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?
- Warum funktioniert mein Test nicht?
    - a) Fehler beim Kompilieren
    - b) Keine Programmiersprache ausgewählt
    - c) Der Style-Checker meckert obwohl Eclipse meinen Code formatiert hat
        - Name 'Ftemp' must match pattern '^[a-z][a-zA-Z0-9]*$'.
        - In line 19: '}' should be on the same line.
    - d) Der Test geht gar nicht
- Ich kann schon Java!
- Mir ist langweilig! Und was ist C4?
- 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?
- Warum besteht in den praktischen Übungen Anwesenheitspflicht?
- Ist das laut/stickig hier im GAP! Was soll ich tun?
- Schließlich, am Ende des Semesters: Soll ich kostbare Lebenszeit für die EvaSys-Umfrage opfern?

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

Auf der Über-Seite im iLearn steht die Kontaktperson für derlei Fälle, Jan Tikovsky. 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 "stuIrgendwas". Die wenigsten kennen alle stu-Nummern auswendig.
2. Erwähne, in welcher Übungsgruppe von welcher Veranstaltung du bist.
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.

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.

a) Fehler beim Kompilieren

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/prog17_ez8SlMxa1CUk/src/main/java/HelloProgram.java:4: error: class Wub
bel 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 --de
bug 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.

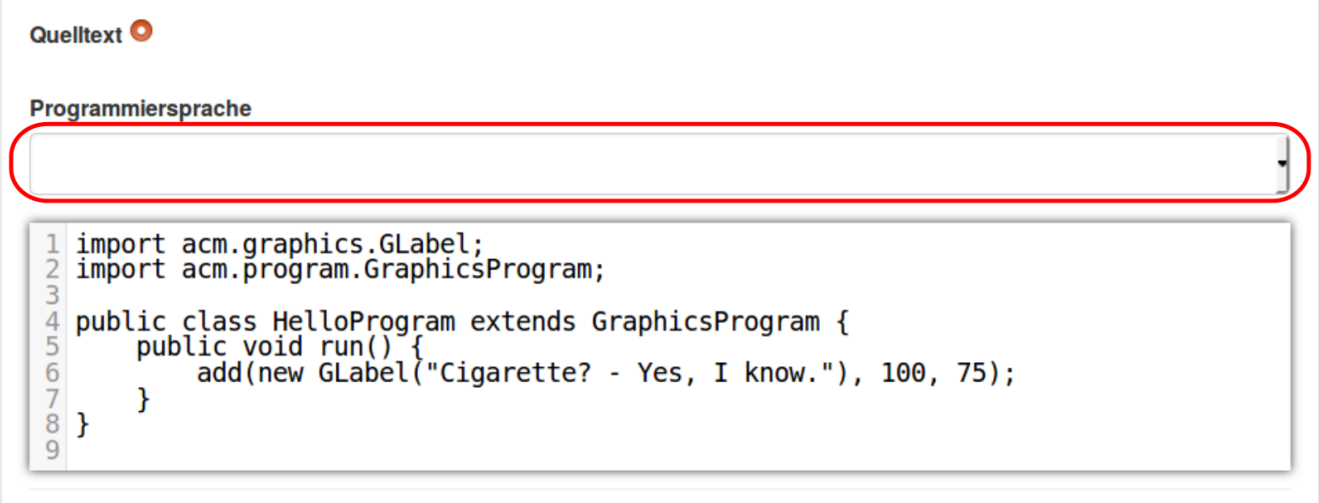
b) Keine Programmiersprache ausgewählt

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.



