Objektorientierte Programmierung

Kapitel 0: Organisatorisches

Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2012/13

http://www.informatik.uni-halle.de/~brass/oop12/

Inhalt

Lernziele

- Programmieren können

  Selbständige Erstellung eigener Programme für gegebene Aufgaben.
  Nur kleinere/einfachere Programme, z.B. werden interessantere Algorithmen und Datenstrukturen hier nicht behandelt (eigene Vorlesung).

- Rechner, Betriebssystem und Editor bzw. Entwicklungsumgebung ausreichend bedienen können.
- Gegebene Programme verstehen können.

  Z.B. Klausurfrage: Was gibt dieses Programm aus?

- Die syntaktische Korrektheit gegebener Programme beurteilen können ($\rightarrow$ Syntax-Formalismen).
- Grundkonzepte von Programmiersprachen kennen, sich leicht in neue Sprachen einarbeiten können.

Zur Programmiersprache (1)

- In dieser Vorlesung wird Programmierung anhand der Sprache "Java" gelehrt (+ Vergleich mit anderen Sprachen).
  - In der Praxis verbreitet.

    Z.B. Umfrage von iX bei 50 großen deutschen IT-Dienstleistern (2011): Alle verwenden auch Java (neben anderen Sprachen) und erwarten Java-Kenntnisse bei Bewerbern: [http://www.heise.de/developer/meldung/Umfrage-Java-ist-verbreitetste-Programmiersprache-bei-IT-Dienstleistern-1198249.html]
    Nach [http://www.langpop.com] und dem TIOBE Programming Community Index steht Java momentan auf Platz 2 (nach C).

  - Gut mit Software-Werkzeugen und Literatur unterstützt.
  - Vermeidet bestimmte Probleme.

    Manche Fehler, die man in C++ machen kann, kann man in Java nicht machen. Dafür hardwarenahe Dinge nicht auszuprobieren.

Zur Programmiersprache (2)

Auf die genaue Sprache kommt es nicht an:

- Wenn man Programmiersprachen mit Fremdsprachen vergleicht, sind Wortschatz und Syntax viel einfacher.

  Der wesentliche Aufwand beim "Programmieren lernen" sind nicht so sehr die spezifischen Konstrukte einer Programmiersprache, als zu lernen, sich in Computer hineinzudenken, und einen Schatz von Mustern für bestimmte Aufgaben im Kopf zu haben, die man nach Bedarf aktivieren und anpassen kann. Diese Muster sind aber in weiten Grenzen unabhängig von der Programmiersprache, außer bei ganz anderen Programmierparadigmen, wie etwa logischer Programmierung. Viele Sprachen stammen wie Java aus der "C-Familie".

- Sie werden noch viele andere Programmiersprachen lernen müssen.

  Z.B. C++, C#, JavaScript, PHP, SQL, sh, Prolog, verschiedene Assembler.

Motivation (1)

Warum sollten Sie programmieren lernen?

- Grundlegende handwerkliche Fähigkeit, im Berufsleben eines Informatikers notwendig.

  Selbst wenn Sie nicht an der Entwicklung wirklich großer Programme mitwirken, ist Programmierung z.B. auch nötig, um Inhalte von Datenbanken schön aufbereitet ins Netz zu stellen.

- Komplexere Software-Werkzeuge bieten für Nicht-Standard-Aufgaben meist Möglichkeiten, die ähnlich zu einer Programmiersprache sind.

  Z.B. ist die Datenbanksprache SQL zwar keine allgemeine Programmiersprache, aber hat doch vieles gemeinsam mit Programmiersprachen. Programmierkenntnisse erleichtern das Erlernen von SQL.

Motivation (2)

Warum sollten Sie programmieren lernen? (Forts.)

- Vieles in der Informatik kann man besser verstehen, wenn man es im Prinzip auch selbst programmieren könnte.
- Es erleichtert Verhandlungen mit Programmierern, wenn man auch selber programmieren kann.
- Meine Kollegen und ich sind der Meinung, dass niemand ein Diplom/Bachelor-Abschluß in Informatik, Bioinformatik, oder Wirtschaftsinformatik bekommen sollte, der nicht programmieren kann.

Motivation (3)

Warum macht Programmieren Spaß?

- Man kann mit relativ wenig Aufwand komplexe Maschinen und virtuelle Welten (Spiele) konstruieren.

  Oder reale Maschinen / elektronische Schaltungen (Microcontroller).

- Man kann sich das Leben im Umgang mit dem Computer etwas erleichtern.

  Stupide manuelle Tätigkeiten durch kurzes Programm automatisieren.

- Den Computer beherrschen statt umgekehrt.

  Computer-Experte sein: Dazu gehören Programmierkenntnisse, aber auch Hardware- und Betriebssystem-Kenntnisse.

- Konkrete Anwendung von Mathematik.

Motivation (4)

Warum ist Programmieren eine Herausforderung?

- Computer befolgen genau die gegebenen Regeln (das Programm), aber können kein bißchen selbst denken (nichts versteht sich von selbst).
- Man muß bei der Programmierung also an alles denken: Alle möglichen Fälle, auch Ausnahmen.

  Ein Programm ist wie ein mathematischer Beweis, vielleicht sogar hinsichtlich der Präzision noch anspruchsvoller. Man kann kein Glied in der Kette auslassen, oder sagen, “wie man leicht sieht” / “trivialerweise gilt”.
- Hier muß man Perfektionist sein.
- Man muß sich in die Maschine hineindenken.

