

FAQs – Häufig gestellte Fragen

Okay, ehrlich gesagt haben wir keine Ahnung, ob diese Fragen häufig gestellt werden. Wir haben zu selten unsere Strichliste dabei. Aber es sind zumindest Fragen, von denen wir denken, dass man sie haben könnte. Oder sollte. Kein Druck. Und einige der Punkte - vor allem die später genannten - sind durch Anmerkungen in früheren Lehrevaluationen veranlasst.

- [Ich habe meine iLearn-Anmeldung vergeigt. Was soll ich tun?](#)
- [Wo kann ich außerhalb der praktischen Übungen arbeiten?](#)
- [Wie frage ich so um Hilfe, dass Leute mir gerne und schnell helfen?](#)
- [Wenn ich Z eingebe, taucht auf dem Bildschirm ein Y auf. Was soll ich tun?](#)
- [Warum startet mein Programm nicht?](#)
- [Warum sagt Java mir irgendwas von 32 und 64 Bit?](#)
- [Wie kann ich in Eclipse schnell die main -Methode generieren?](#)
- [Warum funktioniert mein Test nicht?](#)
- [Warum warnt mich Eclipse davor, dass eine "serialVersionUID" fehlt?](#)
- [Ich kann schon Java!](#)
- [Warum Java? Ich finde <beliebige Sprache> besser!](#)
- [Warum dieses komische ACM Java?](#)
- [Warum Englisch? Ich finde <beliebige Sprache> besser!](#)
- [Warum mosert der Prof, wenn ich meinen Laptop aufklappe und nicht hinten sitze?](#)
- [Weswegen schreibt der Prof die Folien ab?](#)
- [Ist das laut/stickig hier im GAP! Was soll ich tun?](#)
- [Soll ich kostbare Lebenszeit für die EvaSys-Umfrage opfern?](#)
- [Gilt auch noch meine Klausurzulassung vom Vorjahr?](#)
- [Wie werden die Punkte bei den automatisch getesteten Aufgaben vergeben?](#)

Ich habe meine iLearn-Anmeldung vergeigt. Was soll ich tun?

Auf der [Über-Seite im iLearn](#) steht die Kontaktperson für derlei Fälle, nämlich ssm. Schreibt ihm eine nette Mail, dann kümmert er sich drum.

Wo kann ich außerhalb der praktischen Übungen arbeiten?

Eine Übersicht über Rechner- und Arbeitsräume findet sich [hier](#). Wenn im Grundausbildungspool (kurz "GAP", in der Hermann-Rodewald-Straße 3, Raum 105a/b) genug Platz ist kann man sich üblicherweise einfach hinsetzen und dort arbeiten. Wenn zu dem Zeitpunkt ohnehin gerade eine Programmierungsübung stattfindet, kann man sogar unseren Hilfskräften Fragen stellen. Toll!

Wie frage ich so um Hilfe, dass Leute mir gerne und schnell helfen?

Du hast ein Problem mit deiner Abgabe zu einer Hausaufgabe? Oder irgendein anderes Problem? Wende dich gerne, falls es passt, an die Hiwis in deiner Übungsgruppe oder, falls das nicht passt, per Mail an deinen Übungsleiter. Um die Wahrscheinlichkeit zu erhöhen, dass er dir gerne hilft, solltest du ein paar Dinge beachten.

1. Erwähne deinen Namen. Im Absender steht oft genug nur "stu123456". Die wenigsten kennen die stu-Nummern all ihrer Studierenden auswendig.
2. Erwähne, in welcher Übungsgruppe von welcher Veranstaltung du bist (Zeit und Übungsleiterperson).
3. Liefere alle Informationen mit, die möglicherweise helfen könnten. Wir könnten sie auch im iLearn suchen, aber das ist aufwendig und bei der Anzahl von Mails, die wir so kriegen, schwer machbar und ehrlich gesagt auch einfach nervig. 😊 Bei einer problematischen Abgabe beispielsweise sind folgende Infos praktisch:
 - a. Der Quellcode.
 - b. Die genaue Testausgabe.
 - c. Eventuelle Theorien dazu, was das Problem sein könnte.
 - d. Schritte, die bislang nicht zur Problemlösung geführt haben.

