutz

Im Rahmen der Studie mit o.g. Bezeichnung werden Daten von Ihnen erhoben, ausgewertet und gespeichert. Die Projektbeteiligten verpflichten sich zur Einhaltung datenschutzrechtlicher Vorschriften.

Die Erläuterungen zum Datenschutz nach Art. 13 EU-DSGVO finden Sie [hier](#).

Maurice Eichenseer

Masterstudent der Informatik für Geistes- und Sozialwissenschaften

E-Mail [\[REDACTED\]](#)

Hiermit erkläre ich, dass ich über die Studie – insbesondere über ihre Ziele, ihren Ablauf samt Dauer und über die Vor- und Nachteile sowie Risiken, die mit der Teilnahme verbunden sein könnten – vollumfänglich aufgeklärt wurde. Die Teilnahmeinformation habe ich gelesen und verstanden. Ich hatte genügend Zeit, um meine Entscheidung zur Studienteilnahme zu überdenken und frei zu treffen. Mir ist bekannt, dass ich jederzeit und ohne Angabe von Gründen meine Einwilligung zur Teilnahme an der Studie zurückziehen kann, ohne dass mir daraus Nachteile entstehen. Ich nehme an der o.g. Studie freiwillig teil. Über den Umgang mit meinen Daten – insbesondere über die Erhebung, Auswertung, Speicherung und Veröffentlichung sowie die Möglichkeiten zur Löschung meiner Daten – wurde ich vollumfänglich aufgeklärt. Die Informationen habe ich gelesen und verstanden. Alle meine Fragen sind zu meiner Zufriedenheit beantwortet worden. Ich hatte genügend Zeit, um meine Entscheidung zum Umgang mit meinen Daten zu überdenken und frei zu treffen. Mir ist bekannt, dass ich meine Einwilligung bis zum Abschluss der Erhebung meiner Daten ohne Angabe von Gründen mit Wirkung für die Zukunft widerrufen kann, ohne dass mir daraus Nachteile entstehen. Ich erkläre mich ausdrücklich und freiwillig, d.h. frei von Zwang und Druck, damit einverstanden, dass meine personenbezogenen Daten im beschriebenen Umfang und zu den beschriebenen Zwecken verarbeitet werden.

Ja

Was ist Ihr Gender (Geschlecht)?

KV01

männlich

weiblich

divers

Wie alt sind Sie?

KV02

Alter (in Jahren):

Wie haben Sie von der Studie erfahren?

KV03

Universität

Firma

Online-Forum

Sonstige

Wie viele Jahre Erfahrung in Programmierung besitzen Sie circa?

KV04

Programmiererfahrung

(in Jahren):

Besitzen Sie bereits Erfahrung in Objektorientierter Programmierung?

KV05

Ja

Nein

Kenne Sie bereits das Clean Code Konzept von Robert C. Martin (Uncle Bob)?

KV06

ja

nein

Wie viele Jahre Erfahrung besitzen Sie circa in der Programmierung in C++?

KV07

Nun beginnt das Checklisten-basierte Code-Review. Im Folgenden werden Ihnen immer zuerst Checklisten angezeigt, die die Fragen zu den Programmiercodefragmenten enthalten. Anschließend werden Sie die Fragen zu jedem Codebeispiel, das Sie überfliegen werden, beantworten. Die relevanten Teile im Code sind dabei jeweils rot markiert. Sie müssen sich die Checkliste nicht komplett merken, da die Frageitems direkt unter den Codebeispielen angezeigt werden. Darauf folgt dann die nächste Checkliste mit neuen Fragen zu den gleichen Codebeispielen, bis alle Checklisten beantwortet worden sind.

Am Ende gibt es zur Belohnung einige Informatikwitze.

Checkliste: Variablen

- [Namensgebung] Wie aussagekräftig und zweckbeschreibend sind die Variablennamen?
- [Namensgebung] Wie gut aussprechbar sind die Variablennamen?
- [Namensgebung] Wie gut suchbar sind die Variablennamen in einer Entwicklungsumgebung / einem Editor?
- [Namensgebung] Wie konsistent ist die Namensgebung der Variablen?

```

1  class UserValidator {
2    private:
3      Cryptographer cryptographer;
4
5    public:
6      bool checkPassword(string userName, string password);
7    };
8
9  bool UserValidator::checkPassword(string userName, string password) {
10   User *user = UserGateway::findByName(userName);
11   if (user != NULL) {
12     string codedPhrase = user -> getPhraseEncodedByPassword();
13     string phrase = cryptographer.decrypt(codedPhrase, password);
14     if (phrase.compare("Valid Password") == 0) {
15       Session::initialize();
16       return true;
17     }
18   }
19   return false;
20 }
```

VA01

[Namensgebung] Die Variablennamen sind aussagekräftig und zweckbeschreibend.

VA03

Stimme überhaupt nicht zu	Stimme eher nicht zu	teils teils	Stimme eher zu	Stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Namensgebung] Die Variablennamen sind gut aussprechbar.

VA04

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Namensgebung] Die Variablennamen sind in einer Entwicklungsumgebung / einem Editor suchbar.

VA05

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

VA06

[Namensgebung] Die Namensgebung der Variablen ist konsistent.

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

VA07

```
1 class PrintPrimes {
2     public:
3         friend int main(char* argv[]);
4     };
5
6     int main(char* argv[]) {
7         const int M = 1000;
8         const int RR = 50;
9         const int CC = 4;
10        const int WW = 10;
11        const int ORDMAX = 30;
12        int* P = new int[M + 1];
13        int PAGENUMBER;
14        int PAGEOFFSET;
15        int ROWOFFSET;
16        int C;
17        int J;
18        int K;
19        bool JPRIME;
20        int ORD;
21        int SQUARE;
22        int N;
23        int* MULT = new int[ORDMAX + 1];
24
25        J = 1;
26        K = 1;
27        P[1] = 2;
28        ORD = 2;
29        SQUARE = 9;
```

```

30    while (K < M) {
31        do {
32            J = J + 2;
33            if (J == SQUARE) {
34                ORD = ORD + 1;
35                SQUARE = P[ORD] * P[ORD];
36                MULT[ORD - 1] = J;
37            }
38            N = 2;
39            JPRIME = true;
40            while (N < ORD && JPRIME) {
41                while (MULT[N] < J) {
42                    MULT[N] = MULT[N] + P[N] + P[N];
43                    if (MULT[N] == J)
44                        JPRIME = false;
45                }
46                N = N + 1;
47            }
48        } while (!JPRIME);
49        K = K + 1;
50        P[K] = J;
51    }
52    {
53        PAGENUMBER = 1;
54        PAGEOFFSET = 1;
55        while (PAGEOFFSET <= M) {
56            cout << "The First " << M <<
57            " Prime Numbers --- Page " << PAGENUMBER;
58            cout << "";
59            for (ROWOFFSET = PAGEOFFSET; ROWOFFSET < PAGEOFFSET + RR; ROWOFFSET++) {
60                for (C = 0; C < CC; C++) {
61                    if (ROWOFFSET + C * RR <= M) {
62                        cout << setw(2) << P[ROWOFFSET + C * RR];
63                        cout << " ";
64                    }
65                }
66                PAGENUMBER = PAGENUMBER + 1;
67                PAGEOFFSET = PAGEOFFSET + RR * CC;
68            }
69        }
70    }
71    return 0;
72 }
```

[Namensgebung] Die Variablennamen sind aussagekräftig und zweckbeschreibend.