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

c) Der Style-Checker meckert obwohl Eclipse meinen Code formatiert hat

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 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 {
```

d) Der Test geht gar nicht

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 Nis (nbw@informatik.uni-kiel.de).

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.

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!
3. Wer an herausfordernden Programmieraufgaben Spaß hat, kann ab dem 2. Semester an Christoph Daniels *Programming Challenges*-Kurs teilnehmen. Als Hinführung dazu bietet C4 einen Haufen von Aufgaben, an denen man sich die Zähne ausbeißen kann. Am besten auch gleich mit anderen zusammen!

Mir ist langweilig! Und was ist C4?

Für die Glücklichen, die in den praktischen Übungen sitzen und den Aufgabenzettel incl. Bonusaufgabe schon komplett gelöst haben, sind die C4 CAU Coding Challenges sind ein vortreffliches Mittel gegen Langeweile. Die meisten der C4 Aufgaben sind dem alljährlich stattfindenden ACM-ICPC International Collegiate Programming Contest entnommen, an dem übrigens auch Kieler Informatikstudierende teilnehmen können (bei Interesse fragt den Dozenten). Viele der Programmierprobleme setzen eher fortgeschrittene algorithmische Kenntnisse voraus, aber folgende Aufgaben sind auch gut für Einsteiger geeignet: All in All, Beat the Spread, Counting, Fibonacci, Fibs, I Love Big Numbers, Light More Light, Odd Sum, Reverse and Add, The Trip, Vito's Family, Word Scramble. Für den Anfang empfehlen wir jedoch den Example-Challenge, um sich mit C4 etwas vertraut zu machen.

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. Für 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.

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. Mehr Details finden sich hier. 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. Mobile Geräte in der Präsenzlehre: Ablenkung oder Lernchance? Von der unstrukturierten Nutzung von Smartphone & Co. hin zu einem orchestrierten Modell für Vorlesungen (Abschnitt 5.3), Laptop multitasking hinders classroom learning for both users and nearby peers und Why I Just Asked My Students To Put Their Laptops Away). 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.

Warum besteht in den praktischen Übungen Anwesenheitspflicht?

Kurze Antwort: wegen der Minitestate. Zitat aus der EvaSys-Umfrage vom WS15/16: "Ich finde das Losverfahren zu den Testaten sehr clever gelost, da man nicht weiß, wann man gepruft wird und sich grundsatzlich immer vorbereiten muss."

Ganz lange Antwort: zunächst einmal *nicht*, weil es uns Spaß macht, Anwesenheiten zu kontrollieren, Leute zu gängeln, oder dogmatische Diskussionen über Sinn und Unsinn von Zwängen jeglicher Art während des Studiums und überhaupt zu führen. Tatsächlich ändern die knapp zwei Stunden Anwesenheitspflicht in der Woche, die ca. 10% der für ein 10-ECTS-Modul vorgesehenen nominellen work load entsprechen, wenig daran, dass Eigenmotivation & Selbstorganisation von Anfang an wichtig für ein erfolgreiches Studium sind und die Verantwortung für den Lernerfolg letztlich auf Seiten der Studierenden liegt, auch in InfProgOO.

Warum also dann Anwesenheitspflicht in den praktischen Übungen, wo dies doch in anderen Lehrveranstaltungen kaum verlangt wird, der verantwortliche Dozent (RvH) in anderen Veranstaltungen auch nicht auf die Idee kommt, Anwesenheitspflicht zu fordern, und wir generell bestrebt sind, unsere Angebote so gut zu machen, dass die Veranstaltungen auch ohne Zwang gut besucht sind? Antwort: primär, um einen (relativ) verlässlichen Rahmen für einen regelmäßigen, persönlichen Austausch zwischen Lehrenden und Studierenden zu schaffen, der bei uns unter dem Namen "Minitestate" läuft. "Austausch" mag hier zunächst etwas nach einem Euphemismus klingen, aber genau das ist damit gemeint: wir unterhalten uns und schauen gemeinsam auf Abgaben, damit beide Seiten wissen, wie der Stand ist, und wo evt. noch nachgesteuert werden sollte. O-Ton eines Testanden: "man geht aus einem Testat immer schlauer heraus als man herein gekommen ist."

