Theaterprojekt Kulturhauptstadt Ruhrgebiet 2010

Programmierpraktikum 2010-2011

2. Oktober 2010

Zusammenfassung

Ziel dieses Projektes ist, die Theaterlandschaft im Ruhrgebiet und insbesondere das Stadttheater Hagen durch ein Computerprogramm zur Planung von Theateraufführungen zu unterstützen. Gerade die Stadt Hagen leidet besonders unter der schlechten finanziellen Ausstattung der Kommunen des Ruhrgebietes. Von daher soll dieses Projekt mit dem Namen 'The Show must go on' = ShowGO Hoffung für die Kulturtreibenden in dieser Region geben.

Willkommen in der Firma Propra2010

Mit Ihrer Anmeldung zum Programmierpraktikum sind Sie für dieses Semester Mitarbeiterin oder Mitarbeiter in der Firma Propra2010 geworden. Ihren Unterlagen haben wir entnommen, dass Sie sich intensiv mit der Programmiersprache Java beschäftigt haben und sich weiter beschäftigen möchten. Und dazu werden Sie jetzt bei uns ausreichend Gelegenheit haben.
Zunächst möchte ich Ihnen unsere Firma vorstellen, danach den Kunden des zu erstellenden Programms.
Die festen Mitglieder unserer Firma sind (in der Reihenfolge ihrer Büros)

- Aufsichtsrat: Friedrich Steimann
- Firmenkontakte und Ablaufplanung: Jennifer Schmidt
- Geschäftsführung: Andreas Thies
- Management: Daniela Keller
- Chefprogrammierer: Christian Kollee
- Chefprogrammiererin: Andrea Frank

Frau Andrea Frank managt die verschiedenen Propra-Firmen und ist Ihre Ansprechpartnerin in der Newsgroup

feu.informatik.kurs.1580+82+84.diskussion.ws

sowohl für Java-Probleme als auch Organisatorisches. Wir anderen werden uns aber auch an passender Stelle zu Worte melden.

Da wir uns alle leider nicht regelmäßig treffen können, ist es ganz wichtig, dass Sie mindestens einmal pro Woche in die Betreuungsnewsgroup

feu.informatik.kurs.1580+82+84.betreuung.ws

schauen. Dort werden weitere Informationen zum organisatorischen Ablauf gegeben wie z.B. Informationen zur Präsenzphase, zu Testbeispielen etc.

Wichtig für Sie ist aber zuallererst Ihr Kunde:
Begeistert vom Programm der Kulturhauptstadt Ruhrgebiet beschließt die Kunstfreundin Karina Engel, die Theaterlandschaft des Ruhrgebietes zu unterstützen. Zunächst beabsichtigt sie, einfach eine Spende an das Stadttheater Hagen zu geben. Da sie aber auch an der FernUniversität in Hagen als Akademiestudierende eingeschrieben ist und neben verschiedenen Kursen aus dem Bereich Geschichte, Soziologie und Management sich jetzt auch mit dem Kurs Einführung in die objektorientierte Programmierung beschäftigt hat, erscheint es ihr sinnvoller, ein Computerprogramm zur Unterstützung der konkreten Theaterarbeit programmieren zu lassen. Deshalb hat sie sich an unsere Firma Propra 2010 gewandt. Ihre Vorstellungen hat sie in dem folgenden Pflichtenheft zusammengefasst und hofft jetzt auf leicht bedienbare, gut strukturierte Programmlösungen, die gegebenenfalls auch die Dinge beachten, die sie bei der Erstellung des Pflichtenheftes übersehen hat.

Wir wünschen Ihnen viel Erfolg und Freude bei der Arbeit!
Für das ganze Team
Ihre Daniela Keller (Email-Adresse: daniela.keller@fernuni-hagen.de)