VIA08 □

Stimme überhaupt	Stimme eher nicht zu	teils	teils	Stimme eher zu	Stimme voll und ganz
	nicht zu				zu

[Namensgebung] Die Variablennamen sind gut aussprechbar.

VA09

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

[Namensgebung] Die Variablennamen sind in einer Entwicklungsumgebung / einem Editor suchbar.

VA10

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

[Namensgebung] Die Namensgebung der Variablen ist konsistent.

VA11

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

VA12

```
1  class CodeAnalyzer {
2      private:
3          int lineCount;
4          int maxWidth;
5          int widestLineNumber;
6          LineWidthHistogram lineWidthHistogram;
7          int totalChars;
8
9          static void findJavaFiles(File& parentDirectory, list<File>& files);
10         void measureLine(string line);
11         void recordWidestLine(int lineSize);
12         int lineCountForWidth(int width);
13         vector<Integer> getSortedWidths();
14     public:
15         CodeAnalyzer();
16         ~CodeAnalyzer();
17         static list<File> findJavaFiles(File& parentDirectory);
18         void analyzeFile(File& javaFile) throw();
19         int getLineCount();
20         int getMaxLineWidth();
21         int getWidestLineNumber();
22         LineWidthHistogram getLineWidthHistogram();
23         double getMeanLineWidth();
24         int getMedianLineWidth();
25     };
26
27     CodeAnalyzer::CodeAnalyzer() {
28         LineWidthHistogram lineWidthHistogram = LineWidthHistogram();
29     }
30 }
```

```
31  CodeAnalyzer::~CodeAnalyzer() {
32  [ delete lineWidthHistogram;
33  ]
34
35  list<File> CodeAnalyzer::findJavaFiles(File& parentDirectory) {
36  [ list<File> files;
37  findJavaFiles(parentDirectory, files);
38  return files;
39  ]
40
41  void CodeAnalyzer::findJavaFiles(File& parentDirectory, list<File>& files) {
42  for (File file : parentDirectory.listFiles()) {
43  if (file.getName().endsWith(".java"))
44  files.insert(files.begin(), file);
45  else if (file.isDirectory())
46  findJavaFiles(file, files);
47  }
48  }
49
50  void CodeAnalyzer::analyzeFile(File& javaFile) throw() {
51  FileReader fr = FileReader(javaFile);
52  BufferedReader br = BufferedReader(fr);
53  string line;
54  while (!(line = br.readLine()).empty()) {
55  measureLine(line);
56  }
57  }
58 }
```

```
59 void CodeAnalyzer::measureLine(string line) {
60     lineCount++;
61     int lineSize = line.length();
62     totalChars += lineSize;
63     lineWidthHistogram.addLine(lineSize, lineCount);
64     recordWidestLine(lineSize);
65 }
66
67 void CodeAnalyzer::recordWidestLine(int lineSize) {
68     if (lineSize > maxLineWidth) {
69         maxLineWidth = lineSize;
70         widestLineNumber = lineCount;
71     }
72 }
73
74 int CodeAnalyzer::getLineCount() {
75     return lineCount;
76 }
77
78 int CodeAnalyzer::getMaxLineWidth() {
79     return maxLineWidth;
80 }
81
82 int CodeAnalyzer::getWidestLineNumber() {
83     return widestLineNumber;
84 }
85
86 LineWidthHistogram CodeAnalyzer::getLineWidthHistogram() {
87     return lineWidthHistogram;
88 }
```

```

90  double CodeAnalyzer::getMeanLineWidth() {
91    return (double)totalChars/lineCount;
92  }
93
94  int CodeAnalyzer::getMedianLineWidth() {
95    vector<Integer> sortedWidths = getSortedWidths();
96    int cumulativeLineCount = 0;
97    for(int width : sortedWidths) {
98      cumulativeLineCount += lineCountForWidth(width);
99      if (cumulativeLineCount > lineCount/2)
100        return width;
101    }
102    throw Error("Cannot get here");
103  }
104
105  int CodeAnalyzer::lineCountForWidth(int width) {
106    return lineWidthHistogram.getLinesforWidth(width).size();
107  }
108
109  vector<Integer> CodeAnalyzer::getSortedWidths() {
110    vector<Integer> widths = lineWidthHistogram.getWidths();
111    vector<Integer> sortedWidths = widths;
112    sort(sortedWidths, sortedWidths + widths.size());
113    return sortedWidths;
114  }

```

[Namensgebung] Die Variablennamen sind aussagekräftig und zweckbeschreibend.

VA13

Stimme überhaupt nicht zu	Stimme eher nicht zu	teils teils	Stimme eher zu	Stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Namensgebung] Die Variablennamen sind gut aussprechbar.

VA14

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Namensgebung] Die Variablennamen sind in einer Entwicklungsumgebung / einem Editor suchbar.

VA15

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Namensgebung] Die Namensgebung der Variablen ist konsistent.

stimme überhaupt nicht zu
stimme eher nicht zu
teils teils
stimme eher zu
stimme voll und ganz zu