Motivation (5)

Ich will programmieren lernen. Was brauche ich?

"Zunächst einmal brauchen Sie wie beim Erlernen einer Fremdsprache sehr viel Eifer — aber zusätzlich noch Freude an kreativer Arbeit und ein ausgeprägtes Abstraktionsvermögen. Letzteres bedeutet, Probleme soweit aufdröseln zu können, dass man die daraus entstehenden Teilprobleme direkt als Programm formulieren kann. Der kreative Anspruch ergibt sich daraus, dass es häufig mehrere Wege zur Lösung gibt, etwa besonders elegante oder besonders kurze — oder vermeidbar umständliche."

Oliver Lau: FAQ Programmieren lernen. c't 2008, Heft 23, S. 178.

Inhalt

Ansprechpartner (1)

Dozent: Prof. Dr. Stefan Brass

- Email: brass@informatik.uni-halle.de

  Betreff-Zeile sollte mit [oop12] beginnen, möglichst aussagefähig.

- Büro: Von-Seckendorff-Platz 1, Raum 313.
- Telefon: 0345/55-24740.
- Sprechstunde: Mittwochs, 12^{00}–13^{00}.
- Frühere Unis: Braunschweig, Dortmund, Hannover, Hildesheim, Pittsburgh, Gießen, Clausthal.
- Spezialgebiet: Datenbanken, Wissensrepräsentation.

  Oracle8 Certified DBA. IBM Certified Advanced DBA (DB2 8.1).

Ansprechpartner (2)

Übung (Gruppen 1, 4, 5, 8): Dipl.-Inform. Annett Thüring

- Büro: Von-Seckendorff-Platz 1, Raum 319.
- Telefon: 0345/55-24739.
- Email: thuering@informatik.uni-halle.de

Übung (Gruppen 2, 3, 6, 7): Dipl.-Inform. Steffen Schiele

- Büro: Von-Seckendorff-Platz 1, Raum 418.
- Telefon: 0345/55-24716.
- Email: steffen.schiele@informatik.uni-halle.de

Ansprechpartner (3)

Sekretärin: Ramona Vahrenhold

- Büro: Von-Seckendorff-Platz 1, Raum 324.
- Telefon: 0345/55-24750, Fax: 0345/55-27333.
- Email: vahrenhold@informatik.uni-halle.de
- Das Sekretariat ist dienstags nicht besetzt.

Rechnerbetriebsgruppe:

- Poolaufsicht (studentische Hilfskräfte): Raum 318.
- Daniel Trull, Lutz Ohme: Raum 320.

Webseiten

- http://www.informatik.uni-halle.de/~brass/oop12/
  - Folien.

    Die Folien werden jeweils vor der Vorlesung ins Web gestellt.
  - Alte Klausuren und Programmiertests.
  - Nützliche Links.
- StudIP: http://studip.uni-halle.de/seminar_main.php?auswahl=d3efb622b9ac02bf318dc4f919182886
  - Anmeldung zu Vorlesung und Übungsgruppen.
  - Link zu Übungsplattform (Hausaufgaben).
  - Forum (z.B. auch Fragen zu Hausaufgaben).

Zeit und Ort (1)

Vorlesung (2 SWS):

- Dienstags, 10^{15}–11^{45}, Raum 3.28.

Übung am Rechner (2 SWS):

- Acht Gruppen, je max. 28 Teilnehmer, Beginn 15./16.10.:

Nr (Raum)	Tag	Zeit	Teiln.
4 (3.32)	Montag	10^{15}–13^{45}	11
1 (3.34) / 2 (3.32)	Montag	12^{15}–13^{45}	14/12
3 (3.34)	Montag	14^{15}–15^{45}	27
5 (3.34) / 6 (3.32)	Montag	16^{15}–17^{45}	15/14
8 (3.34)	Dienstag	12^{15}–13^{45}	32
7 (3.34)	Dienstag	14^{15}–15^{45}	47

Zeit und Ort (2)

Anmeldung zu Übungsgruppen:

- Tragen Sie sich in StudIP für eine Gruppe ein.

  [http://studip.uni-halle.de/]. Direkte Links auf der Vorlesungs-Webseite:
  [http://www.informatik.uni-halle.de/~brass/oop12/termine.html].

- Wir garantieren, dass Sie einen Platz in einer der Übungsgruppen bekommen.

  Wenn Sie zur Teilnahme an dieser Vorlesung verpflichtet sind.

  Notfalls werden zusätzliche Gruppen angeboten (zu anderen Zeiten).

- Wir können leider nicht garantieren, dass Sie einen Platz in einer bestimmten Gruppe bekommen.

  Bitte wechseln Sie möglichst aus den überfüllten Gruppen in eine andere.

  Wenn sich das Problem nicht freiwillig löst, werden wir umverteilen.

Zeit und Ort (3)

Anmeldung zu Übungsgruppen, Forts.:

- Für die Anmeldung zu einer Übungsgruppe über StudIP brauchen Sie einen Benutzernamen und ein Passwort, das Ihnen mit der Immatrikulationsbescheinigung zugegangen sein müßte.

  Der Benutzername identifiziert Sie eindeutig im System (Vor- und Nachname sind nicht immer eindeutig). Das Passwort sollten nur Sie wissen. Es dient als Ausweis, dass Sie auch wirklich Sie sind.

