Klausur Softwaretechnik

15.03.2007

Prof. Dr. Walter F. Tichy
Dipl.-Inform. T. Gelhausen
Dipl.-Inform. G. Malpohl

Hier das Namensschild aufkleben.

Zur Klausur sind keine Hilfsmittel und kein eigenes Papier zugelassen.
Die Bearbeitungszeit beträgt 60 Minuten.
Die Klausur ist vollständig und geheftet abzugeben.
Mit Bleistift oder roter Farbe geschriebene Angaben werden nicht bewertet.

Aufgabe	**1**	**2**	**3**	**4**	**5**	**∑**
Maximal	18	12	13	10	7	**60**
K1						
K2						
K3						

Aufgabe 1: Aufwärmen (18P)

a.) Kreuzen Sie an, ob die Aussage wahr oder falsch ist. (8P)
Hinweis: Jedes korrekte Kreuz zählt 0,5 Punkte, jedes falsche Kreuz bewirkt 0,5 Punkte Abzug! Die Teilaufgabe wird mit mindestens 0 Punkten bewertet.

Wahr	Falsch	Aussage
		Software ist leichter zu ändern als ein physikalisches Produkt vergleichbarer Komplexität.
		Das Lastenheft beschreibt die Eigenschaften, die das Produkt aus Sicht des Kunden erfüllen soll, während das Pflichtenheft zusätzlich die genauen Vorschriften für den Entwickler enthält.
		Bei der Anforderungsvalidierung in der Planungsphase wird nicht überprüft, ob die Anforderungen erfüllt werden können.
		Eine vollständige Methodensignatur in UML besteht aus dem Rückgabetyp und der Parameterliste einer Funktion.
		Klassendiagramme sind Multi-Hyper-Graphen.
		Wird das Attribut „Multiplizität“ einer Assoziation in einem UML-Klassendiagramm nicht angegeben, bedeutet dies „nicht spezifiziert“ und damit spezifikationsgemäß „beliebig viele, aber mindestens 1“.
		In Java muss eine Klasse, die eine Schnittstelle implementiert, alle in der Schnittstelle vorgegebenen Methoden implementieren.
		Auf ein `private`-Attribut eines Exemplars einer Klasse kann nur das Exemplar selber zugreifen.
		In UML-Aktivitätsdiagrammen unterscheidet man zwischen Objekt- und Kontrollflüssen.
		Die Diagramme „Kollaborationsdiagramm“, „Zeitdiagramm“, „Zustandsdiagramm“ und „Sequenzdiagramm“ zählen zur Gruppe der „Interaktionsdiagramme“.
		Bei Modulen gibt es zu jeder Definitionseinheit nicht mehr als eine Implementierungseinheit.
		Das Entwurfsmuster „Iterator“ ermöglicht „polymorphe Iteration“.
		Kombiniert man das Entwurfsmuster „Befehl“ mit dem Entwurfsmuster „Stellvertreter“, erhält man einen Makrobefehl.
		Die Hauptaufgaben der Implementierungsphase des Wasserfallmodells sind die Umsetzung der Spezifikation in korrekte, ablauffähige Programme, die Dokumentation und das Testen der Komponenten.
		Bei der Feinoptimierung ändern sich lediglich Konstanten des Laufzeitverhaltens, die asymptotische Laufzeit des Algorithmus bleibt unverändert.
		Beim Planungsspiel im XP gilt: „Das Entwicklungsteam trifft die geschäftsrelevanten Entscheidungen.“

b.) Definieren Sie die Begriffe „Signaturvererbung“ und „Implementierungsvererbung“ aus der objektorientierten Programmierung. In welchem Zusammenhang stehen diese beiden Eigenschaften? (3P)

c.) Was ist der Nachteil von Zyklen in der Benutzt-Relation zwischen Modulen? (1P)

d.) Definieren Sie die zwei Begriffe „Wartung“ und „Pflege“ von Softwareprodukten. (1P)

e.) Nennen sie jeweils zwei Qualitätssicherungsverfahren aus der Kategorie der dynamischen und der statischen Verfahren. (2P)

Dynamische Verfahren:

1. _____ .