VA17

```
1 static string testableHtml(
2     PageData pageData,
3     bool includeSuiteSetup
4 ) {
5     WikiPage *wikiPage = pageData.getWikiPage();
6     StringBuffer *buffer = new StringBuffer();
7     if (pageData.hasAttribute("Test")) {
8         if (includeSuiteSetup) {
9             WikiPage *suiteSetup =
10                PageCrawlerImpl::getInheritedPage(
11                    SuiteResponder::SUITE_SETUP_NAME, *wikiPage
12                );
13             if (suiteSetup != NULL) {
14                 WikiPagePath *pagePath =
15                     suiteSetup.getPageCrawler().getFullPath(suiteSetup);
16                 string pagePathName = PathParser.render(pagePath);
17                 buffer -> append("!include -setup .")
18                     -> append(pagePathName)
19                     -> append("\n");
20             }
21         }
22         WikiPage *setup =
23             PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
24         if (setup != NULL) {
25             WikiPagePath *setupPath =
26                 wikiPage.getPageCrawler().getFullPath(setup);
27             string setupPathName = PathParser.render(setupPath);
28             buffer -> append("!include -setup .")
29                 -> append(setupPathName)
30                 -> append("\n");
31         }
32     }
```

```

33     buffer.append(pageData.getContent());
34     if (pageData.hasAttribute("Test")) {
35         WikiPage *teardown =
36             PageCrawlerImpl.getInheritedPage("TearDown", wikiPage);
37         if (teardown != NULL) {
38             WikipagePath tearDownPath =
39                 wikiPage.getPageCrawler().getFullPath(teardown);
40             string tearDownPathName = PathParser.render(tearDownPath);
41             buffer -> append("\n")
42                 -> append("!include -teardown .")
43                 -> append(tearDownPathName)
44                 -> append("\n");
45     }
46     if (includeSuiteSetup) {
47         WikiPage *suiteTeardown =
48             PageCrawlerImpl.getInheritedPage(
49                 SuiteResponder.SUITE_TEARDOWN_NAME,
50                 wikiPage
51             );
52         if (suiteTeardown != NULL) {
53             WikipagePath *pagePath =
54                 suiteTeardown.getPageCrawler().getFullPath(suiteTeardown);
55             buffer -> append("!include -teardown .")
56                 -> append(pagePathName)
57                 -> append("\n");
58         }
59     }
60 }
61 }
62 pageData.setContent(buffer.toString());
63 return pageData.getHtml();
64 }
```

[Namensgebung] Die Variablennamen sind aussagekräftig und zweckbeschreibend.

VA18

Stimme überhaupt nicht zu	Stimme eher nicht zu	teils teils	Stimme eher zu	Stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Namensgebung] Die Variablennamen sind gut aussprechbar.

VA19

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

VA20

[Namensgebung] Die Variablennamen sind in einer Entwicklungsumgebung / einem Editor gut suchbar.

stimme überhaupt nicht zu
stimme eher nicht zu
teils teils
stimme eher zu
stimme voll und ganz zu

[Namensgebung] Die Namensgebung der Variablen ist konsistent.

VA21

stimme überhaupt nicht zu
stimme eher nicht zu
teils teils
stimme eher zu
stimme voll und ganz zu

```

1 class BoldWidget : public ParentWidget {
2     public:
3         static const string REGEXP;
4         static const Pattern& pattern;
5         ParentWidget* parent;
6         BoldWidget(ParentWidget& parent, string text);
7         string render();
8     };
9     const Pattern& BoldWidget::pattern = Pattern::compile("'''(.+?)'''",
10             Pattern::MULTILINE + Pattern::DOTALL
11     );
12     const string BoldWidget::REGEXP = "'''.+?'''";
13     BoldWidget::BoldWidget(ParentWidget& parent, string text) {
14         Matcher *match = pattern.matcher(text);
15         match -> find();
16         addChildWidgets(match -> group(1));
17     }
18     string BoldWidget::render() {
19         StringBuffer *html = new StringBuffer("<br>");
20         html -> append(childHtml()).append("<br>");
21         return html -> toString();
22     }

```

VA22

[Namensgebung] Die Variablennamen sind aussagekräftig und zweckbeschreibend.

VA23

Stimme überhaupt	Stimme eher nicht zu	teils teils	Stimme eher zu	Stimme voll und ganz zu

[Namensgebung] Die Variablennamen sind gut aussprechbar.

VA24

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu

[Namensgebung] Die Variablennamen sind in einer Entwicklungsumgebung / einem Editor suchbar.

VA25

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu

[Namensgebung] Die Namensgebung der Variablen ist konsistent.

stimme überhaupt nicht zu
stimme eher nicht zu
teils teils
stimme eher zu
stimme voll und ganz zu

Seite 11

CM

ME01

Checkliste: Funktionen / Methoden

- [Namensgebung] Wie aussagekräftig und zweckbeschreibend sind die Methodennamen?
- [Namensgebung] Wie konsistent ist die Namensgebung der Methoden?
- [Namensgebung] Wie gut aussprechbar sind die Methodennamen?
- [Formatierung] Wie konsistent sind die Einrückungen in den Methoden?
- [Klarheit] Wie eindeutig sind die Funktionalitäten der Methoden?
- [Komplexität] Besitzt die Methode unerwartete Effekte abseits der im Methodennamen beschriebenen Funktionalitäten?
- [Komplexität] Wie bewerten Sie die Komplexität der Methoden?
- [Komplexität] Wie bewerten Sie die Komplexität der Blöcke (if/else, ..)?

```

1  class UserValidator {
2    private:
3      Cryptographer cryptographer;
4
5    public:
6      bool checkPassword(string userName, string password);
7    };
8
9  bool UserValidator::checkPassword(string userName, string password) {
10   User *user = UserGateway::findByName(userName);
11   if (user != NULL) {
12     string codedPhrase = user -> getPhraseEncodedByPassword();
13     string phrase = cryptographer.decrypt(codedPhrase, password);
14     if (phrase.compare("Valid Password") == 0) {
15       Session::initialize();
16       return true;
17     }
18   }
19   return false;
20 }
```

ME02

[Namensgebung] Die Methodennamen sind aussagekräftig und zweckbeschreibend.

ME03

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Namensgebung] Die Namensgebung der Methoden ist konsistent.

ME04

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Namensgebung] Die Methodennamen sind gut aussprechbar.

ME05

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

ME06

[Formatierung] Die Einrückungen in den Methoden sind konsistent.

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME07**[Klarheit] Die Funktionalitäten der Methoden sind eindeutig.**

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME08**[Komplexität] Die Methode besitzt keine unerwarteten Effekte abseits der im Methodennamen beschriebenen Funktionalitäten.**

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME09**[Komplexität] Die Komplexität der Methoden ist passend.**

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME10**[Komplexität] Die Komplexität der Blöcke (if/else, ..) ist passend.**