- Bitte tragen Sie sich bei StudIP nicht nur für die Übungsgruppe Ihrer Wahl, sondern auch für die Vorlesung selbst ein.

Anmeldung: Übersicht / Checkliste

Um dieses Modul erfolgreich abzuschließen, müssen Sie

- ☐ sich bei StudIP für eine Übungsgruppe anmelden,
- ☐ sich zum Modul anmelden (Löwenportal),
- ☐ Hausaufgaben bearbeiten und abgeben,
- ☐ in den Übungen ausreichend aktiv mitarbeiten,
- ☐ den praktischen Programmiertest bestehen,
- ☐ sich zur Prüfung anmelden (Löwenportal),
- ☐ die Prüfung (Klausur) bestehen.

Modulanmeldung

- Für fast alle Studiengänge ist die Modulanmeldung über das Löwenportal Pflicht.

  [http://loewenportal.uni-halle.de/]

  Anleitung im grünen Kasten rechts unter Weitere Einstellungen/Hilfe.

  Falls über Löwenportal nicht möglich, dann im Prüfungsamt.

- Die Modulanmeldung ist nur bis zum 24.10.2012 möglich.

- Die Modulanmeldung ist zwingende Voraussetzung für die spätere Anmeldung zur Prüfung.

  Ein Zweck der Modulanmeldung ist die Prüfung von Voraussetzungen, dieses Modul hat aber keine Voraussetzungen.

Inhalt

Eigentlich sollte kein Druck nötig sein ...

- Programmieren sollte Ihnen Spass machen, oder Sie sollten den Wunsch verspüren, es zu lernen.

  Sonst hätten Sie nicht Informatiker werden sollen! Lange Zeit nur programmieren oder unter großem Druck programmieren macht mir auch keinen Spass, aber immer mal wieder finde ich bis heute schön.

- Unser Dilemma ist:
  - Wenn wir keinerlei Druck ausüben, fürchten wir, dass ein Teil der Studierenden nicht ausreichend Zeit in das Lernen investiert.

    Z.B. könnte es sein, dass der Druck in anderen Lehrveranstaltungen größer ist.

  - Ohne ausreichend Vorbereitung wird das Klausurergebnis schlecht sein.

    Das fällt dann auch auf die Dozenten zurück.

Hausaufgaben (1)

- Es gibt wöchentlich Hausaufgaben (zum größten Teil Programmieraufgaben).
- Die Aufgaben werden Mittwoch abend ins StudIP gestellt.
- Sie sind dann bis zum Mittwoch der nächsten Woche (12:00) über die Übungsplattform im StudIP abzugeben.

  Programme bitte als unter Linux mit `javac` compilierbarer Quellcode (ASCII), nicht als Word oder PDF-Dateien. Für theoretische Aufgaben wird ASCII oder PDF akzeptiert.
- Zu ähnliche (abgeschriebene) Abgaben werden mit 0 Punkten für alle Beteiligten bewertet.

  In nicht ganz klaren Fällen werden Sie Ihre Abgabe vor der ganzen Gruppe erklären müssen, oder zu einem mündlichen Test beim Professor bestellt. Es gibt Plagiats-Erkennungssoftware.

Hausaufgaben (2)

- Die Aufgaben sind einzeln oder in Gruppen bis 3 Personen zu bearbeiten.

  Alle Gruppenmitglieder müssen die von ihrer Gruppe abgegebenen Lösungen jeweils einzeln erklären können. Das wird gelegentlich in den Übungen getestet (Stichproben, bei wenigen Terminen auch vollständig mit Einsatz der Tutoren). Für Erfolg in der Klausur ist aber auch wichtig, dass jeder einzeln auch den Weg zur Lösung kennt, nicht nur fertige Lösungen verstehen kann. Ideale Gruppenarbeit: Alle Mitglieder kommen mit einer eigenen Lösung zum Treffen, man vergleicht die Ansätze, und konstruiert aus den verschiedenen Ideen gemeinsam eine "noch perfektere" Lösung.

- Wenn man gemeinsam die Lösung erarbeitet, sollten stärkere Gruppenmitglieder den schwächeren ausreichend Zeit lassen.

  Insbesondere auch die schwächeren am Rechner sitzen und tippen lassen. Das Vorwissen ist bei dieser Vorlesung sehr unterschiedlich.

Hausaufgaben (3)

- Die Hausaufgaben werden von den Tutoren (Studenten aus höherem Semester) korrigiert.

  Bei Programmieraufgaben gibt es bei der Abgabe eine automatische Vorkontrolle durch die Übungsplattform: Besser nicht bis zur letzten Minute warten, damit ggf. noch Korrektur/Hilfe möglich.

- Falls Sie die Korrektur nicht verstehen, fragen Sie bitte.

  Fehler kommen leider gelegentlich vor. Selbst wenn sich am Ende herausstellt, dass der Punktabzug berechtigt war, haben bei der Diskussion beide Seiten etwas gelernt. "Wer nicht fragt, bleibt dumm."

- Sie müssen mindestens 50% der Punkte erreichen.

  Feilschen Sie nicht wegen einem Punkt. Wenn eindeutig ein Fehler vorliegt, lohnt es sich meist nicht, über einen einzelnen Punkt zu reden. Für die Note ist nur die Klausur wichtig. Wenn Sie am Ende 49% haben, können Sie dann noch mal über einzelne Punkte diskutieren.

Hausaufgaben (4)

- Übliche Regeln für guten Programmierstil müssen eingehalten werden:
  - Sinnvolle Formatierung.