Die Tips hier gelten übrigens nicht nur in unserem Kontext. Generell steigt die Wahrscheinlichkeit, dass Leute einem helfen, mit der Mühe, die man sich macht, ihnen alle möglicherweise relevanten Informationen zu liefern.

Wenn ich Z eingebe, taucht auf dem Bildschirm ein Y auf. Was soll ich tun?

Im GAP kann man zwischen englischem und deutschem Tastaturlayout umschalten. Das macht man, indem man Strg-Shift gleichzeitig drückt.

Warum startet mein Programm nicht?

Du hast ein Programm geschrieben, welches im Prinzip top aussieht, sich aber nicht starten lässt. Zum Beispiel das hier:

```
import acm.program.ConsoleProgram;

public class Wubbel extends ConsoleProgram {
    public void run() {
        println("Yo!");
    }
}
```

Laut den Beispielen in unserem Buch sollte das Programm laufen. Tatsächlich brauchen wir aber in unserer Vorlesung noch eine sogenannte `main`-Methode, die unser Programm startet:

```
import acm.program.ConsoleProgram;

public class Wubbel extends ConsoleProgram {
    public void run() {
        println("Yo!");
    }

    public static void main(String[] args) {
        new Wubbel().start();
    }
}
```

Nun kann man das Programm wie gewohnt per Rechtsklick *Run As Java Application* starten.

Wie man die `main`-Methode ganz schnell durch Eclipse generieren lassen kann steht übrigens im nächsten Eintrag.

Warum sagt Java mir irgendwas von 32 und 64 Bit?

Beim Starten des Programms kommt eine Fehlermeldung wie diese hier:

```
Exception in thread "main" java.lang.UnsatisfiedLinkError: C:\Users\xxxxx\GCMDLN.DLL: Can't load IA 32-bit .dll on a AMD 64-bit platform
```

Das liegt oft an einem von zwei Dingen:

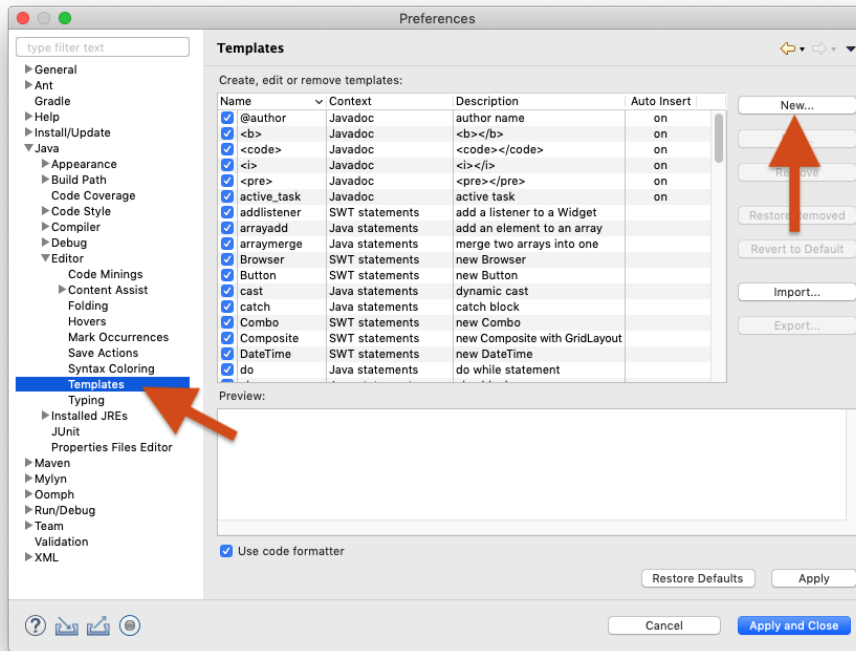
1. Es fehlt eine `main`-Methode (siehe vorige Frage).
2. Es wurde versucht, das Programm durch Klick auf den Start-Knopf in Eclipse zu starten. Besser funktioniert es, mit der rechten Maustaste in den Code zu klicken und per *Run As Java Application* zu starten. Dann weiß Eclipse nämlich exakt, was zu starten ist, während es beim Klick auf den Start-Knopf gegebenenfalls (falsch) räd.

Wie kann ich in Eclipse schnell die `main` -Methode generieren?

Man kann Eclipse so konfigurieren, dass man zum Schreiben unserer `main`-Methode nur `acmmain` eingeben muss und dann durch `Strg+Leertaste` Eclipse bitten kann, die komplette Methode zu implementieren.