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME11

```
1 class PrintPrimes {
2     public:
3         friend int main(char* argv[]);
4     };
5
6     int main(char* argv[]) {
7         const int M = 1000;
8         const int RR = 50;
9         const int CC = 4;
10        const int WW = 10;
11        const int ORDMAX = 30;
12        int* P = new int[M + 1];
13        int PAGENUMBER;
14        int PAGEOFFSET;
15        int ROWOFFSET;
16        int C;
17        int J;
18        int K;
19        bool JPRIME;
20        int ORD;
21        int SQUARE;
22        int N;
23        int* MULT = new int[ORDMAX + 1];
24
25        J = 1;
26        K = 1;
27        P[1] = 2;
28        ORD = 2;
29        SQUARE = 9;
```

```

30    while (K < M) {
31        do {
32            J = J + 2;
33            if (J == SQUARE) {
34                ORD = ORD + 1;
35                SQUARE = P[ORD] * P[ORD];
36                MULT[ORD - 1] = J;
37            }
38            N = 2;
39            JPRIME = true;
40            while (N < ORD && JPRIME) {
41                while (MULT[N] < J) {
42                    MULT[N] = MULT[N] + P[N] + P[N];
43                    if (MULT[N] == J)
44                        JPRIME = false;
45                }
46                N = N + 1;
47            }
48        } while (!JPRIME);
49        K = K + 1;
50        P[K] = J;
51    }
52    {
53        PAGENUMBER = 1;
54        PAGEOFFSET = 1;
55        while (PAGEOFFSET <= M) {
56            cout << "The First " << M <<
57            " Prime Numbers --- Page " << PAGENUMBER;
58            cout << "";
59            for (ROWOFFSET = PAGEOFFSET; ROWOFFSET < PAGEOFFSET + RR; ROWOFFSET++) {
60                for (C = 0; C < CC; C++) {
61                    if (ROWOFFSET + C * RR <= M) {
62                        cout << setw(2) << P[ROWOFFSET + C * RR];
63                        cout << " ";
64                    }
65                }
66                PAGENUMBER = PAGENUMBER + 1;
67                PAGEOFFSET = PAGEOFFSET + RR * CC;
68            }
69        }
70    }
71    return 0;
72 }
```

[Namensgebung] Die Methodennamen sind aussagekräftig und zweckbeschreibend.

ME12

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

[Namensgebung] Die Namensgebung der Methoden ist konsistent.

ME13

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Namensgebung] Die Methodennamen sind gut aussprechbar.

ME14

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Formatierung] Die Einrückungen in den Methoden sind konsistent.

ME15

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Klarheit] Die Funktionalitäten der Methoden sind eindeutig.

ME16

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Komplexität] Die Methode besitzt keine unerwarteten Effekte abseits der im Methodennamen beschriebenen Funktionalitäten.

ME17

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Komplexität] Die Komplexität der Methoden ist passend.

ME18

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Komplexität] Die Komplexität der Blöcke (if/else, ..) ist passend.

ME19

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

ME20

```
1  class CodeAnalyzer {
2      private:
3          int lineCount;
4          int maxWidth;
5          int widestLineNumber;
6          LineWidthHistogram lineWidthHistogram;
7          int totalChars;
8
9          static void findJavaFiles(File& parentDirectory, list<File>& files);
10         void measureLine(string line);
11         void recordWidestLine(int lineSize);
12         int lineCountForWidth(int width);
13         vector<Integer> getSortedWidths();
14     public:
15         CodeAnalyzer();
16         ~CodeAnalyzer();
17         static list<File> findJavaFiles(File& parentDirectory);
18         void analyzeFile(File& javaFile) throw();
19         int getLineCount();
20         int getMaxLineWidth();
21         int getWidestLineNumber();
22         LineWidthHistogram getLineWidthHistogram();
23         double getMeanLineWidth();
24         int getMedianLineWidth();
25     };
26
27     CodeAnalyzer::CodeAnalyzer() {
28         LineWidthHistogram lineWidthHistogram = LineWidthHistogram();
29     }
30 }
```

```
31  CodeAnalyzer::~CodeAnalyzer() {
32  [ delete lineWidthHistogram;
33  ]
34
35  list<File> CodeAnalyzer::findJavaFiles(File& parentDirectory) {
36  list<File> files;
37  findJavaFiles(parentDirectory, files);
38  return files;
39  }
40
41  void CodeAnalyzer::findJavaFiles(File& parentDirectory, list<File>& files) {
42  for (File file : parentDirectory.listFiles()) {
43  if (file.getName().endsWith(".java"))
44  files.insert(files.begin(), file);
45  else if (file.isDirectory())
46  findJavaFiles(file, files);
47  }
48  }
49
50  void CodeAnalyzer::analyzeFile(File& javaFile) throw() {
51  FileReader fr = FileReader(javaFile);
52  BufferedReader br = BufferedReader(fr);
53  string line;
54  while (!(line = br.readLine()).empty()) {
55  measureLine(line);
56  }
57  }
58 }
```

```
59 void CodeAnalyzer::measureLine(string line) {
60     lineCount++;
61     int lineSize = line.length();
62     totalChars += lineSize;
63     lineWidthHistogram.addLine(lineSize, lineCount);
64     recordWidestLine(lineSize);
65 }
66
67 void CodeAnalyzer::recordWidestLine(int lineSize) {
68     if (lineSize > maxWidthWidth) {
69         maxWidthWidth = lineSize;
70         widestLineNumber = lineCount;
71     }
72 }
73
74 int CodeAnalyzer::getLineCount() {
75     return lineCount;
76 }
77
78 int CodeAnalyzer::getMaxLineWidth() {
79     return maxWidthWidth;
80 }
81
82 int CodeAnalyzer::getWidestLineNumber() {
83     return widestLineNumber;
84 }
85
86 LineWidthHistogram CodeAnalyzer::getLineWidthHistogram() {
87     return lineWidthHistogram;
88 }
89
```

```

90  double CodeAnalyzer::getMeanLineWidth() {
91  |   return (double)totalChars/lineCount;
92  }
93
94  int CodeAnalyzer::getMedianLineWidth() {
95  |   vector<Integer> sortedWidths = getSortedWidths();
96  |   int cumulativeLineCount = 0;
97  |   for(int width : sortedWidths) {
98  |       cumulativeLineCount += lineCountForWidth(width);
99  |       if (cumulativeLineCount > lineCount/2)
100 |           return width;
101  }
102  throw Error("Cannot get here");
103 }
104
105 int CodeAnalyzer::lineCountForWidth(int width) {
106 |   return lineWidthHistogram.getLinesforWidth(width).size();
107 }
108
109 vector<Integer> CodeAnalyzer::getSortedWidths() {
110 |   vector<Integer> widths = lineWidthHistogram.getWidths();
111 |   vector<Integer> sortedWidths = widths;
112 |   sort(sortedWidths, sortedWidths + widths.size());
113 |   return sortedWidths;
114 }
```

[Namensgebung] Die Methodennamen sind aussagekräftig und zweckbeschreibend.

ME21

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
------------------------------	----------------------	-------------	----------------	----------------------------

[Namensgebung] Die Namensgebung der Methoden ist konsistent.

ME22

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
------------------------------	----------------------	-------------	----------------	----------------------------

[Namensgebung] Die Methodennamen sind gut aussprechbar.

ME23

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
------------------------------	----------------------	-------------	----------------	----------------------------

ME24

[Formatierung] Die Einrückungen in den Methoden sind konsistent.