Terminplan und wichtige Infos

- Offizieller Arbeitsbeginn: 05. Oktober 2010
- Abgabe des fertigen und getesteten Programms:
  Die Programme müssen bis Montag, 03.01.2011, 23.59 Uhr, bei uns eingegangen sein, da unsere Kundin am Dienstag gegen 10 Uhr zur Besprechung des weiteren Vorgehens vorbeischaut.
  Planen Sie daher die typischen Pannen wie ein zusammengebrochenes Internet, einen defekten Computer, einen missgelaunten FernUni-Server und sonstige Katastrophen mit ein!
- Sichtung der Ergebnisse durch unser Propra2010-Team: ab dem 05.01., Dauer: circa drei Wochen.
- Korrekturphase: Sie beginnt, nachdem Sie die vorläufige Bewertung Ihrer Arbeit erhalten haben - gegebenenfalls müssen Sie noch nachbessern, also halten Sie sich ab Mitte Januar noch Zeit frei.
- Treffen aller Programmierenden am 19./20.02.2011 in Hagen, um die fertigen Produkte vorzustellen und die besten Programme zu finden.
  - Das Ganze hat eher informellen Charakter, d.h. Sie brauchen keinen Vortrag mit Powerpoint oder Ähnlichem vorzubereiten. Es genügt, wenn Sie Ihr Programm perfekt kennen und uns auch noch die versteckteste Variable erklären können.
  - Es ist recht wahrscheinlich, dass zu diesem Zeitpunkt noch kleinere Sonderwünsche der Kundin anfallen. Diese Erweiterungen Ihres Programms nehmen Sie dann direkt bei unserem Treffen vor.
  - Sie müssen nur an einem der Termine anwesend sein. Die genaue Terminplanung für diese 'Präsenzphase' erfolgt nach der Sichtung der fertigen Programme.
    Bitte Terminprobleme wie Klausuren, Hochzeiten und Arbeitseinsätze für andere Firmen rechtzeitig bekannt geben, damit wir eine für alle passende Lösung finden können.

Wann erhalten Sie Ihr Gehalt?

Um Ihr Gehalt in Form eines (unbenoteten) Leistungsnachweises zu erhalten, erwarten wir von Ihnen:

1. Eigenständige Implementierung eines fehlerfreien Programms gemäß dem nachfolgenden Pflichtenheft.
   Wichtig: Gruppenlösungen sind nicht zulässig.
2. Abgabe des Programms bis zum 03.01.2011 unter Berücksichtigung der von uns angegebenen Programmier- und Dokumentationsrichtlinien.
   Genauere Informationen zu den Abgabemodalitäten erhalten Sie von uns per Email und über die Newsgroup.
3. Unsere Kundin erwartet, dass zum Programm die folgenden Komponenten gehören:

- das compilierte Programm als .jar-Datei, welches sich durch ein einfaches
  java -jar IHR_NAME.jar
  ausführen und testen lassen muss,
- der Quelltext als .zip-Datei mit dem Namen **IHR_NAME.zip**.
  Innerhalb der zip-Datei soll der Quelltext den Packages entsprechend in Unterverzeichnissen organisiert sein,
- die Dokumentation (das Handbuch).

Es ist ganz wichtig für uns, dass Sie hier keine Fehler machen, denn oft wird die Arbeit einer Firma auch schon danach bewertet, ob alles so verpackt ist, wie es der Kunde erwartet. Wenn wir hier gute Arbeit leisten, schaut sich unsere Kundin unser Programm mit viel mehr Freude und Zufriedenheit an.

4. Diesen Punkt schreiben wir nur hinzu, weil unsere Rechtsabteilung darauf besteht. Wir selbst sind sicher, nur verantwortungsvolle Programmiererinnen und Programmierer zu haben.

   'Manipulationsversuche mit den eingesandten Programmen, z.B. durch Aufspielen von Computerviren oder anderen Programmkomponenten, die nicht der Lösung der Programmieraufgabe dienen, führen - unabhängig von Schadensersatzansprüchen seitens der FernUniversität - zu einer Nichterteilung des Leistungsnachweises.'

   'Das Kopieren von Sourcecode aus gleichen oder ähnlichen Projekten aus dem Internet ist unter keinen Umständen gestattet und wird mit einem sofortigen Ausschluss geahndet.'

Programmier- und Dokumentationsrichtlinien

Neben dem eigentlichen Programm werden wir uns auch ausführlich mit Ihrer Dokumentation beschäftigen. Sie sollten daher bei der Gestaltung Ihrer Programmdokumentation die gleichen Maßstäbe ansetzen, nach denen Sie beispielsweise eine schriftliche Seminarausarbeitung erstellen würden.
Halten Sie sich bei der Erstellung Ihrer Dokumentation bitte an die folgenden Richtlinien:

Ihre Dokumentation besteht aus einer kurzen Einführung und dem kommentierten Programmcode.