1. Die *Preferences* aufrufen. Unter Mac dafür einfach *Command+Komma* drücken (oder oben links im *Eclipse*-Menu auf *Preferences* klicken). Unter Linux und Windows im *Window*-Menu auf *Preferences* klicken.

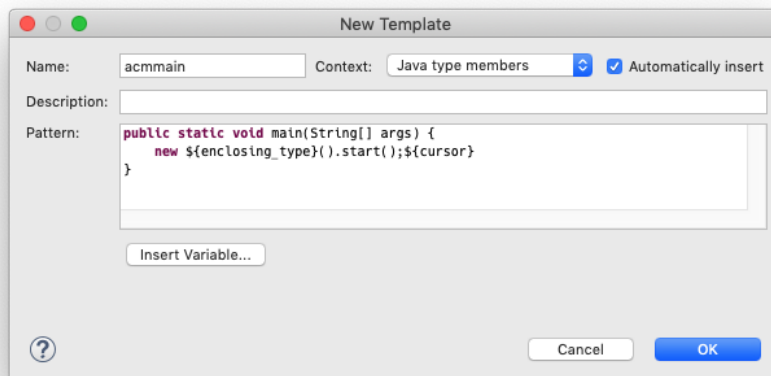
2. Im *Preferences*-Dialog links auf *Java Editor Templates* klicken. Folgende Seite sollte sich auf tun:



3. Auf *New...* klicken und den Dialog folgendermaßen ausfüllen:

Name	acmmain
Context	Java type members
Description	Egal
Pattern	<pre>public static void main(String[] args) { new \${enclosing_type}().start();\${cursor} }</pre>

Das sollte dann folgendermaßen aussehen:



4. Den *Preferences*-Dialog durch Klick auf *Apply and Close* schließen.

5. Im Körper einer Klasse kann man nun `acmmain` eingeben und nach Druck auf *Strg+Leertaste* das neue Template in vollsten Zügen genießen!

Warum funktioniert mein Test nicht?

Zunächst sollten alle Testdurchläufe bei Bedarf eine Fehlermeldung erzeugen. Diese könnt ihr sehen, wenn ihr auf den entsprechenden Durchlauf klickt. In den meisten Fällen sollte die Fehlermeldung eigentlich schon erklären, was falsch läuft. Ein paar Meldungen können aber trotzdem nochmal erklärt werden.

Falls wir das abgegebene Programm nicht kompilieren konnten, wird diese (oder zumindest eine ähnliche) Fehlermeldung erzeugt:

Testdurchlauf #82

Kompilieren nicht erfolgreich

  24.10.2017

Ausgabe des Compilers

There was an error during compilation of the program. This usually indicates some problem with the class names. Please ensure that you use **exactly the same name** as required in the assignment. Additionally you should ensure that you **only use a package when you are asked to do so**.

The following error log was generated by gradle:

```
/tmp/progl7_ez8SlMxa1CUk/src/main/java/HelloProgram.java:4: error: class Wubbel is public, should be declared in a file named Wubbel.java
public class Wubbel extends GraphicsProgram {
      ^
1 error

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':compileJava'.
> Compilation failed; see the compiler error output for details.

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.
```

Für Kompilierfehler gibt es normalerweise zwei Ursachen:

- **Falscher Klassenname:** Prüft ob die hochgeladene Klasse genau so heißt wie in der Aufgabe vorgegeben.
- **Falsches Package:** Ein Package darf nur verwendet werden, wenn das auch in der Aufgabe erwähnt wird. Besonders am Anfang werden wir noch kein Package verwenden. Die entsprechende Warnung in Eclipse könnt ihr einfach ignorieren. Später muss das Package genau so heißen wie in der Aufgabe angegeben.

Testdurchlauf #86

Fehler des Testservers

  Heute um 13:10 Uhr

Fehlermeldung des Servers

ERROR 1337: Submitted source code is not set to a valid programming language. **Please set the programming language to java in the submission interface.**

Wir akzeptieren nur Java-Code in den automatischen Tests. Eine Stolperfalle ist, dass auch im iLearn die Programmiersprache auf Java eingestellt werden muss. Hierzu müsst ihr in der kleinen Box oberhalb eurer Abgabe **Java** auswählen.