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME25**[Klarheit] Die Funktionalitäten der Methoden sind eindeutig.**

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME26**[Komplexität] Die Methode besitzt keine unerwarteten Effekte abseits der im Methodennamen beschriebenen Funktionalitäten.**

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME27**[Komplexität] Die Komplexität der Methoden ist passend.**

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME28**[Komplexität] Die Komplexität der Blöcke (if/else, ..) ist passend.**

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME29

```
1 static string testableHtml(
2     PageData pageData,
3     bool includeSuiteSetup
4 ) {
5     WikiPage *wikiPage = pageData.getWikiPage();
6     StringBuffer *buffer = new StringBuffer();
7     if (pageData.hasAttribute("Test")) {
8         if (includeSuiteSetup) {
9             WikiPage *suiteSetup =
10                 PageCrawlerImpl::getInheritedPage(
11                     SuiteResponder::SUITE_SETUP_NAME, *wikiPage
12                 );
13             if (suiteSetup != NULL) {
14                 WikiPagePath *pagePath =
15                     suiteSetup.getPageCrawler().getFullPath(suiteSetup);
16                 string pagePathName = PathParser.render(pagePath);
17                 buffer -> append("!include -setup .")
18                     -> append(pagePathName)
19                     -> append("\n");
20             }
21         }
22         WikiPage *setup =
23             PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
24         if (setup != NULL) {
25             WikiPagePath *setupPath =
26                 wikiPage.getPageCrawler().getFullPath(setup);
27             string setupPathName = PathParser.render(setupPath);
28             buffer -> append("!include -setup .")
29                 -> append(setupPathName)
30                 -> append("\n");
31         }
32     }
```

```

33     buffer.append(pageData.getContent());
34     if (pageData.hasAttribute("Test")) {
35         WikiPage *teardown =
36             PageCrawlerImpl.getInheritedPage("TearDown", wikiPage);
37         if (teardown != NULL) {
38             WikipagePath tearDownPath =
39                 wikiPage.getPageCrawler().getFullPath(teardown);
40             string tearDownPathName = PathParser.render(tearDownPath);
41             buffer -> append("\n")
42                 -> append("!include -teardown .")
43                 -> append(tearDownPathName)
44                 -> append("\n");
45     }
46     if (includeSuiteSetup) {
47         WikiPage *suiteTeardown =
48             PageCrawlerImpl.getInheritedPage(
49                 SuiteResponder.SUITE_TEARDOWN_NAME,
50                 wikiPage
51             );
52         if (suiteTeardown != NULL) {
53             WikiPagePath *pagePath =
54                 suiteTeardown.getPageCrawler().getFullPath(suiteTeardown);
55             buffer -> append("!include -teardown .")
56                 -> append(pagePathName)
57                 -> append("\n");
58         }
59     }
60 }
61 }
62 pageData.setContent(buffer.toString());
63 return pageData.getHtml();
64 }
```

[Namensgebung] Die Methodennamen sind aussagekräftig und zweckbeschreibend.

ME30

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

[Namensgebung] Die Namensgebung der Methoden ist konsistent.

ME31

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

ME32

[Namensgebung] Die Methodennamen sind gut aussprechbar.

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME33

[Formatierung] Die Einrückungen in den Methoden sind konsistent.

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME34

[Klarheit] Die Funktionalitäten der Methoden sind eindeutig.

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME35

[Komplexität] Die Methode besitzt keine unerwarteten Effekte abseits der im Methodennamen beschriebenen Funktionalitäten.

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME36

[Komplexität] Die Komplexität der Methoden ist passend.

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

ME37

[Komplexität] Die Komplexität der Blöcke (if/else, ..) ist passend.

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

```

1 class BoldWidget : public ParentWidget {
2     public:
3         static const string REGEXP;
4         static const Pattern& pattern;
5         ParentWidget* parent;
6         BoldWidget(ParentWidget& parent, string text);
7         string render();
8     };
9     const Pattern& BoldWidget::pattern = Pattern::compile("'''(.+?)'''",
10     | | | Pattern::MULTILINE + Pattern::DOTALL
11 );
12     const string BoldWidget::REGEXP = "'''.+?'''";
13     BoldWidget::BoldWidget(ParentWidget& parent, string text) {
14         Matcher *match = pattern.matcher(text);
15         match -> find();
16         addChildWidgets(match -> group(1));
17     }
18     string BoldWidget::render() {
19         StringBuffer *html = new StringBuffer("<br>");
20         html -> append(childHtml()).append("<br>");
21         return html -> toString();
22     }

```

ME38

[Namensgebung] Die Methodennamen sind aussagekräftig und zweckbeschreibend.

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

ME39

[Namensgebung] Die Namensgebung der Methoden ist konsistent.

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

ME40

[Namensgebung] Die Methodennamen sind gut aussprechbar.

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

ME41

[Formatierung] Die Einrückungen in den Methoden sind konsistent.

ME42

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

[Klarheit] Die Funktionalitäten der Methoden sind eindeutig.

ME43

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

[Komplexität] Die Methode besitzt keine unerwarteten Effekte abseits der im Methodennamen beschriebenen Funktionalitäten.

ME44

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

[Komplexität] Die Komplexität der Methoden ist passend.

ME45

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

[Komplexität] Die Komplexität der Blöcke (if/else, ..) ist passend.

ME46

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

KL01

Checkliste: Klassen

- [Namensgebung] Wie aussagekräftig und zweckbeschreibend ist der Klassename?
- [Komplexität] Wie bewerten Sie die Komplexität der Klasse?
- [Klarheit] Wie eindeutig sind die Funktionalitäten der Klasse?

```

1  class UserValidator {
2    private:
3      Cryptographer cryptographer;
4
5    public:
6      bool checkPassword(string userName, string password);
7    };
8
9  bool UserValidator::checkPassword(string userName, string password) {
10   User *user = UserGateway::findByName(userName);
11   if (user != NULL) {
12     string codedPhrase = user -> getPhraseEncodedByPassword();
13     string phrase = cryptographer.decrypt(codedPhrase, password);
14     if (phrase.compare("Valid Password") == 0) {
15       Session::initialize();
16       return true;
17     }
18   }
19   return false;
20 }
```

KL02

[Namensgebung] Der Klassenname ist aussagekräftig und zweckbeschreibend.

KL03

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Komplexität] Die Komplexität der Klasse ist passend.

KL04

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Klarheit] Die Funktionalitäten der Klasse sind eindeutig.

