Kausur Software-Engineering 2, Medieninformatik, Steppat, SS 2012

Name: _____ **Matr.–Nr.:**_____

letzter Versuch? (ja/nein): nein

38 Punkte 1,7 Ste

Aufgabe 1 - Entwurf von Oberflächen

1.1 Der grafische Entwurf einer Benutzeroberfläche kann in bis zu vier Schritten erfolgen. Wie heißen diese und welchem Zweck dienen sie?

(2 Punkte) 2

1.2 Geben Sie zwei Regeln für die Gruppierung von Steuerelementen in Benutzeroberflächen an.

(2 Punkte) 2

1.3 Wie kann die einheitliche Umsetzung von UI-Elementen in einem größeren Softwaresystem sichergestellt werden?

(2 Punkte) 2

Aufgabe 2 - Entwurf und Entwurfsmuster

2.1 Warum ist das Klassendiagramm zur Beschreibung von Verhaltensmustern nicht ausreichend? Begründen Sie, wozu noch ein weiteres Diagramm eingesetzt werden muss.

(3 Punkte)

2.2 Nennen Sie die Klassen und Interfaces die für ein Fabrikmethodemuster mindestens notwendig sind und beschreiben Sie deren Aufgaben.

(4 Punkte) 4

2.3 Welche Vorteile (mindestens zwei) bietet das Proxymuster?

(2 Punkte) 2

2.4 Wie werden Assoziationsnamen beim Entwurf und bei der Implementierung umgesetzt?

(2 Punkte) 2

Aufgabe 3 - Implementierung

3.1 Ein Installationsprogramm für ein Softwarsystem nimmt die Installation in den Schritten: Überprüfung des Rechners, Kopieren der Dateien, Konfigurieren des Systems vor. Jeder Schritt, welcher einen Zustandsübergang darstellt, wird durch die Methode *weiter()* eingeleitet. Die Installation kann in jedem Zustand mit der Methode *abbruch()* beendet werden. Dabei wird der Ausgangszustand des Rechners wiederhergesellt. Als bereits implementierte Klasse steht Ihnen die Klasse *InstallationsAPI* mit den Methoden: *pruefeHardware(), pruefePlattenkapazitaet(), pruefeBetriebssystem(), entpackeDateien(), kopiereDateien(), loescheInstallierteDateien(), erzeugeKonfigurationsdaten(), loesche Konfigurationsdaten()* zur Verfügung. Verwenden Sie dafür das Zustandsmuster.

(6 Punkte) 6

3.2 Wie wird beim Beobachtermuster die Benachrichtigung aller registrierten Beobachter implementiert? In welcher Klasse befindet sich diese Methode?

(3 Punkte) 3

3.3 Beschreiben Sie die einzelnen Schichten des Model View Controllers und welche Aufgaben sie haben.

(2 Punkte) 2

Aufgabe 4 - Datenbankanbindung

4.1 Welche Ansätze gibt es, eine Generalisierungshierarchie von Klassen auf eine Datenbank abzubilden. Beschreiben Sie diese. (2 Punkte) 2

4.2 Welche beiden Möglichkeiten gibt es, die Textausgabe von JSP-Seiten mit HTML zu formatieren?

(2 Punkte) 2

4.3 Wie werden beim Abbilden von Klassen auf Datenbanken die Assoziationen berücksichtigt. Welche Rolle spielen dabei die Kardinalitäten? (3 Punkte) 3

Aufgabe 5 - Softwaretest

5.1 Was versteht man unter Anweisungs- und Zweigüberdeckung beim Testen einer Methode?

(2 Punkte) 2

5.2 Wie wird eine Testumgebung aufgebaut? Beschreiben Sie, aus welchen Teilen sie besteht und welche Funktion sie haben.

(2 Punkte) 2

5.3 Beschreiben Sie, wie mit JUnit getestet werden kann, dass ein Singleton auch tatsächlich nur eine Instanz hat.

(2 Punkte) 2

5.4 Geben Sie für die untenstehende Methode die Testfälle an, welche die Anweisungsüberdeckung sicherstellen und die Testfälle, welche eine Zweigüberdeckung gewährleisten:

```
public int FehlerBewertung(double sollWert,      // Sollwert
                           double[] messWerte,   // Array mit Messwerten
                           int anzahlMesswerte, // Anzahl der Messwerte) {
   int i = 0;
   int strafpunkte = 0;
   while(i < anzahlMesswerte) {
      if(messWerte[i] / sollWert > 0.1)
         strafpunkte = strafpunkte + 3;
      if(messWerte[i] / sollWert > 0.05)
         strafpunkte = strafpunkte + 1;
      i++;
   }
   return strafpunkte;
}
```