Quelltext 

Programmiersprache

```
1 import acm.graphics.GLabel;
2 import acm.program.GraphicsProgram;
3
4 public class HelloProgram extends GraphicsProgram {
5     public void run() {
6         add(new GLabel("Cigarette? - Yes, I know."), 100, 75);
7     }
8 }
9
```

Als kleiner Bonus habt ihr dann auch ein schönes Syntax-Highlighting in eurer Abgabe.

Manche Fehlermeldungen beginnen mit "The following output was generated checking the style of your code:"

Die meisten Formatierungsprobleme können in Eclipse recht einfach mit Ctrl-Shift-F (Cmd-Shift-F) behoben werden. Es gibt jedoch Ausnahmen, wie die folgenden, wobei wir gerne noch Ergänzungen dieser Liste entgegennehmen:

Name 'fTemp' must match pattern '[a-z][a-zA-Z0-9]\$'.*

Hier ist das Problem, dass der Variablenname nicht mit einem Kleinbuchstaben beginnt. Zur Erklärung der Fehlermeldung: "^" steht für den Anfang des Namens, dann soll ein Kleinbuchstabe kommen, dann beliebig viele Buchstaben oder Ziffern, dann mit "\$" das Ende des Namens.

In line 19: '}' should be on the same line.

Die (in diesem Fall zugegebenermaßen nicht sehr hilfreiche) Fehlermeldung hätte lauten sollen: "'}' should be on the same line as the else keyword". Der Code hier sah folgendermaßen aus:

```
}
else {
```

Die Fehlermeldung verschwindet bei folgendem Code:

```
} else {
```

Es kann eine andere Fehlermeldung mit einem Fehlercode angezeigt werden. Üblicherweise ist das kein Fehler von euch (zur Ausnahme siehe b), sondern etwas ist an unserer Infrastruktur kaputt. Bitte meldet solche Fehler bei eurem Übungsgruppenleiter und/oder bei [sm](#).

 **Testdurchlauf #20**

Fehler des Testservers

  Gestern um 13:50 Uhr

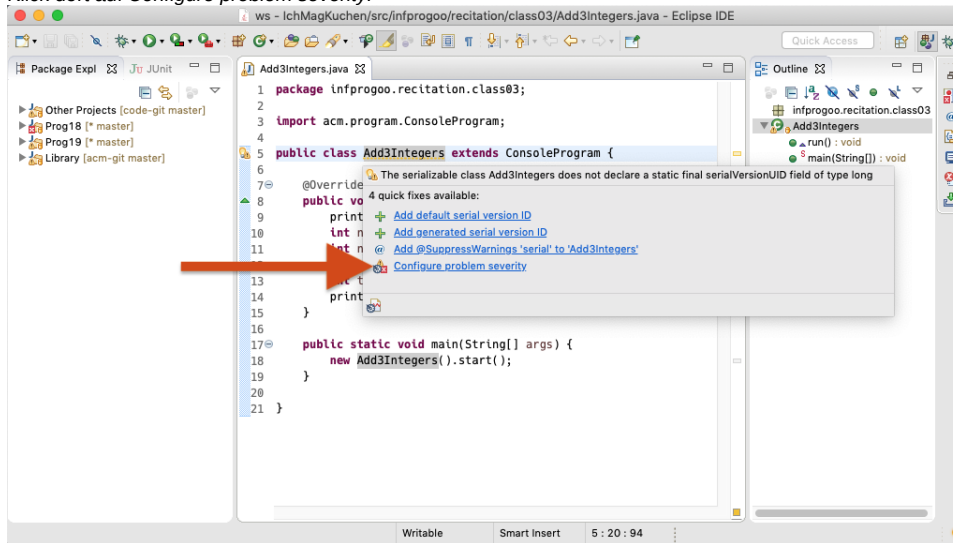
Fehlermeldung des Servers

ERROR 3547: This test is not available in your region due to GEMA regulations.