KL05

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

KL06

```
1 class PrintPrimes {
2     public:
3         friend int main(char* argv[]);
4     };
5
6     int main(char* argv[]) {
7         const int M = 1000;
8         const int RR = 50;
9         const int CC = 4;
10        const int WW = 10;
11        const int ORDMAX = 30;
12        int* P = new int[M + 1];
13        int PAGENUMBER;
14        int PAGEOFFSET;
15        int ROWOFFSET;
16        int C;
17        int J;
18        int K;
19        bool JPRIME;
20        int ORD;
21        int SQUARE;
22        int N;
23        int* MULT = new int[ORDMAX + 1];
24
25        J = 1;
26        K = 1;
27        P[1] = 2;
28        ORD = 2;
29        SQUARE = 9;
```

```

30     while (K < M) {
31         do {
32             J = J + 2;
33             if (J == SQUARE) {
34                 ORD = ORD + 1;
35                 SQUARE = P[ORD] * P[ORD];
36                 MULT[ORD - 1] = J;
37             }
38             N = 2;
39             JPRIME = true;
40             while (N < ORD && JPRIME) {
41                 while (MULT[N] < J) {
42                     MULT[N] = MULT[N] + P[N] + P[N];
43                     if (MULT[N] == J)
44                         JPRIME = false;
45                 }
46                 N = N + 1;
47             }
48         } while (!JPRIME);
49         K = K + 1;
50         P[K] = J;
51     }
52 {
53     PAGENUMBER = 1;
54     PAGEOFFSET = 1;
55     while (PAGEOFFSET <= M) {
56         cout << "The First " << M <<
57             " Prime Numbers --- Page " << PAGENUMBER;
58         cout << "";
59         for (ROWOFFSET = PAGEOFFSET; ROWOFFSET < PAGEOFFSET + RR; ROWOFFSET++) {
60             for (C = 0; C < CC; C++) {
61                 if (ROWOFFSET + C * RR <= M) {
62                     cout << setw(2) << P[ROWOFFSET + C * RR];
63                     cout << " ";
64                 }
65             }
66             PAGENUMBER = PAGENUMBER + 1;
67             PAGEOFFSET = PAGEOFFSET + RR * CC;
68         }
69     }
70 }
71 return 0;
72 }
```

[Namensgebung] Der Klassenname ist aussagekräftig und zweckbeschreibend.

KL07

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
				zu
nicht zu				

[Komplexität] Die Komplexität der Klasse ist passend.**KL08** 

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

[Klarheit] Die Funktionalitäten der Klasse sind eindeutig.**KL09** 

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz zu

KL10

```
1 class CodeAnalyzer {
2     private:
3         int lineCount;
4         int maxWidth;
5         int widestLineNumber;
6         LineWidthHistogram lineWidthHistogram;
7         int totalChars;
8
9         static void findJavaFiles(File& parentDirectory, list<File>& files);
10        void measureLine(string line);
11        void recordWidestLine(int lineSize);
12        int lineCountForWidth(int width);
13        vector<Integer> getSortedWidths();
14    public:
15        CodeAnalyzer();
16        ~CodeAnalyzer();
17        static list<File> findJavaFiles(File& parentDirectory);
18        void analyzeFile(File& javaFile) throw();
19        int getLineCount();
20        int getMaxLineWidth();
21        int getWidestLineNumber();
22        LineWidthHistogram getLineWidthHistogram();
23        double getMeanLineWidth();
24        int getMedianLineWidth();
25    };
26
27 CodeAnalyzer::CodeAnalyzer() {
28     lineWidthHistogram = LineWidthHistogram();
29 }
```

```
31  CodeAnalyzer::~CodeAnalyzer() {
32  [ delete lineWidthHistogram;
33  ]
34
35  list<File> CodeAnalyzer::findJavaFiles(File& parentDirectory) {
36  list<File> files;
37  findJavaFiles(parentDirectory, files);
38  return files;
39  }
40
41  void CodeAnalyzer::findJavaFiles(File& parentDirectory, list<File>& files) {
42  for (File file : parentDirectory.listFiles()) {
43  if (file.getName().endsWith(".java"))
44  files.insert(files.begin(), file);
45  else if (file.isDirectory())
46  findJavaFiles(file, files);
47  }
48  }
49
50  void CodeAnalyzer::analyzeFile(File& javaFile) throw() {
51  FileReader fr = FileReader(javaFile);
52  BufferedReader br = BufferedReader(fr);
53  string line;
54  while (!(line = br.readLine()).empty()) {
55  measureLine(line);
56  }
57  }
58 }
```

```
59 void CodeAnalyzer::measureLine(string line) {
60     lineCount++;
61     int lineSize = line.length();
62     totalChars += lineSize;
63     lineWidthHistogram.addLine(lineSize, lineCount);
64     recordWidestLine(lineSize);
65 }
66
67 void CodeAnalyzer::recordWidestLine(int lineSize) {
68     if (lineSize > maxWidthWidth) {
69         maxWidthWidth = lineSize;
70         widestLineNumber = lineCount;
71     }
72 }
73
74 int CodeAnalyzer::getLineCount() {
75     return lineCount;
76 }
77
78 int CodeAnalyzer::getMaxLineWidth() {
79     return maxWidthWidth;
80 }
81
82 int CodeAnalyzer::getWidestLineNumber() {
83     return widestLineNumber;
84 }
85
86 LineWidthHistogram CodeAnalyzer::getLineWidthHistogram() {
87     return lineWidthHistogram;
88 }
89 }
```

```

90  double CodeAnalyzer::getMeanLineWidth() {
91  |   return (double)totalChars/lineCount;
92  }
93
94  int CodeAnalyzer::getMedianLineWidth() {
95  |   vector<Integer> sortedWidths = getSortedWidths();
96  |   int cumulativeLineCount = 0;
97  |   for(int width : sortedWidths) {
98  |       cumulativeLineCount += lineCountForWidth(width);
99  |       if (cumulativeLineCount > lineCount/2)
100 |           return width;
101  }
102  throw Error("Cannot get here");
103 }
104
105 int CodeAnalyzer::lineCountForWidth(int width) {
106 |   return lineWidthHistogram.getLinesforWidth(width).size();
107 }
108
109 vector<Integer> CodeAnalyzer::getSortedWidths() {
110 |   vector<Integer> widths = lineWidthHistogram.getWidths();
111 |   vector<Integer> sortedWidths = widths;
112 |   sort(sortedWidths, sortedWidths + widths.size());
113 |   return sortedWidths;
114 }
```

[Namensgebung] Der Klassename ist aussagekräftig und zweckbeschreibend.

KL11

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Komplexität] Die Komplexität der Klasse ist passend.

KL12

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

[Klarheit] Die Funktionalitäten der Klasse sind eindeutig.

KL13

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

```

1 class BoldWidget : public ParentWidget {
2     public:
3         static const string REGEXP;
4         static const Pattern& pattern;
5         ParentWidget* parent;
6         BoldWidget(ParentWidget& parent, string text);
7         string render();
8     };
9     const Pattern& BoldWidget::pattern = Pattern::compile("'''(.+?)'''",
10             Pattern::MULTILINE + Pattern::DOTALL
11         );
12     const string BoldWidget::REGEXP = "'''.+?'''";
13     BoldWidget::BoldWidget(ParentWidget& parent, string text) {
14         Matcher *match = pattern.matcher(text);
15         match -> find();
16         addChildWidgets(match -> group(1));
17     }
18     string BoldWidget::render() {
19         StringBuffer *html = new StringBuffer("<br>");
20         html -> append(childHtml()).append("<br>");
21         return html -> toString();
22     }

```

KL14

[Namensgebung] Der Klassenname ist aussagekräftig und zweckbeschreibend.