(4 Punkte)

25. Juni 2012

1.1 Der 1. Schritt ist die Erstellung eines Mockups. Hier wird bildlich veranschaulicht wie die Oberfläche gestaltet ist

Der 2. Schritt ist ein Klassendiagramm mit den Grafischen Elementen.
Dient der genaueren Gruppierung und hilft die Anzahl der unterschiedlichen Elemente im Blick zu behalten.

Der 3. Schritt ist die Implementierung durch auf einer HTML-Seite.
Ist die Umsetzung des Entwurfes in HTML-Code und ggf. Anbindung von Funktionalitäten.

Der 0. Schritt ist die Funktionalität der Oberfläche.
Hier wird festgelegt welche Funktionen die entsprechende Seite bereitstellen muss. ✓

1.2 Es soll bei mehreren Überschrift maximal 4-5 Hauptgruppen geben. Jede Gruppe hat maximal 6-7 Elemente. ✓
Desweiteren soll Symmetrie beachtet werden und die Anzahl der virtuellen Linien gering gehalten werden.

1.3 In dem im Vorfeld jedes UI-Element genau definiert wird und in den Umsetzungsrichtlinien für alle festgehalten ist mit den notwendigen Werten.
Desweiteren durch eine gemeinsame Entwicklungsumgebung Bsp. Git. Jederzeit kann auf die Elemente zugegriffen werden und auch durch Vererbung kann die optische Präsens erhalten und die Funktionalität verändert werden. Styleguides ✓

2.1 Ein Klassendiagramm gibt keinen Aufschluss über die Assoziationen zwischen den einzelnen Klassen ✓
Aus diesem Grund ist es wichtig durch Entwurfsdiagramme die Assoziation abzubilden und somit auch die Beziehungen zwischen den Klassen zu verdeutlichen. Je nach Entwurfsdiagramm werden andere Verhalten oder Strukturen genauer betrachtet bzw. veranschaulicht. Sequenzdiagramme

2.2 Es gibt das Interface "Creator" mit den Methoden, die für die Klassen der "ConcreteCreator" alle gleich sind und die dort enthalten sein müssen um ein "Product" zu erstellen. "Product" ist ein Interface, das durch die Klassen der "ConcreteProduct" implementiert wird und ein konkretes Produkt zurückgibt.
ConcreteCreator hat die Aufgabe ein ConcreteProduct zu erzeugen. ConcreteProduct gibt ein bestimmtes Product (Objekt) zurück. Die Interfaces dienen der Übermittlung der Methode, die zur Erzeugung notwendig ist. Sie definieren die Schnitt-

stelle zwischen dem ConcreteCreator und dem Aufruf durch den Client. ✓

2.3 Das Proxy ist eine Verwaltung von Threads und startet bzw. stoppt diese. f
Desweiteren definiert es die Schnittstelle zwischen HTML-Doc. und der Anwendung. Es dient dem Laden von Internetseiten. (dynamisches Nachladen).
Auch wird dadurch eine Lose statt einer festen Kopplung erzeugt. ✓

2.4 Je nach Art der Assoziation wird diese durch entweder keine, eine oder beide Klassen halten einen Verweis auf die entsprechend andere Klasse. Das heißt jede Klasse hat eine Variable in der die ID der anderen gespeichert ist bei einer bidirektionalen Assoziation. Bei unidirektional die Klasse, die die andere kennt und bei unspezifisch keine von beiden, dies gilt auch beim Ausschluss. ✓

3.1
```
public abstract class Zustand {
    public InstallationsAPI iApi;
    public void weiter(){};
    public void abbruch(){};
}

public class PrüfeRechner extends Zustand {
    public PrüfeRechner(InstallationsAPI api){
        iApi = api;
    }
    public void weiter(){
        iApi.pruefeHardware();
        iApi.pruefePlattenkapazität();
        iApi.pruefeBetriebssystem();
    }
    public void abbruch(){};
}

public class KopiereDateien extends Zustand {
    public KopiereDateien(InstallationsAPI api){
        iApi = api;
    }
    public void weiter(){
        iApi.entpackeDateien();
        iApi.kopiereDateien();
    }
    public void abbruch(){
        iApi.loescheInstallierteDateien();
    }
}
```
✓ ✓

4.1 In Form von OIDs, die jeweils der Teil erhält, der erbt. Also der in der Hierarchie am höchsten stehende erhält keine fremde OID und die Nachfolger jeweils nur die des Vorgängers.