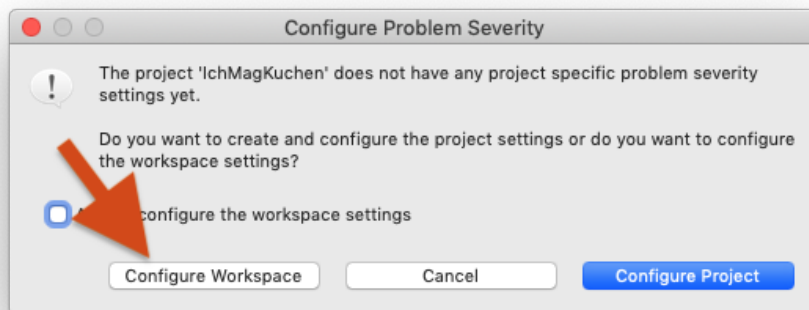
Warum warnt mich Eclipse davor, dass eine "serialVersionUID" fehlt?

Aus Gründen. Ist egal. Das kann man aber ausschalten. Dafür einfach die Anleitung ausklappen...

1. Beweg zunächst deinen Mauscursor über den gelb unterstrichenen Klassennamen. Nach einer kurzen Zeit taucht die Fehlermeldung auf. Klick dort auf *Configure problem severity*.

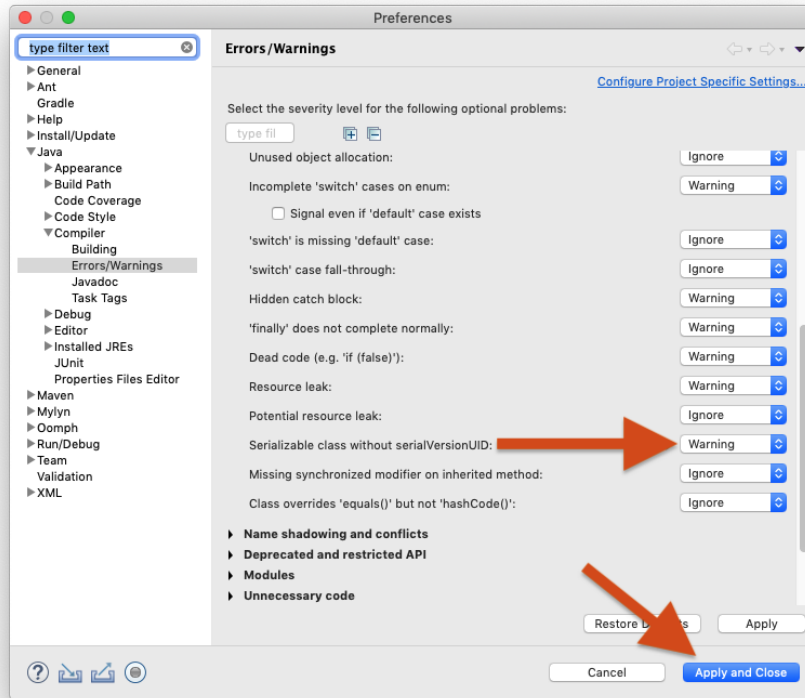


2. Wir wollen die Einstellung nicht nur für's aktuelle Projekt ändern, sondern für alle. Also klick im auftauchenden Dialog auf *Configure*

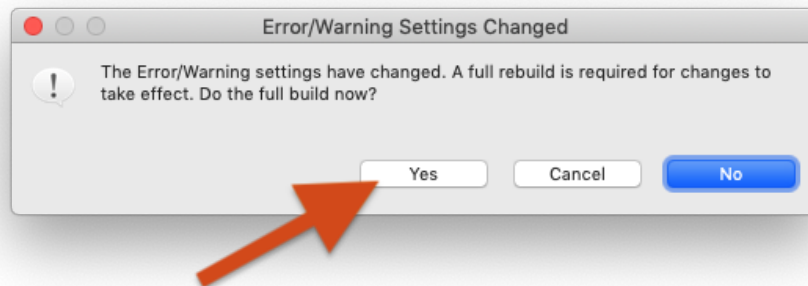


Workspace.

3. In den Optionen runter scrollen bis du die im Bild hervorgehobene Option findest, welche du dann auf *Ignore* änderst bevor auf *Apply and Close* klickst.



4. Schlussendlich musst du Eclipse erlauben, deinen Code neu zu kompilieren. Dann sollte die Warnung verschwunden sein.



Ich kann schon Java!