KL15

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

[Komplexität] Die Komplexität der Klasse ist passend.

KL16

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

[Klarheit] Die Funktionalitäten der Klasse sind eindeutig.

KL17

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

GE01

Checkliste: Gesamt

- [Klarheit] Lesbarkeit beschreibt den mentalen Aufwand, der benötigt wird, um den Code zu verstehen. Wie bewerten Sie die Lesbarkeit des Codeabschnitts?

GE02

```

1  class UserValidator {
2    private:
3      Cryptographer cryptographer;
4
5    public:
6      bool checkPassword(string userName, string password);
7    };
8
9  bool UserValidator::checkPassword(string userName, string password) {
10    User *user = UserGateway::findByName(userName);
11    if (user != NULL) {
12      string codedPhrase = user -> getPhraseEncodedByPassword();
13      string phrase = cryptographer.decrypt(codedPhrase, password);
14      if (phrase.compare("Valid Password") == 0) {
15        Session::initialize();
16        return true;
17      }
18    }
19    return false;
20 }
```

GE03

[Klarheit] Lesbarkeit kann so definiert werden, dass sie den mentalen Aufwand beschreibt, den benötigt wird, um den Code zu verstehen. Die Lesbarkeit des vorliegenden Codes ist angemessen.

stimme überhaupt	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz
nicht zu				zu

GE04

```
1 class PrintPrimes {
2     public:
3         friend int main(char* argv[]);
4     };
5
6     int main(char* argv[]) {
7         const int M = 1000;
8         const int RR = 50;
9         const int CC = 4;
10        const int WW = 10;
11        const int ORDMAX = 30;
12        int* P = new int[M + 1];
13        int PAGENUMBER;
14        int PAGEOFFSET;
15        int ROWOFFSET;
16        int C;
17        int J;
18        int K;
19        bool JPRIME;
20        int ORD;
21        int SQUARE;
22        int N;
23        int* MULT = new int[ORDMAX + 1];
24
25        J = 1;
26        K = 1;
27        P[1] = 2;
28        ORD = 2;
29        SQUARE = 9;
```

```
30    while (K < M) {
31        do {
32            J = J + 2;
33            if (J == SQUARE) {
34                ORD = ORD + 1;
35                SQUARE = P[ORD] * P[ORD];
36                MULT[ORD - 1] = J;
37            }
38            N = 2;
39            JPRIME = true;
40            while (N < ORD && JPRIME) {
41                while (MULT[N] < J) {
42                    MULT[N] = MULT[N] + P[N] + P[N];
43                    if (MULT[N] == J)
44                        JPRIME = false;
45                }
46                N = N + 1;
47            }
48        } while (!JPRIME);
49        K = K + 1;
50        P[K] = J;
51    }
52    {
53        PAGENUMBER = 1;
54        PAGEOFFSET = 1;
55        while (PAGEOFFSET <= M) {
56            cout << "The First " << M <<
57            " Prime Numbers --- Page " << PAGENUMBER;
58            cout << "";
59            for (ROWOFFSET = PAGEOFFSET; ROWOFFSET < PAGEOFFSET + RR; ROWOFFSET++) {
60                for (C = 0; C < CC; C++) {
61                    if (ROWOFFSET + C * RR <= M) {
62                        cout << setw(2) << P[ROWOFFSET + C * RR];
63                        cout << " ";
64                    }
65                }
66                PAGENUMBER = PAGENUMBER + 1;
67                PAGEOFFSET = PAGEOFFSET + RR * CC;
68            }
69        }
70    }
71    return 0;
72 }
```



[Klarheit] Lesbarkeit kann so definiert werden, dass sie den mentalen Aufwand beschreibt, der benötigt wird, um den Code zu verstehen. Die Lesbarkeit des vorliegenden Codes ist angemessen.

stimme überhaupt nicht zu
stimme eher nicht zu
teils teils
stimme eher zu
stimme voll und ganz zu

GE06

```
1  class CodeAnalyzer {
2      private:
3          int lineCount;
4          int maxWidth;
5          int widestLineNumber;
6          LineWidthHistogram lineWidthHistogram;
7          int totalChars;
8
9          static void findJavaFiles(File& parentDirectory, list<File>& files);
10         void measureLine(string line);
11         void recordWidestLine(int lineSize);
12         int lineCountForWidth(int width);
13         vector<Integer> getSortedWidths();
14     public:
15         CodeAnalyzer();
16         ~CodeAnalyzer();
17         static list<File> findJavaFiles(File& parentDirectory);
18         void analyzeFile(File& javaFile) throw();
19         int getLineCount();
20         int getMaxLineWidth();
21         int getWidestLineNumber();
22         LineWidthHistogram getLineWidthHistogram();
23         double getMeanLineWidth();
24         int getMedianLineWidth();
25     };
26
27     CodeAnalyzer::CodeAnalyzer() {
28         LineWidthHistogram lineWidthHistogram = LineWidthHistogram();
29     }
30 }
```

```
31  CodeAnalyzer::~CodeAnalyzer() {
32  [ delete lineWidthHistogram;
33  ]
34
35  list<File> CodeAnalyzer::findJavaFiles(File& parentDirectory) {
36  list<File> files;
37  findJavaFiles(parentDirectory, files);
38  return files;
39  }
40
41  void CodeAnalyzer::findJavaFiles(File& parentDirectory, list<File>& files) {
42  for (File file : parentDirectory.listFiles()) {
43  if (file.getName().endsWith(".java"))
44  files.insert(files.begin(), file);
45  else if (file.isDirectory())
46  findJavaFiles(file, files);
47  }
48  }
49
50  void CodeAnalyzer::analyzeFile(File& javaFile) throw() {
51  FileReader fr = FileReader(javaFile);
52  BufferedReader br = BufferedReader(fr);
53  string line;
54  while (!(line = br.readLine()).empty()) {
55  measureLine(line);
56  }
57  }
58 }
```

```
59 void CodeAnalyzer::measureLine(string line) {
60     lineCount++;
61     int lineSize = line.length();
62     totalChars += lineSize;
63     lineWidthHistogram.addLine(lineSize, lineCount);
64     recordWidestLine(lineSize);
65 }
66
67 void CodeAnalyzer::recordWidestLine(int lineSize) {
68     if (lineSize > maxWidthWidth) {
69         maxWidthWidth = lineSize;
70         widestLineNumber = lineCount;
71     }
72 }
73
74 int CodeAnalyzer::getLineCount() {
75     return lineCount;
76 }
77
78 int CodeAnalyzer::getMaxLineWidth() {
79     return maxWidthWidth;
80 }
81
82 int CodeAnalyzer::getWidestLineNumber() {
83     return widestLineNumber;
84 }
85
86 LineWidthHistogram CodeAnalyzer::getLineWidthHistogram() {
87     return lineWidthHistogram;
88 }
89
```

```

90  double CodeAnalyzer::getMeanLineWidth() {
91  |   return (double)totalChars/lineCount;
92  }
93
94  int CodeAnalyzer::getMedianLineWidth() {
95  |   vector<Integer> sortedWidths = getSortedWidths();
96  |   int cumulativeLineCount = 0;
97  |   for(int width : sortedWidths) {
98  |       cumulativeLineCount += lineCountForWidth(width);
99  |       if (cumulativeLineCount > lineCount/2)
100 |           return width;
101 |   }
102 |   throw Error("Cannot get here");
103 }
104
105 int CodeAnalyzer::lineCountForWidth(int width) {
106 |   return lineWidthHistogram.getLinesforWidth(width).size();
107 }
108
109 vector<Integer> CodeAnalyzer::getSortedWidths() {
110 |   vector<Integer> widths = lineWidthHistogram.getWidths();
111 |   vector<Integer> sortedWidths = widths;
112 |   sort(sortedWidths, sortedWidths + widths.size());
113 |   return sortedWidths;
114 }
```