    Zeilenumbrüche, Einrückungen ähnlich wie in der Vorlesung.
  - Ausreichend Kommentare.
  - Keine üblen Programmiertricks.
- Kann der Tutor Ihr Programm nach ca. 5 Minuten noch nicht verstehen, bekommen Sie 0 Punkte.
- Wenn Ihre Abgabe nicht mit `javac` (JDK für Java SE 7) unter Linux compilierbar ist, bekommen Sie 0 Punkte.

Übung

- In der Übung werden die Hausaufgaben besprochen.

  Nutzen Sie die Gelegenheit, Alternativen zu Ihrer Lösung anzuschauen. Auch interessante Fehler werden besprochen. Bringen Sie interessante Aspekte Ihrer Lösung aktiv ein, nicht nur nach Aufforderung.

- Sie können Fragen in einer kleineren Gruppe stellen.

  Sie können Fragen selbstverständlich auch in der Vorlesung stellen!

- Lösen weiterer Aufgaben (Präsenzaufgaben) ohne Bewertung, aber mit Hilfe/Feedback.

- Bei der Übung wird eine Anwesenheitsliste geführt.

  Es gibt keine genaue Schranke, wie häufig Sie fehlen dürfen. Drei Mal wäre noch ok. Bei zu häufigem Fehlen werden Sie zu einem Gespräch gebeten. Bei Ermessensspielräumen schauen wir ggf. in der Anwesenheitsliste nach. Sie schaden sich selbst, wenn Sie nicht kommen: Es gibt wichtige Erklärungen und Beispielaufgaben (für Programmiertest/Klausur).

Programmiertest (1)

- Die Modulvorleistung (Voraussetzung für die Zulassung zur Klausur) besteht aus:
  - Erreichen von 50% der Hausaufgabenpunkte,
  - Bestehen von einem von zwei Programmiertests (praktische Prüfung am Rechner).

    Wer die Hausaufgaben selbst bearbeitet hat, und wenigstens hinterher Aufgaben von diesem Typ ohne fremde Hilfe lösen kann, sollte mit den Programmiertests keine Schwierigkeiten haben. Leider gab es in der Vergangenheit "schwarze Schafe", die sich bei den Hausaufgaben durchgemogelt haben. Wir wollen aber mit Erreichen der Modulvorleistung wenigstens ein Minimum an praktischen Programmierfähigkeiten garantieren können.

Programmiertest (2)

- Die Programmiertests findet an zwei Übungsterminen statt:
  - 10./11.12.2012 (Montags/Dienstags-Übung)
  - 22./23.01.2013

    Die Aufgaben zu den verschiedenen Übungszeiten sind natürlich unterschiedlich. Wenn Sie den ersten Test bestanden haben, brauchen Sie zum zweiten nicht mehr erscheinen (wir bitten darum, damit wir für restlichen Kandidaten mehr Zeit haben).
- Sie müssen in gegebener Zeit (60–90 Minuten) eine Methode/Klasse schreiben, die mit einem vorgegebenen Testprogramm zusammen funktioniert (ohne Hilfe).

  Natürlich muß Ihr Programm nicht nur für die Beispielaufrufe funktionieren, sondern insgesamt korrekt sein. Es gibt nur "bestanden" oder "nicht bestanden". Unfertige oder nicht richtig funktionierende Programme werden nicht akzeptiert. Bei verdeckten Fehlern, die von unseren Tests nicht erfasst werden, werden wir eher großzügig sein.

Programmiertest (3)

- Bestehen Sie die Modulvorleistung (Hausaufgaben und Programmiertest) nicht, so können Sie diese Vorlesung nur ein Jahr später erneut belegen.

  Bestimmte Module des 2. und 3. Semesters können Sie dann auch nicht belegen (weil sie OOP bzw. die Modulvorleistung als Voraussetzung haben). Bei besonderen Belastungen (längerer Krankheit etc.) wenden Sie sich an den Dozenten wegen möglicher Ersatzleistungen.

- Sollten Sie also den ersten Programmiertest nicht bestehen, werden Sie bitte aktiv.

  Möglicherweise könnten wir eine Art Nachhilfe vermitteln oder ein Tutorium anbieten, aber auf jeden Fall müssen Sie mehr Zeit investieren.

- Trost: Wenn Sie nicht zur Klausur zugelassen werden, haben Sie keinen Prüfungsversuch verbraucht.

Prüfung (1)

- Die Prüfung erfolgt als Klausur.
- Termin: 19.03.2013, 10^{00}-12^{00}, Raum 5.09 u.a.

  Der Termin ist noch nicht ganz endgültig, achten Sie auf weitere Ankündigungen. Melden Sie ggf. Konflikte möglichst frühzeitig.
- Man muß sich spätestens vier Wochen vorher zur Klausur anmelden (im Löwenportal).

  Voraussetzung: zum Modul angemeldet und Modulvorleistung erfüllt.
- Man kann sich bis drei Tage vorher ohne Angabe von Gründen wieder abmelden.

  Alle Angaben zur Prüfungsordnung sind ohne Gewähr (möglicherweise auch Studiengangs-abhängig). Informieren Sie sich bitte selbst beim Prüfungsamt oder dem Studienberater.

Prüfung (2)

- Bücher, eigene Notizen, Kopien sind erlaubt.