2. _____ .

Statische Verfahren:

1. _____ .

2. _____ .

f.) In der objektorientierten Analyse und in UML gibt es das Konzept der Assoziation. (i.) Was bietet dieses Konzept für einen semantischen Mehrwert gegenüber den einfachen Referenzen, wie sie z.B. in Java zu finden sind? (ii.) Welche Probleme hat man ohne dieses Konzept der Assoziation? (iii.) Java kennt keine Assoziationen. Wie setzt man also Assoziationen in Java um? (3P)

Aufgabe 2: Anforderungserhebung (12P)

Ihr Chef hat beim Golf spielen einen Auftrag an Land gezogen: Ihre Firma (insges. 15 Mitarbeiter) soll den Webauftritt eines mittelständigen Reiseunternehmens komplett neu gestalten. Das Unternehmen hat bisher nur einige wenige statische Seiten und möchte gerne „einen komfortablen, interaktiven Reisekatalog“ anbieten. Für den Vertragsabschluss sollen Sie nun die Anforderungen erheben. Ihr Chef gibt Ihnen eine Woche Zeit.

a) Welche Erhebungstechniken haben wir in der Vorlesung kennen gelernt? (4P)

b) Diskutieren Sie für jede einzelne Technik, ob und warum (oder warum nicht) sie sich **im beschriebenen Szenario** anbietet. (8P)

Aufgabe 3: Kontrollflussorientierter Strukturtest (13P)

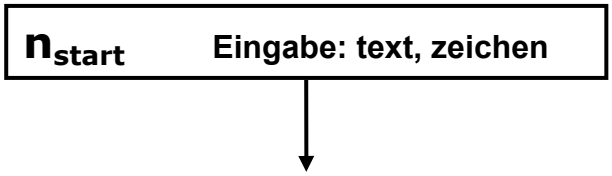
Gegeben ist die folgende Java-Funktion:

```
int zaehleZeichen(String text, char zeichen) {
 int zaehler = 0;
 if (text != null) {
  for (int i=text.length()-1; i>=0; i--) {
   if (text.charAt(i) == zeichen) {
    zaehler++;
   }
  }
 }
 return zaehler;
}
```

a.) Wandeln Sie obiges Programm mit einer strukturerhaltenden Transformation in eine der Definition der Vorlesung entsprechenden Zwischensprache um. (3P)

Eingabeparameter: text, zeichen

b.) Erstellen Sie den Kontrollflussgraphen der Funktion `zaehleZeichen()`. Wenden Sie dabei das aus der Vorlesung bekannte Verfahren an. (4P)



c.) Nennen Sie eine minimale Testfall-Menge für die Anweisungsüberdeckung und geben Sie zu jedem Testfall den durchlaufenen Pfad an. (1P)

d.) Nennen Sie eine minimale Testfall-Menge für die Zweigüberdeckung des obigen Programms. Geben Sie zu jedem Testfall den durchlaufenen Pfad an. (2P)

e.) Nennen Sie die Teilpfade der Schleifenquerer (Boundary-Interior-Test). (1P)

f.) Geben Sie eine Testfallmenge für den Grenztest (Boundary-Interior-Test) an. (1P)

g.) Geben Sie eine Testfallmenge für den Interieurtest (Boundary-Interior-Test). (1P)

Aufgabe 4: Entwurfsmuster (10P)