[Klarheit] Lesbarkeit kann so definiert werden, dass sie den mentalen Aufwand beschreibt, den man benötigt wird, um den Code zu verstehen. Die Lesbarkeit des vorliegenden Codes ist angemessen.

GE07

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

GE08

```
1 static string testableHtml(
2     PageData pageData,
3     bool includeSuiteSetup
4 ) {
5     WikiPage *wikiPage = pageData.getWikiPage();
6     StringBuffer *buffer = new StringBuffer();
7     if (pageData.hasAttribute("Test")) {
8         if (includeSuiteSetup) {
9             WikiPage *suiteSetup =
10                 PageCrawlerImpl::getInheritedPage(
11                     SuiteResponder::SUITE_SETUP_NAME, *wikiPage
12                 );
13             if (suiteSetup != NULL) {
14                 WikiPagePath *pagePath =
15                     suiteSetup.getPageCrawler().getFullPath(suiteSetup);
16                 string pagePathName = PathParser.render(pagePath);
17                 buffer -> append("!include -setup .")
18                     -> append(pagePathName)
19                     -> append("\n");
20             }
21         }
22         WikiPage *setup =
23             PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
24         if (setup != NULL) {
25             WikiPagePath *setupPath =
26                 setup.getPageCrawler().getFullPath(setup);
27             string setupPathName = PathParser.render(setupPath);
28             buffer -> append("!include -setup .")
29                 -> append(setupPathName)
30                 -> append("\n");
31         }
32     }
```

```

33     buffer.append(pageData.getContent());
34     if (pageData.hasAttribute("Test")) {
35         WikiPage *teardown =
36             PageCrawlerImpl.getInheritedPage("TearDown", wikiPage);
37         if (teardown != NULL) {
38             WikipagePath tearDownPath =
39                 wikiPage.getPageCrawler().getFullPath(teardown);
40             string tearDownPathName = PathParser.render(tearDownPath);
41             buffer -> append("\n")
42                 -> append("!include -teardown .")
43                 -> append(tearDownPathName)
44                 -> append("\n");
45         }
46         if (includeSuiteSetup) {
47             WikiPage *suiteTeardown =
48                 PageCrawlerImpl.getInheritedPage(
49                     SuiteResponder.SUITE_TEARDOWN_NAME,
50                     wikiPage
51                 );
52             if (suiteTeardown != NULL) {
53                 WikiPagePath *pagePath =
54                     suiteTeardown.getPageCrawler().getFullPath(suiteTeardown);
55                 buffer -> append("!include -teardown .")
56                     -> append(pagePathName)
57                     -> append("\n");
58             }
59         }
60     }
61 }
62 pageData.setContent(buffer.toString());
63 return pageData.getHtml();
64 }
```

[Klarheit] Lesbarkeit kann so definiert werden, dass sie den mentalen Aufwand beschreibt, den benötigt wird, um den Code zu verstehen. Die Lesbarkeit des vorliegenden Codes ist angemessen.

GE09

stimme überhaupt nicht zu	stimme eher nicht zu	teils teils	stimme eher zu	stimme voll und ganz zu
---------------------------	----------------------	-------------	----------------	-------------------------

```
1 class BoldWidget : public ParentWidget {
2     public:
3         static const string REGEXP;
4         static const Pattern& pattern;
5         ParentWidget* parent;
6         BoldWidget(ParentWidget& parent, string text);
7         string render();
8     };
9     const Pattern& BoldWidget::pattern = Pattern::compile("'''(.+?)'''",
10     | | | Pattern::MULTILINE + Pattern::DOTALL
11 );
12     const string BoldWidget::REGEXP = "'''.+?'''";
13     BoldWidget::BoldWidget(ParentWidget& parent, string text) {
14         Matcher *match = pattern.matcher(text);
15         match -> find();
16         addChildWidgets(match -> group(1));
17     }
18     string BoldWidget::render() {
19         StringBuffer *html = new StringBuffer("<br>");
20         html -> append(childHtml()).append("<br>");
21         return html -> toString();
22     }
```

GE10

[Klarheit] Lesbarkeit kann so definiert werden, dass sie den mentalen Aufwand beschreibt, den man benötigt wird, um den Code zu verstehen. Die Lesbarkeit des vorliegenden Codes ist angemessen.

GE11

stimme überhaupt nicht zu stimme eher nicht zu teils teils stimme eher zu stimme voll und ganz
nicht zu zu

Sie haben es geschafft! Vielen Dank für Ihre Teilnahme. Sie dürfen das Fenster nun schließen.

Random Informatikwitze zur Belohnung:

Woran erkennt man eine*n extrovertierte*n Informatiker*in?

Er*Sie schaut beim Reden auf DEINE Schuhe.

Wie viele Programmierer*innen braucht man, um eine Glühbirne zu wechseln?

Keine*n, es ist ein Hardware-Problem.

Wieviel Platz wurde in der EU frei, als der Brexit beendet wurde?

1 GB

Passant: „Wissen Sie wie viel Uhr wir haben?“

Informatiker*in: „Ja, weiß ich.“

Wie heißt eine Schlange, die circa 3,14 Meter lang ist?

πthon

Ein*e Informatiker*in ist Vater*Mutter geworden. Ein*e Freund*in fragt: „Ist es ein Junge oder ein Mädchen?“

Informatiker*in: „Ja“

B.Sc. Maurice Eichenseer, Technische Universität Chemnitz – 2022