Es wäre sicher denkbar, die Minitestate auch ohne Anwesenheitspflicht zu regeln, mit individuellen Terminvereinbarungen. Jedoch wäre dies zum einen weniger "clever" im Sinne des Eingangszitats; zum anderen, für uns zugegebenermaßen noch wichtiger, würde dies bei der Anzahl abzunehmender Testate pro Semester (im WS 15/16 waren es 643 an der Zahl) einen noch deutlich größeren administrativen Aufwand bedeuten, selbst wenn wir optimistischerweise annehmen, dass 95% der Terminvereinbarungen reibungslos klappen würden. Aber selbst hierüber würden wir noch nachdenken, wenn wir das Gefühl hätten, dass die Anwesenheitspflicht ein ernsthaftes Problem für Studierende wäre, und das soweit geäußerte Unverständnis der Anwesenheitspflicht (ist vorgekommen, deswegen dieser ausführliche Text) repräsentativ wäre. Tatsächlich scheint jedoch die breite Mehrheit der Vorlesungsteilnehmenden die jetzige Regelung nachvollziehen zu können und als zumutbar zu empfinden. Von 225 anonym abgegebenen Freitextkommentaren, welche in einer laufenden Umfrage im WS 16/17 abgegeben wurden, äußerte sich keiner (in Worten: keiner) zur Anwesenheitspflicht. (Übrigens auch nicht in Sachen "Englisch", siehe oben.) Bei einer Umfrage unter den Klausurteilnehmenden im WS 15/16 haben sich 75 von 113 Befragten explizit *für* eine Beibehaltung der Anwesenheitspflicht ausgesprochen (für die komplette Umfrage siehe http://www.informatik.uni-kiel.de/~rvh/downloads/UmfrageEndeWS2015-16.pdf). Wir bemühen uns auch, die praktischen Unannehmlichkeiten eines festen Termins in der Woche so gering wie möglich zu halten, auch z.B. unter dem Aspekt der Familienfreundlichkeit. So bekamen bisher immer alle Teilnehmenden von den möglichen Übungsterminen (in der Regel derer 10) einen ihrer Top-3-Wünsche (dank Informatik!), die meisten sogar ihren Erstwunsch. Und wenn es in Einzelfällen harte Zeitkonflikte gegeben hat, haben wir stets einvernehmliche Lösungen gefunden.

Schließlich auch nochmal die positive Sicht - über die man sicher beim Kaltgetränk trefflich philosophieren kann: das "Gruppenerlebnis" beim Programmieren und die unmittelbare Verfügbarkeit von erfahrenen Ansprechpartnern, ein Gutteil davon aus Steuergeldern bezahlt, können tatsächlich hilfreich sein. Dies vielleicht auch und gerade für diejenigen, die sonst eher "im stillen Kämmerlein" alleine an den Aufgaben sitzen würden. Und ja, es gibt sicher auch Leute, die schon vorher (fast) alles wissen, und es vielleicht zu recht als überflüssig empfinden, zwangsverpflichtet zu werden, gemeinsam mit ihren weniger vorgebildeten Studiengenossen Zeit im GAP zu verbringen. Aber denjenigen sei gesagt, dass nicht nur selbst Dinge lernen, sondern auch anderen Leuten Dinge zeigen Spaß machen kann - wir wissen, wovon wir reden. Und neben der Gelegenheit, seinen Mitmenschen zu helfen, gibt es gegen eventuelle Langeweile noch Dinge wie das C4 System. Und jetzt danke für's Lesen bis hierher und viel Spaß beim Programmieren 🙂

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. Deswegen sollten beide Türen geöffnet sein, damit die Lüftung vernünftig arbeiten kann. Falls es trotzdem stickig wird, bitte nicht zögern, dies kundzutun; der freundliche, normalerweise testierende Mensch am Dozententisch kann dann z.B. den Technikservice (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.

Schließlich, am Ende des Semesters: 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.