Voll gut! Aber erstens haben wir bislang noch niemanden getroffen, der nichts noch hätte dazulernen können, und zweitens haben wir Dinge vorbereitet, um auch Leute mit Vorerfahrung nicht zu langweilen:

1. Die meisten Hausaufgaben haben eine optionale, schwierigere Aufgabe, die schon mehr Kenntnisse voraussetzt oder generell etwas schwieriger ist.
2. Viele Kommilitonen und Kommilitoninnen können Java noch nicht. Denen könnte man helfen, indem man ihnen Dinge zu Java erklärt. Allerdings: einfach deren Hausaufgaben herunterprogrammieren hilft niemandem und ist gegen die Regeln!

Warum Java? Ich finde <beliebige Sprache> besser!

Um mal aus dem Buch zu zitieren:

The purpose of this book is to teach you the fundamentals of programming. Along the way, you will become quite familiar with a particular programming language called Java, but the details of that language are not the main point. Programming is the science of solving problems by computer, and most of what you learn from this text will be independent of the specific details of Java.

Das fasst ganz gut zusammen, worum es uns geht: wir benutzen Java als das Beispiel, an dem wir Programmieren lernen wollen. Natürlich könnten wir auch andere Sprachen benutzen, und Sie sollten auf jeden Fall neugierig auf andere Sprachen bleiben - ein paar davon werden Sie auch noch im Laufe Ihres Studiums kennenlernen. Für den Einstieg mit Java haben wir uns aus verschiedenen Gründen entschieden:

- Wir können an Java alles zeigen, was wir zeigen wollen.
- Es gibt im Internet viele Ressourcen zu Java.
- Java ist weiterhin eine der meist verwendeten Sprachen.

So listet z.B. der [TIOBE-Index](#) Java seit 2003 auf Platz 1 oder 2; andere Dauerkandidaten für die Top-3 in den letzten dreissig Jahren sind C und C++, die "Vorfahren" von Java. Selbst wenn - unwahrscheinlicherweise - ab heute keine einziges neues Programm mehr in Java geschrieben werden würde, wird es noch für geraume Zeit (d.h., voraussichtlich, bis auch Sie in den wohlverdienten Ruhestand gehen) reichlich Java-Software geben, für welche Expertise gefragt ist.

Wir könnten jetzt natürlich ewig über die speziellen Vor- und Nachteile von Java gegenüber anderen Sprachen diskutieren, aber ehrlich gesagt verbringen wir unsere Zeit lieber produktiv...

Warum dieses komische ACM Java?

Insbesondere für diejenigen, welche bereits mit Java gearbeitet haben, mag das "ACM Java" erst einmal befremdlich erscheinen, auch wenn es sich nur in Details von "richtigem Java" unterscheidet. Zunächst einmal ist "ACM Java" aber ganz normales Java, nur dass wir Packages verwenden, die von der "ACM Java Task Force" (JTF) entwickelt worden sind, um den Einstieg in die Programmierung für diejenigen zu erleichtern, welche noch kein Java können. Das Ziel der JTF war, *"To review the Java language, APIs, and tools from the perspective of introductory computing education and to develop a stable collection of pedagogical resources that will make it easier to teach Java to first-year computing students without having those students overwhelmed by its complexity."* (<http://cs.stanford.edu/people/eroberts/jtf/>). Insbesondere wird etwas historischer Ballast, den Java von C geerbt hat, versteckt, und Programme werden als Objekte behandelt. Tatsächlich sind die Unterschiede aber relativ gering, so dass zum einen diejenigen, welche schon Java können, auch keine Probleme mit den ACM Packages haben sollten, und zum anderen diejenigen, welche mit ACM Java eingestiegen sind, den gegen Ende der Vorlesung erfolgenden Umstieg auf normales Java problemlos meistern können sollten.

Und, nebenbei bemerkt – es geht in der Vorlesung, wie weiter oben erläutert, um allgemeine Konzepte der imperativen objektorientierten Programmierung; dies ist kein "Java-Kurs"!

Warum Englisch? Ich finde <beliebige Sprache> besser!

Kurze Antwort: weil Englisch für Informatiker wichtig ist.