Gegeben sei folgendes Programm (`S.o.p` steht für `System.out.println`):

```
import java.util.*;

abstract class Tee {
 protected boolean gezogen = false;
 public abstract void ziehen();
} // Ende der Klasse Tee


class Schwarztee extends Tee {
 public void ziehen() {
  S.o.p("Der Schwarztee zieht.");
  gezogen = true;
 }
} // Ende der Klasse Schwarztee

class GruenerTee extends Tee {
 public void ziehen() {
  S.o.p("Der grüne Tee zieht.");
  gezogen = true;
 }
} // Ende der Klasse GruenerTee

class Chai extends Tee {
 private Tee basisTee;
 private List<String> zutaten
      = new ArrayList<String>();

 public Chai(Tee grundlage) {
  if (grundlage==null) throw new
   IllegalArgumentException();
  basisTee = grundlage;
  zutaten.add("Lorbeerblatt");
  zutaten.add("Zimtstange");
  zutaten.add("Ingwer");
  zutaten.add("Honig");
  zutaten.add("Milch");
  zutaten.add("Vanilleschote");
 }

 public void ziehen() {
  basisTee.ziehen();
  for (String z : zutaten) {
   S.o.p(z + " zieht");
  }
  S.o.p("Chai zieht");
  S.o.p("Tee zieht mit Chai");
  gezogen = true;
 }
} // Ende der Klasse Chai



class Test {
 public static void main
                 (String[] a) {
  Tee t = new Schwarztee();
  Tee chai = new Chai(t);
  chai.ziehen();
 }
} // Ende der Klasse Test
```

a.) Welches Entwurfsmuster wird hier verwendet? („Oberklasse“ ist nicht gemeint!) **Begründen** Sie Ihre Antwort. (2P)

b.) Geben Sie an, welche Klasse welche Rolle des Entwurfsmusters einnimmt. (1P)

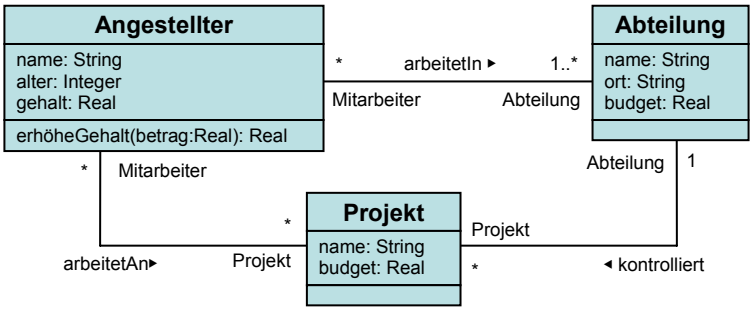
c.) Geben Sie das UML-Klassendiagramm des hinter dem Code stehenden Modells an. Ihr Diagramm soll alle Klassen (außer der Klasse „Test“) mit allen Attributen, Methoden (incl. Sichtbarkeit etc.) und Assoziationen enthalten. (3P)
Hinweis: Achten Sie darauf, Attribute und Assoziationen nicht zu verwechseln!

d.) Wie heißt (i.) das übergeordnete Entwurfsmuster zu dem im angegebenen Programm implementierten? Was ist (ii.) der Unterschied? („Oberklasse“ ist nicht gemeint!) (2P)

e.) Was haben (i.) das im angegebenen Programm implementierte Entwurfsmuster und der „Adapter“ gemeinsam? Und (ii.) wie unterscheiden sich die beiden? (2P)

Aufgabe 5: OCL (7P)

Gegeben seien das nebenstehende UML-Klassendiagramm und der folgende OCL-Ausdruck. In der Vorlesung wurde gezeigt, wie Klassendiagramme in Java umgesetzt werden. Was ist hingegen bei der Umsetzung des gegebenen OCL-Ausdrucks in Java zu tun?



Hinweise: Nennen Sie alle Stellen, an denen Maßnahmen zu treffen sind. Beachten Sie, dass sich die Zusicherung auch auf etwaige Verwaltungsmethoden der Assoziationen auswirken könnte. Sie brauchen in dieser Aufgabe keinen konkreten Code anzugeben, beschreiben Sie die Maßnahmen abstrakt.

```
context Abteilung inv:
 self.Mitarbeiter->forAll(a1, a2 |
  a1.Projekt->size() > a2.Projekt->size()
  implies a1.gehalt > a2.gehalt)
```