Es gibt auch die Möglichkeit durch ein Mapping die Dateien entsprechend auf der Datenbank abzulegen. ✓

4.2 Einmal über ein Auslesen des Textes aus der JSP-Datei und dann durch die Tags <> ausgeben.
Mithilfe eines Servlets, der die JSP-Datei übersetzt.
z.B. out.println("<B> fette Text </B>");

4.3 Je nach Assoziation wird die fremd OID gespeichert bei keiner, einer oder beiden Klassen in der Datenbank. Die Kardinalitäten spielen nur bei n zu m eine Rolle, bei den anderen wird auf der Seite mit der 1 die OID gespeichert. Bei n-m gibt es zwei Möglichkeiten, entweder über eine Assoziationsklasse oder Zeigersammlungen. ✓

5.1 Anweisungsüberdeckung heißt jede Anweisung muss ausgeführt werden, mindestens 1 mal. Im Fall einer if-Anweisung reicht 1 Variante (also if oder else).
Zweigüberdeckung heißt jeder Zweig einer Anweisung muss zusätzlich mindestens 1 mal ausgeführt werden. (also if und else-Anweisungen, alle vollständig). ✓

5.2 Testumgebung besteht aus Assoziationsklassen und Testmethoden. Hier können mehrere Methoden auch zusammengefügt werden.

Assoziationsklassen fassen Testfälle zusammen.
Testmethoden geben mit Hilfe der Junit-Testmethoden Ergebnisse für die getesten Methoden aus.
Testsuiten sind zusammengefasste Testmethoden. ✓

5.3 Durch die assertSame(object o) wird festgestellt, ob es sich beim Erstellen um ein identisches Objekt handelt. Ist dies nicht so schlägt der Test fehl. ✓

5.4 Anweisungsüberdeckung: T-01 anzahlMesswerte = 0
T-02 anzahlMesswerte = 1
Zweigüberdeckung: T-03 anzahlMesswerte = 1
sollWert = 100
T-04 sollWert = 0,1
T-05 sollWert = 10

Beim Test müssen immer alle Parameter angegeben werden:
T-01: FehlerBewertung(1.0, 0.2, 1);

T-02: FehlerBewertung(1.0, 0.1, 0.05, 2);
T-03: FehlerBewertung(1.0, 0.1, 0.05, 2);

```
public class KonfiguriereSystem extends Zustand {
    public KonfiguriereSystem(InstallationsAPI api) {
        iApi = api;
    }
    public void weiter() {
        iApi.erzeugeKonfigurationsdaten();
    }
    public void abbruch() {
        iApi.loescheKonfigurationsdaten();
        iApi.loescheInstallierteDateien();
    }
}

public class Installation {
    public InstallationsAPI api = new InstallationsAPI(this);
    public PrüfeRechner prüfe = new PrüfeRechner(api);
    public KopiereDateien kopiere = new KopiereDateien(api);
    public KonfiguriereSystem konfiguriere = new KonfiguriereSystem(api);
    public Zustand zustand = prüfe;

    public void prüfe() {
        zustand = kopiere;
    }
    public void kopiere() {
        zustand = konfiguriere;
    }
    public void abbruch() {
        zustand.abbruch();
    }

    public void weiter() {
        zustand.weiter();
    }
}
```

3.2 Die Methode notify() in der Klasse Subject benachrichtigt alle beim Subject registrierten Beobachter. Speziell ConcreteSubject.
Implementierung:

```
public notify() {
    for (Observer o : observers)
        o.notify();
}
```

observers ← Liste aller Beobachter

3.3
- Model → hält die Daten
- Controller → benachrichtigt View über Änderungen im Model, ist Vermittler zwischen Model + View
- View → zeigt die Daten an