  Bücher sind nur nützlich, wenn man sie vorher gelesen hat. Die Zeit bei der Klausur ist begrenzt, man muß also wissen, wo man etwas suchen muß. Sich einen "Spickzettel" oder "Quick Reference" zu machen, kann eine gute Prüfungsvorbereitung sein.

- Rechner aller Art sind nicht erlaubt.

  Sie können also zu entwickelnde Programme nicht ausprobieren. Wenn man vorher nur mit langem Herumprobieren und viel Feedback vom Compiler ein funktionierendes Programm entwickeln konnte, ist man noch nicht reif für die Klausur. Üben Sie auch das Programmieren auf dem Papier und anschließendes Eingeben in den Rechner. Man denkt wahrscheinlich ohnehin besser mit etwas Abstand vom Rechner.

Prüfung (3)

- Die Note für das Modul basiert nur auf der Klausur.

  Der Dozent behält sich das Recht vor, in seltenen Ausnahmefällen Extrapunkte (bis max. 5% der Klausurpunkte) zu vergeben für das Finden von Fehlern im Skript, außergewöhnlich gute Hausaufgabenergebnisse verbunden mit entsprechend aktiver Übungsteilnahme, oder eventuell andere Aktivitäten, die allen Teilnehmern zugute kommen.

- Es wäre nicht klug, mit den Hausaufgaben vorzeitig aufzuhören (wenn Sie 50% sicher erreicht haben).

  Die Hausaufgaben sind eine wichtige Prüfungsvorbereitung.

- Falls Sie 60% der Klausurpunkte erreicht haben, haben Sie garantiert bestanden.

  Mit 50% ist das Bestehen noch nicht garantiert. Ab 95% gibt es die bestmögliche Zensur (1.0). Die Grenzen werden möglicherweise noch nach unten verschoben.

Prüfung (4)

- Falls Sie an der Klausur teilnehmen und bestehen, gibt es keine Möglichkeit zur Zensurverbesserung.
- Falls Sie bei der Klausur durchfallen, gibt es noch eine Wiederholungsmöglichkeit.

  Wenn Sie nicht erscheinen, obwohl Sie angemeldet sind, und keine Krankschreibung vorlegen, sind Sie automatisch durchgefallen. Es ist auch möglich, sich erst zum 2. Termin anzumelden. Dann gibt es aber keine Wiederholungsmöglichkeit im Rahmen dieser Vorlesung mehr.
- Nachholtermin: voraussichtlich 28.05.2013, 10–12.

  Bitte informieren Sie sich über Ihre Prüfungsordnung, inwieweit eine extra Anmeldung nötig ist, oder Sie automatisch angemeldet sind, falls Sie die erste Klausur nicht bestanden haben.

Prüfung (5)

- Falls Sie die Nachholklausur auch nicht bestehen, müssen Sie das Modul ein Jahr später neu belegen.

  Die Modulvorleistung gilt nur für zwei Prüfungsversuche.

- Es gibt maximal drei Prüfungsversuche.

  Nach dem dritten nicht erfolgreichen Prüfungsversuch für diese Vorlesung ist für Informatiker und Wirtschaftsinformatiker das Studium beendet. Im Laufe des Studiums sind nur 7 bzw. 8 zweite Wiederholungsprüfungen erlaubt, eventuell nur mit Antrag. Endgültiges Durchfallen (nach drei Prüfungsversuchen) reduziert Ihre Optionen zum Studienfachwechsel deutlich (z.B. Wirtschaftsinformatik $\rightarrow$ Wirtschaftswissenschaften ist dann nicht mehr möglich). Lassen Sie sich vor dem letzten Prüfungsversuch unbedingt kompetent beraten.

Inhalt

Rechnernutzung, Software (1)

Rechnerpools:

- PC-Pool: Raum 332

  29 PCs (Athlon64 X2 4200+, 1 GB RAM, 19" TFT-Monitor).
  Windows XP Professional / Ubuntu Linux 10.04 64bit

- ThinClient-Pool: Raum 334

  Windows Server "odin" (4x Opteron 852, 16 GB, Windows 2003)
  Linux Server "anubis" (4x Opteron 852, 16 GB, Ubuntu 8.04 64bit)
  Unix Server "turing" (4x Sparc II 400 Mhz, 2 GB, Solaris 10 sparc)

- Server für Home-Verzeichnisse (odin)

  RAID-System mit 1 Terabyte, Quota pro Nutzer: 200 MB.

- Siehe: [http://www.informatik.uni-halle.de/studium/pools/]

Rechnernutzung, Software (2)

Rechnerpools (Forts.):

- Damit Sie auf den Rechnern im Pool 3.34 arbeiten können, muß ein Benutzerkonto eingerichtet werden. Die Daten entsprechen dem StudIP-Account.

  Hierzu melden Sie sich bitte vor der ersten Übung bei der Rechneraufsicht in Raum 318 (möglichst nach 14 Uhr). Bringen Sie Ihren Studentenausweis mit. Es wird das Start-Passwort von StudIP übernommen (das Sie mit Ihrer Immatrikulation bekommen haben).

- Remote Login (per `ssh`) z.B. auf Linux-Rechner `anubis.informatik.uni-halle.de`

  Sie können z.B. `PuTTY` für das remote Login nutzen. Sie müssen für diese Vorlesung nicht unter Linux arbeiten, aber bis zum Ende Ihres Studiums sollten Sie es gelernt haben.