Längere Antwort: weil Englisch für Informatiker wichtig ist. Wirklich wichtig. Deswegen auch der entsprechende Hinweis im [Studieninformationsblatt](#) (Punkt 5, Voraussetzungen und Kenntnisse) für den Informatik-Bachelor. Englisch ist bereits im Studium nützlich, nachdem hier das "wissenschaftliche Arbeiten" (§2 der Fachprüfungsordnung für den Informatik-Bachelor, in anderen FPOs werden sich ähnliche Begriffe finden) vermittelt werden soll, und die z.B. für Seminar und Abschlussarbeit relevante Literatur zum weit überwiegenden Teil auf Englisch verfügbar ist. Englisch ist auch in InfProgOO nützlich, und das nicht nur, weil viele der zur Verfügung gestellten Materialien (Buch, Folien) auf englisch sind. Etwas weiter geschaut – ein Anspruch dieser Vorlesung ist, dass erfolgreiche Teilnehmende etwas mit "Primärliteratur" wie der Java Language Specification anfangen können, welche, wie die allermeisten Sprachstandards, auf englisch gehalten ist. Auf wichtigen Foren wie Stackoverflow, welche sich Informatiker zu nutze machen können sollten, wird englisch gesprochen. Die Suche nach "programming" auf google liefert zum Zeitpunkt des Schreibens dieser Zeilen ca. 30 mal mehr Treffer als die Suche nach "Programmierung". Die allermeiste Software, ob open-source oder nicht, ist auf englisch dokumentiert und auf englisch "geschrieben" (Kommentare, Namen, etc.). So ist es z.B. auch bei Bewerbungen sicher nicht von Nachteil, wenn man darauf verweisen kann, im Studium von Anfang an (auch) mit englischen Materialien gearbeitet zu haben.

Zum Glück scheint, laut Umfrageergebnissen hierzu, für die allermeisten die Verwendung von Englisch kein erhebliches Problem zu sein, und sehr viele begrüßen ausdrücklich die Verwendung von Englisch. Tatsächlich kann Deutsch in der Informatik auch ziemlich grausam sein, mit Stapelspeichern, Programmbindern, und nicht zuletzt der Müllabfuhr, die die Halde regelmäßig von Abfall befreit. Trotzdem sind wir uns darüber im Klaren, dass Englisch eine zusätzliche Hürde sein kann, neben allen anderen Herausforderungen, die ein Studium so mit sich bringt. Von daher bemühen wir uns, den Einstieg so einfach wie möglich zu machen, z.B., indem Erläuterungen in Vorlesung und Übungen auf deutsch erfolgen, und wir auch auf deutschsprachige Literatur verweisen. Zum Schluss, vielleicht etwas zur Beruhigung, falls sich doch noch jemand Sorgen macht: in der Klausur werden Aufgaben auf englisch und deutsch gestellt, und Antworten können in beiderlei Sprachen verfasst werden.

Warum mosert der Prof, wenn ich meinen Laptop aufklappe und nicht hinten sitze?

Weil es nicht nur Laptopnutzer vom Geschehen ablenkt, sondern auch die Drumherumsitzenden (siehe z.B. [hier](#) (Abschnitt 5.3), [hier](#) und [hier](#)). Tatsächlich ist auch für die Laptopnutzer selbst der unmittelbar vorlesungsbezogene Einsatz von Laptops für Vorlesungsmitschriften nicht unbedingt förderlich (siehe z. B. diese [Studie](#) und diesen [Artikel](#)).

Weswegen schreibt der Prof die Folien ab?

Um einzelne Punkte als besonders wichtig und mitschreibenswert zu kennzeichnen. Wie bereits oben bemerkt sind handschriftliche Notizen nachgewiesenermaßen förderlich für den Lernerfolg. Weiterhin hat der Dozent tatsächlich manchmal einen Plan, wie es innerhalb der Vorlesung weitergehen soll, und auf welche dann bereits an der Tafel festgehaltenen Punkte man später nochmal verweisen kann.

Ist das laut/stickig hier im GAP! Was soll ich tun?

Es ist ok und sogar gewünscht, sich nicht nur mit dem "Personal" sondern auch untereinander über Lösungsstrategien etc. zu unterhalten. Dabei sollte aber bitte Rücksicht auf andere genommen werden. Das insbesondere was die Lautstärke betrifft. Normalerweise funktioniert das auch ganz gut. Wer sich trotzdem nachhaltig gestört fühlt, möge bitte a) die Störer freundlich bitten, leiser zu sein, oder b) sich eine ruhigere Ecke im GAP suchen. Fall c), der/die aufsichtführende Mitarbeiter/in muss um Hilfe gebeten werden, ist auch denkbar, bisher aber noch nicht vorgekommen. Und schließlich: im GAP kann es bei voller Belegung und schlecht eingestellter Lüftung schnell stickig werden. Dann bitte nicht zögern, dies kundzutun; der freundliche, normalerweise testierende Mensch am Dozententisch kann dann z.B. den Technikerservice (<http://www.inf.uni-kiel.de/de/service/technik-service>) oder das Gebäudemanagement (Herr Groth, Tel. 2656) anrufen, in der Regel erfolgt dann schnell Linderung.