- Überlegen Sie sich einen **einheitlichen Kommentarkopf mit Javadoc-Elementen**, den Sie vor jeder Klasse und Methode einfügen. In diesem Kommentarkopf beschreiben Sie den Zweck der verwendeten Parameter, welche Aufgabe von der betreffenden Klasse bzw. Methode erfüllt wird, sowie die Einordnung der Klasse/Methode in den Gesamtkontext des Programmes. Vermeiden Sie es, in dem Kommentarkopf ganze 'Romane' zu schreiben; je zwei bis drei prägnante Sätze zur Aufgabe und Einordnung der Methode bzw. Klasse sagen mehr als eine halbe Seite Erläuterungstexte. Benutzen Sie das **Javadoc-Werkzeug**. Beschreiben Sie damit die Parameter und Rückgabewerte.
- **Kommentieren Sie Ihren Programmtext.**
  Das soll nicht heißen, dass Sie zu jeder Anweisung einen Kommentar schreiben müssen,

aber Ihr Programm muss mit Hilfe der Kommentare soweit verständlich sein, dass der Leser Ihre Lösung ohne ein langwieriges Hineindenken in Ihre Java-Konstrukte nachvollziehen kann. Bedenken Sie dabei auch, dass der Leser im Gegensatz zu Ihnen nicht mit den von Ihnen eingeführten Datenstrukturen und Funktionen vertraut ist. Ersparen Sie ihm daher ein Nachblättern der betreffenden Datentypen oder Funktionen, indem Sie bei Wertzuweisungen stichwortartig erklären, was dort bezweckt/ausgeführt wird, wenn dies nicht offensichtlich ist.
Schließlich noch ein Hinweis: Wenn Sie bemerken, dass innerhalb einer Methode die Notwendigkeit auftritt, mehrere Sätze zur Erläuterung eines Programmabschnittes zu schreiben, dann ist dies ein Anzeichen dafür, dass Ihr Programm noch nicht genügend modularisiert ist. In diesem Fall sollten Sie erwägen, die betreffenden Programmteile als eigenständige Methode auszugliedern. Auf keinen Fall dürfen Sie aber das Problem so 'lösen', dass Sie den betreffenden Teil nicht oder nur unzureichend kommentieren!

- Verwenden Sie **aussagekräftige Bezeichner** für Ihre Variablen, Klassen und Methoden: Ein gut gewählter Bezeichner ist so kurz wie möglich, aber lang genug, um seine Funktion verständlich zu beschreiben. Abkürzungen sind erlaubt, sollten aber 'entschlüsselbar' sein. Geben Sie Abkürzungen aus dem normalen Sprachgebrauch den Vorzug vor Eigenkonstrukten (also z.B. 'nachf' für 'Nachfolger' und nicht 'nfolg'.) Achten Sie auch darauf, dass Abkürzungen aussprechbar bleiben (z.B. 'elem' für 'Element' und nicht 'elmt').

- Benutzen Sie ein einheitliches Vorgehen bei der Gliederung von Deklarationen und der Einrückung von Programmteilen.

- Vermeiden Sie es, mehr als eine Anweisung in eine Zeile zu schreiben.

Hinweise zum Testen des Programms

Das Testen von Programmen ist ein sehr umfangreiches Fachgebiet innerhalb der Informatik, das Stoff genug enthält, um eine ganze Serie von Vorlesungen oder Kurseinheiten zu füllen. Wir wollen Ihnen hier nur einige Denkanstöße aus diesem Gebiet liefern, um Ihnen das Testen und damit das Abliefern einer (fast, s.u.) korrekten Lösung zu erleichtern.

1. Untersuchungen haben ergeben, dass ein Programm in der Realität niemals fehlerfrei ist. Für ein 'frisch programmiertes' Programm (also vor der Testphase) ist es realistisch anzunehmen, dass auf 100 Zeilen Code ca. 4 bis 8 Fehler kommen!

2. Für nicht triviale Programme ist es aus Komplexitätsgründen nicht möglich, einen lückenlosen Korrektheitsbeweis zu führen, d.h. es ist für jedes reale Programm effektiv nicht beweisbar, dass es korrekt ist.