Rechnernutzung, Software (3)

Falls Sie einen eigenen PC haben:

- Sie brauchen mindestens das JDK für Java SE 6 oder Java SE 7 von Sun/Oracle:
  [http://www.oracle.com/technetwork/java/javase/downloads/]

  Sun hat Java entwickelt, wurde aber inzwischen von Oracle gekauft.
  JDK steht für "Java Development Kit", und SE für "Standard Edition".
  Die Unterschiede zwischen den Versionen 6 und 7 der Sprache sind nicht groß und für diese Vorlesung nicht wichtig. Im Moment ist Java SE 7u7 aktuell ("u7" steht für "Update 7"). Das JDK enthält das JRE (Java Runtime Environment), das zur Ausführung von Java-Programmen nötig ist. Das JRE ist auf vielen Rechnern schon installiert (spätestens mit dem ersten Java-Programm, das Sie installieren). Zur Programmentwicklung brauchen Sie die zusätzlichen Bestandteile des JDK (insbesondere den Compiler).

Rechnernutzung, Software (4)

Falls Sie einen eigenen PC haben, Forts.:

- Nach der Installation unter Windows müssen Sie den Suchpfad für Programme so erweitern, das die Programme `java` und `javac` auch gefunden werden.

  Öffnen Sie die Systemsteuerung (Control Panel), System und Sicherheit, System, Erweiterte Systemeinstellungen (Change settings), Erweitert (Advanced), Umgebungsvariablen (Environment Variables).
  Dort die Variable "`Path`" ändern und an den aktuellen Wert nach einem Semikolon das bin-Verzeichnis der JDK-Installation anhängen (z.B. `C:\Program Files\Java\jdk1.7.0_07\bin`).

- Anschliessend starten Sie unter "Zubehör" die "Eingabeaufforderung" ("Command Prompt") und geben z.B. "`java -version`" und "`javac`" ein.

  Nach der Änderung von "`Path`" die Eingabeaufforderung neu starten.

Rechnernutzung, Software (5)

Falls Sie einen eigenen PC haben, Forts.:

- Sie brauchen außerdem einen Editor.

  Ein Editor ist ein Programm zum Eingeben und Ändern von Text-Dateien, z.B. Java-Programmen.

- `notepad` wäre möglich, ist aber wenig komfortabel.

  Word ist nicht für Programm-Dokumente gedacht!

- `notepad++` wird häufig empfohlen (für Windows).

  [http://notepad-plus-plus.org/]

- Bei Linux sind ausreichende Editoren schon dabei.

  Z.B. `gedit`.

- Weitere (mächtige) Editoren sind z.B. `gvim`, `emacs`.

  [http://www.vim.org/], [http://www.gnu.org/software/emacs/]

Rechnernutzung, Software (6)

- Es gibt aber auch umfangreiche Programmpakete mit vielen weiteren Werkzeugen zur Programmentwicklung, sogenannte IDEs (Integrated Development Environment).
  - Eclipse [http://www.eclipse.org/] ist sehr bekannt.
  - NetBeans [http://netbeans.org/] kommt von Sun/Oracle.
  - BlueJ [http://www.bluej.org/] gilt als einfacher und wird gern in Schulen verwendet.

Selbstverständlich werden Sie früher oder später mit einer IDE arbeiten wollen (u.a. wegen der graphischen Oberfläche). Dem in der Klausur verlangten "Programmieren auf Papier" würde aber ein gewöhnlicher Editor noch am nächsten kommen. Die IDE erkennt Syntaxfehler während Sie tippen und bietet automatische Vervollständigungen/Korrekturen an. Das ist nützlich, aber Sie sollten davon nicht abhängig werden.
Die Vielzahl der Funktionen überfordert den Anfänger eventuell auch.

Lehrbücher für Anfänger

- Peter Pepper:
  Programmieren lernen: Eine grundlegende Einführung mit Java, 3. Auflage.

  Springer, Sept. 2007, ISBN: 3-540-72363-3, ISBN-13: 978-3-540-72363-9, 604 Seiten, 24.95 €.

- Hans-Peter Habelitz:
  Programmieren lernen mit Java.

  Galileo Computing, Mai 2012, mit DVD, ISBN: 3-8362-1788-0, ISBN-13: 978-3-8362-1788-0, 512 Seiten, 19.90 €.

- Dirk Louis, Peter Müller:
  Jetzt lerne ich Java 7.

  Markt+Technik Verlag, Okt. 2011, ISBN: 3-8272-4439-0, ISBN-13: 978-3-8272-4439-0, 464 Seiten, mit DVD, 19.95 €.

RRZN-Handbücher

- Dr. Thomas Krökertskothen, Dr. Peter Heusch: Java (1. Band).

  RRZN, ca. 230 Seiten, ca. 5.00 €.

  [http://www.rrzn.uni-hannover.de/buch.html?&no_cache=1&titel=java6]

  Im normalen Buchhandel nicht zu haben (wegen öffentlicher Förderung), nur über das Rechenzentrum zu beziehen.

  Wir haben eine Sammelbestellung über 100 Exemplare ausgelöst. Sie werden in der Übung verkauft, sobald sie eingetroffen sind.

- Fortgeschrittene Techniken und APIs (2. Band)

  U.a. Graphische Benutzeroberflächen, Netzwerkprogrammierung, Threads.

  Ca. 174 Seiten. Ca. 5.00 €. Wir haben 50 Exemplare bestellt.