Soll ich kostbare Lebenszeit für die EvaSys-Umfrage opfern?

Falls zunächst noch nicht klar ist, was die EvaSys-Umfrage ist - kein Problem, gegen Ende des Semesters trudeln diverse Mails ein, typischerweise eine für jede Pflichtlehrveranstaltung, in der man freundlich eingeladen wird, an einer Umfrage teilzunehmen. Die Teilnahme an den Umfragen ist wesentlicher Bestandteil des (Achtung:) Lehrqualitätssicherungsprozesses. Besonders interessant sind typischerweise die Freitextantworten. Nachdem es letztlich um den Studienerfolg geht, sollte gerne insbesondere auf folgende Fragen eingegangen werden: 1. Was hilft mir beim Lernen? 2. Was hindert mich am Lernen? 3. Wie könnte man die unter 2. genannten Punkte verbessern?

Die Umfrageergebnisse kommen den Dozenten zu und werden auch von einer regelmäßig tagenden (Vorsicht:) Qualitätssicherungskommission gesichtet. Hieraus wird ein Bericht generiert, der z.B. im Fachschaftsraum eingesehen werden kann. Aus hoffentlich nachvollziehbaren Gründen wird dieser nicht elektronisch verbreitet und/oder öffentlich gepostet - aber so hat man einen guten Grund, doch mal bei den netten Fachschaftlern vorbeizuschauen. Falls etwas wirklich im Argen zu sein scheint, wird ein kollegiales Gespräch mit dem/der Lehrverantwortlichen geführt. Die primäre Funktion der Lehrevaluation ist jedoch, dass Vorlesungsteilnehmende einen einfachen, anonymen Weg haben, den Lehrenden direkt mitzuteilen, was ihnen auf der Seele brennt.

Übrigens: Dozenten sind auch Menschen! D.h., Kritik kommt wirksamer an, wenn sie freundlich & konstruktiv formuliert wird, und auch loben darf man gerne mal. Und: Lehrevaluationen sind mittlerweile auch üblicher Bestandteil von Bewerbungen auf Dozentenstellen; d.h., insbesondere für Übungsleiter, die sich später mal bewerben wollen, können (hoffentlich positive) Lehrevaluationen wichtig sein. Dafür ist natürlich auch wichtig, dass - im Falle von mehreren Übungsleitern - das Feedback persönlich zugeordnet werden kann, oder zumindest die betreffende Übungsgruppe ("Fr 20-22 Uhr") identifiziert wird.

Gilt auch noch meine Klausurzulassung vom Vorjahr?

Ja. Aber es ist natürlich trotzdem eine gute Idee, sich nochmal mit den Übungsaufgaben auseinanderzusetzen.

Wie werden die Punkte bei den automatisch getesteten Aufgaben vergeben?

Sofern die Aufgabe nicht einen eigenen Verteilungsschlüssel angibt, bekommt man die volle Punktzahl sobald die Tests **und** die Stylechecks fehlerfrei durchlaufen. Sobald die Hälfte der Tests erfolgreich durchlaufen, erhält man die Hälfte der Testpunkte plus die Stylepunkte, wenn zusätzlich die Stylechecks auch erfolgreich sind. In der Regel werden 10 Stylepunkte vergeben. Bei schlechtem Code können noch einzelne Punkte abgezogen werden. Zum Beispiel: Gibt eine automatisch getestete Aufgabe 30 Punkte, bekommt man 10 Punkte wenn mind. die Hälfte der Tests durchlaufen, 20 Punkte wenn die Hälfte der Tests und die Stylechecks erfolgreich sind und 30 Punkte, wenn alles erfolgreich ist. In Grenzfällen bzgl. des Bestehens der Serie werden auch die automatisch getesteten Aufgaben hinterher noch einmal manuell überprüft. Das Schema gilt natürlich nicht für die ersten Aufgaben, in denen keine Stylechecks erforderlich waren.