Daraus folgt, dass der Sinn des Testens eines Programms nicht darin bestehen kann, die völlige Fehlerfreiheit eines Programms nachzuweisen, da dies nach 1. extrem unwahrscheinlich und nach 2. objektiv ohnehin nicht beweisbar ist. Daher definiert man das Testen häufig wie folgt:

Testen bedeutet, ein Programm mit der Absicht auszuführen, Fehler zu finden.
Eine Testeingabe wird als erfolgreich bezeichnet, wenn sie das Programm zu falschem Verhalten verleitet (fehlerhafte Ausgabe, Programmabbruch etc.). Hingegen betrachtet man eine

Testeingabe als nicht erfolgreich, wenn sich das Programm korrekt verhält (korrekte Ausgabe oder Zurückweisung einer Eingabe, die außerhalb des gültigen Bereiches liegt).
Die Teststrategie besteht also darin, gezielt möglichst viele Fehler in dem Programm zu finden. Damit kann man zwar nicht beweisen, dass ein Programm überhaupt keine Fehler mehr enthält (s.o.), das Vertrauen in die Zuverlässigkeit des Programmes wird jedoch mit der steigenden Anzahl nicht erfolgreicher Testfälle (= korrekter Reaktionen des Programmes) und daraufhin eliminierter Fehler erhöht.
Nachfolgend wollen wir Ihnen einige Anregungen geben, wie Sie Ihr Programm effektiv testen können:

- Zunächst einmal: Lassen Sie sich nicht dazu verleiten, Programmfehler als persönliche Fehlleistung aufzufassen (nach dem oben Gesagten sollte klar sein, dass sich Fehler zwangsläufig und unvermeidbar in Programme einschleichen)! Im Testen unerfahrene Programmierer empfinden das Testen häufig als unangenehm, da jeder Fehler als Rückschlag bzw. 'Versagen' aufgefasst wird, und die Konsequenz ist häufig, dass entweder überhaupt nicht oder nur sehr halbherzig (mit korrekten, für das Programm harmlosen Testfällen) getestet wird. Sehen Sie die ganze Sache positiv: Jeder Fehler, den Sie finden und beheben, macht Ihr Programm ein Stück perfekter!

- Wählen Sie Ihre Testeingaben effizient aus. Ein geeignetes Verfahren ist z.B. die **Grenzwertanalyse**. Dabei ermitteln Sie zunächst für einen Eingabewert alle gültigen und ungültigen Werte, und wählen dann jeweils einen Repräsentanten aus, der gerade noch gültig ist ('auf der Grenze liegt') und je einen Repräsentanten, der knapp außerhalb der Grenze liegt und damit ungültig ist.
  Wenn Sie z.B. eine Funktion testen, die einen Text mit einer Länge von 1...40 Zeichen als Eingabe bekommt, würden Sie nach der Grenzwertmethode jeweils einen Text der Länge 1 und einen Text der Länge 40 als gültige Eingabe sowie Texte der Längen 0 bzw. 41 als ungültige Werte auswählen.
  Die ungültigen Werte nimmt man in die Testmenge auf, um zu sehen, ob das Programm auch auf fehlerhafte Eingaben sinnvoll reagiert (indem es z.B. eine Warnung ausgibt o.ä). Falls solche Testfälle nicht vorgesehen werden, kann es vorkommen, dass zum Beispiel ein Programm in inneren Modulen später aus 'unerklärlichen' Gründen abstürzt (weil im Verlauf der Berechnung intern ein ungültiger Wert erzeugt wurde), oder bei bestimmten Eingaben nur unsinnige (weil undefinierte) Ausgaben erscheinen.

- Eine ähnliche Strategie besteht darin, nach **Spezialfällen** (neutrale Elemente, Definitionslücken wie die berühmte Division durch Null) Ausschau zu halten. Arbeitet ein Sortieralgorithmus z.B. auch korrekt, wenn die Liste der zu sortierenden Worte leer, einelementig oder bereits sortiert ist?

- Versuchen Sie, eine Menge von Testeingaben so zu wählen, dass insgesamt alle Programmteile in möglichst allen Kombinationen durchlaufen werden.

- Testen Sie Ihr Programm klassenweise. Dies bedeutet, dass Sie die zu testende Klasse direkt mit passenden Testwerten aufrufen und die zurückgegebenen Werte überprüfen. Eine in das Gesamtprogramm integrierte Klasse lässt sich nicht mehr zielgerichtet testen, da die Eingabe des Hauptprogrammes auf dem 'Aufrufweg' zur Zielklasse oft so stark transformiert wird, dass man für die Klasse keine individuellen Testwerte mehr

erzeugen kann. Gleiches gilt natürlich für die Ausgabe des Moduls, die bis zur endgültigen (Bildschirm-)ausgabe im kompletten Programm so weit umtransformiert werden kann, dass sie zu dem betreffenden Modul nicht mehr eindeutig in Beziehung gesetzt werden kann.