- Es gibt viele weitere RRZN Handbücher, z.B. zu C, C++, C#, Unix (gut und sehr preiswert).

  [http://www.urz.uni-halle.de/dienstleistungen/handbuecher/]

Lehrbücher für Fortgeschrittene (1)

- Guido Krüger, Heiko Hansen:
  Handbuch der Java-Programmierung: Standard Edition Version 7.

  Addison Wesley/Pearson, 2011, ISBN 3-8273-2751-2,
  ISBN-13: 978-3-8273-2751-2, 1408 Seiten, mit DVD, 49.80 €.
  HTML-Version kostenlos: [http://www.javabuch.de/download.html]

- Christian Ullenboom:
  Java ist auch eine Insel: Das umfassende Handbuch, 10. Auflage.

  Galileo Computing, 2011, ISBN 3-8362-1802-X,
  ISBN-13: 978-3-8362-1802-3, 1308 Seiten, 49.90 €.
  [http://openbook.galileocomputing.de/javainsel/]
  Zweiter Band: [http://openbook.galileocomputing.de/java7/]

Lehrbücher für Fortgeschrittene (2)

- Ken Arnold, James Gosling, David Holmes:
  The Java Programming Language, 4th Edition.

  Addison Wesley, 2005, ISBN 0-321-34980-6, 891 pages, ca. 50.90 €. (Neue Auflage soll April 2013 erscheinen.)

- Cay S. Horstmann, Gary Cornell:
  Core Java, Volume 1: Fundamentals, Eighth Edition.

  Prentice Hall / Sun Microsystems Press, ISBN: 0-13-235476-4, Sept. 2007, 862 Seiten, ca. 48.99 €.
  (Neue Auflage soll im November 2012 erscheinen)

- Joshua Bloch:
  Effective Java, Second Edition.

  Addison Wesley, 2008, ISBN 0-321-35668-3, ISBN-13: 978-0-321-35668-0, 384 pages, ca. 36.95 €.

Spezialthemen

- James Gosling, Bill Joy, Guy Steele, Gilad Bracha:
  The Java Language Specification, Third Edition.

  Addison Wesley, 2005, ISBN 0-321-24678-0, ISBN-13: 978-0-321-24678-3, 651 pages, ca. 48.00 €. Kostenlos online: [http://docs.oracle.com/javase/specs/]

- Arnd Poetzsch-Heffter: Konzepte objektorientierter Programmierung (Mit einer Einführung in Java).

  Springer, 2009, ISBN 3540894705, 352 Seiten, 29,95 €.

- Kathy Sierra, Bert Bates:
  Sun Certified Programmer for Java 6 Study Guide.

  McGraw-Hill/Osborne, 2008, ISBN-13: 978-0-07-159106-5, 851 Seiten, mit CD-ROM, ca. 34.95 €.

- [http://haweb1.bibliothek.uni-halle.de/]

Online-Dokumentation

- Dokumentation zu Java SE 7:
  [http://docs.oracle.com/javase/]
- Insbesondere die API ("Application Program Interface")
  Dokumentation ist fast unverzichtbar:
  [http://docs.oracle.com/javase/7/docs/api/]

  Sie dokumentiert die vordefinierten Klassen, also Programmcode von den Java-Entwicklern, den Sie in Ihren Programmen aufrufen können.
- Java SE 6: [http://docs.oracle.com/javase/6/docs/]
- Online Tutorials:
  [http://docs.oracle.com/javase/tutorial/]
- Spezifikationen:
  [http://docs.oracle.com/javase/specs/]

Inhalt

Vorkenntnisse (1)

- Diese Vorlesung wird von Studierenden mit sehr unterschiedlichen Vorkenntnissen besucht.
  - Einige können schon programmieren, ggf. sogar in Java.

    Bedenken Sie aber bitte, dass diese Vorlesung auch Konzepte vermitteln soll, die in einer rein praktischen Ausbildung fehlen. Vorkenntnisse sind nicht nur positiv: Man kann den Punkt verpassen, wo es dann doch neu wird.

  - Für andere Teilnehmer ist die Programmierung ein völliges Neuland.

    Wir haben leider keine Lehrkapazität für zwei getrennte Vorlesungen. "Programmieren können" ist auch ein weiter Bereich.

- Mein Anspruch ist, dass auch die zweite Gruppe der Vorlesung folgen kann: Ich möchte den Stoff "von Grund auf" logisch und vollständig präsentieren.

Vorkenntnisse (2)

- Stellen Sie Fragen, wenn Sie etwas nicht verstehen, gerne auch während der Vorlesung.

  Wer schon so lange programmiert wie ich, übersieht manchmal, dass bestimmte Dinge nicht selbstverständlich sind. Ich bin dankbar für Hinweise, die es mir erlauben, mein Skript (Foliensatz) noch zu verbessern. Andere Studierende werden Ihnen auch dankbar sein.

- Lassen Sie sich nicht einschüchtern, wenn andere mehr als Sie wissen (ich lerne Java auch gerade erst).

  Offiziell fordert diese Vorlesung keine Vorkenntnisse, es ist also ok, wenn Sie keine haben. Es ist Ihr gutes Recht, zu fragen. Es ist auch völlig unklar, wie es gegen Ende der Vorlesung mit dem Wissen aussieht: Manche Programmier-Neulinge können sich gut in Maschinen und Formalismen hineindenken und lernen schnell. Mancher Teilnehmer mit Vorwissen hat über viele Details noch nie nachgedacht und wüßte die auch nicht.

Zeitliche Belastung (1)

- Für dieses Modul gibt es 5 Leistungspunkte (LP).

  Pro Semester soll man 30 LP erwerben, so dass man nach 6 Semestern den Bachelor-Grad (180 LP) erworben hat. Jeder LP entspricht einer durchschnittlichen Stundenbelastung von 25–30 Stunden, insgesamt also 125–150 Stunden (zum Teil in vorlesungsfreier Zeit zu leisten).

- Aufteilung von 150 Stunden Arbeitszeit:

Lernform	SWS	Stunden
Vorlesung	2	30
Praktische Übung	2	30
Hausaufgaben/direkte Nacharbeit	0	30
Selbststudium	0	45
Spezielle Prüfungsvorbereitung	0	15

Zeitliche Belastung (2)

- Die 150 Stunden sind nur ein Durchschnittswert, in den auch Studierende eingehen, die schon Programmierkenntnisse von der Schule mitbringen.

  Wenn Sie keine Vorkenntnisse haben, müssen Sie also vermutlich mit etwas mehr Zeit rechnen.

- Vorlesungen gehen deutlich schneller vorwärts als Schulunterricht: Es gibt nur wenig Wiederholung.

  Für die Wiederholung müssen Sie selbst sorgen, indem Sie sich zu Hause hinsetzen, die Folien und Ihre Notizen noch einmal durchgehen, darüber nachdenken, weitere Literatur (Bücher, Internet) studieren, die Hausaufgaben machen, und sich vielleicht noch eigene Programmierprojekte vornehmen. Das kann auch Spaß machen!

Zeitliche Belastung (3)

- Praktische Fähigkeiten wie die Programmierung erwirbt man neben dem
  - notwendigen theoretischen Wissen auch durch
  - praktisches Tun und Lernen aus Fehlern.
- Oft können auch Lösungen, die im Prinzip korrekt sind (funktionieren), noch verbessert werden.

  Z.B. einfacher, kürzer, übersichtlicher, leichter änderbar, resourcen-sparender (schneller/weniger CPU-Zeit oder weniger Speicherplatz). Nutzen Sie die Übung zum Austausch mit anderen Studierenden und dem Tutor (natürlich möglichst nach Abgabe der Hausaufgaben).
- Wie viel Sie lernen, hängt an der investierten Zeit.

Zeitliche Belastung (4)

- Wenn praktische Fähigkeiten auch notwendig sind, ist dies doch eine Vorlesung an einer Universität:
  - Man darf über Java auch kritisch nachdenken.

    Mir gefällt auch nicht alles in Java. Überlegen Sie sich, was Sie als Programmiersprachen-Entwerfer anders machen würden.
  - Ziel ist auch ein möglichst leichter Übergang zu anderen Sprachen (allgemeine Konzepte!).
  - Auswendiglernen sollte nicht nötig sein.

    Wichtige Dinge prägen sich bei ausreichender theoretischer und praktischer Beschäftigung mit der Programmierung automatisch ein, selten benötigte Details kann man bei Bedarf nachschlagen.
  - Ihre Mitwirkung (Fragen, Vorschläge) ist wichtig!

Vorlesungs-Etikette

- Vermeiden Sie Verhalten, das Ihre Mitstudenten oder den Professor ablenkt:
  - Vermeiden Sie Gespräche während der Vorlesung.

    Glauben Sie nicht, dass Sie in der Masse untergehen: Der Professor kann durchaus sehen, wer sich unterhält. Wenn Sie Ihren Nachbarn etwas zur Vorlesung fragen müssen, machen Sie es leise und kurz. Wenn die Frage möglicherweise auch für andere interessant ist, stellen Sie sie offiziell (melden, ggf. rufen).
  - Wenn Sie zu spät kommen oder früher gehen müssen, setzen Sie sich möglichst an den Rand.
  - Notebooks sollten währen der Vorlesung nur die Folien anzeigen (eventuell Editor, Compiler).

Ratschläge zum Studium (1)

- Professoren freuen sich über Fragen!

  Selbst wenn der Professor die Frage nicht (gleich) beantworten kann, zeugt sie von Interesse, und am Ende lernen alle etwas davon.

- Lesen Sie möglichst mehrere Bücher zum Thema der Vorlesung, nicht nur das Skript.

  Seien Sie selbständig und etwas kritisch. Man kann Dinge auch ganz ohne Vorlesung lernen. Glauben Sie nicht etwas, nur weil Sie es zufällig gehört haben. Versuchen Sie zu verstehen, nicht auswendig zu lernen.

- Nehmen Sie das Studium ernst.

  Soviel Zeit, wie Sie jetzt haben, sich fortzubilden und Bücher zu lesen, haben Sie später nie wieder. Versuchen Sie, mit 6/10 Semestern auszukommen, wenn es keine besonderen Gründe gibt.

Ratschläge zum Studium (2)

- Lernen Sie programmieren (und wirklich gut).
  Es ist das Handwerk, auf dem alles beruht.
- Lernen Sie mathematisches Denken (Formalisieren, Beweisen), und Techniken wie z.B. vollständige Induktion, Grundlagen der Algebra und Logik.
- Arbeiten Sie in Gruppen zusammen, aber sorgen Sie dafür, dass jedes Gruppenmitglied alles lernt.
- Computerspiele können süchtig machen.
- Falls Frage nach Sinn des Lebens: Ich bin Christ.