1. Fragen
   1. Was beschreibt ein UC
      1. Zusammenhang zweier Java-Objekte
      2. Beziehung zweier Akteure
      3. Bez. Akteur und System
   2. Was beschreibt ein Domänenmodell?
      1. Techn. Begriffe und Konzepte
      2. Fachliche Begriffe und Konzepte
      3. Grundlage für die Benutzerschnittstelle
   3. Kann ein zu beschreibendes System ein Akteur sein
      1. Ja
      2. Nein
      3. Kommt drauf an
   4. Welche Anforderung ist funktional?
      1. Erreichbar zwischen 6:00 – 20:00
      2. Nur registrierte Kunden dürfen Bestellungen aufgeben
      3. Verbindung wird über SSL realisiert
   5. Was bildet die Anforderungsanalyse ab
      1. Externe Sicht
      2. Interne fachliche Sicht
      3. Technische Anforderungen
   6. Was ist das Geschäftsprozessmodell?
      1. Statische Systemsicht
      2. Dynamische Systemsicht
      3. Objektorientierte Systemsicht
   7. Wie oft wird die Anforderungsanalyse ausgeführt?
      1. 1 mal
      2. 2 mal
      3. N-mal
   8. Wie ist der englische Fachbegriff für Anforderungsanalye?
      1. Domain-Specification
      2. Conceptual Analysis
      3. Requirement Analysis
   9. Wie wird ein Domänenmodell dargestellt
      1. Sequenzdiagramm
      2. Zustandsdiagramm
      3. Klassendiagramm
   10. Warum müssen nichtfunktionale Anforderungen quantifizierbar sein
       1. Um zu prüfen, ob sie erfüllt sind.
       2. Damit man weiss wie viele vorhanden sind
       3. Damit 2 vergleichbar sind

a) Beschreiben Sie die 4 **Projektphasen** des **Unified Process**. Welcher **Schwerpunkt**

wird in jeder Phase gesetzt?

b) Beschreiben Sie kurz, warum der Unified Process ein **iterativer** und **inkrementeller**

Entwicklungsprozess ist.

c) Was versteht man unter **nichtfunktionalen Anforderungen** an ein Software-System?

Geben Sie mindestens **3 Beispiele** für nichtfunktionale Anforderungen an!

**Aufgabe 2: Kompositum-Muster**

Entwickeln Sie ein Klassenmodell unter Verwendung des Kompositum-Pattern für das

folgende Beispiel:

– eine Hardware-Komponente (z.B. ein PC) besteht aus einem einzelnen Bauteil (mit

Namen und Preis) oder aus einem Teilsystem

– ein Teilsystem hat einen Namen und besteht aus einzelnen Bauteilen und kann

wiederum Teilsysteme besitzen. Der Preis eines Teilsystems ergibt sich aus den

Preisen seiner Komponenten.

a) Erstellen Sie ein Klassenmodell (mit Methodensignaturen) unter Verwendung des

**Kompositum-Pattern** mit seinen erforderlichen Beziehungen.

b) Spezifizieren Sie die erforderlichen Klassen und Methoden (in Pseudocode oder Java),

um eine Hardware-Komponente mit seinen Bauteilen und Teilsystemen zu konstruieren

und um den Preis des Gesamtsystems zu bestimmen. Geben Sie dabei an, welche

Methoden ggfs. abstrakt sind (eine main-Methode ist nicht notwendig).

**Aufgabe 3: Zustands- und Aktivitätsdiagramme**

a) Erstellen Sie ein **Aktivitätsdiagramm** für den folgenden Ablauf. Dabei gibt es zwei

Objekte Besucher und Kassiererin

– Ein Theaterbesucher kommt am Theater an und stellt sich vor der Kasse an.

– wenn am Theater angeschlagen ist, daß das Theaterstück ausverkauft ist, geht der

Besucher nach Hause

– die Kassiererin wartet auf Besucher. Sie verkauft sukzessive jedem, der vor dem

Schalter steht, eine Eintrittskarte. Die Kassiererin kann jeweils nur einen Besucher zur

Zeit bedienen

– der Theaterbesucher wartet so lange bis er von der Kassiererin eine Karte bekommt,

dann geht er in den Theatersaal und die Kassiererin wartet auf den nächsten Besucher

Das Aktivitätsdiagramm soll sowohl die Tätigkeit der Kassiererin wie auch das eventuelle

Warten der Besucher abbilden!

Entwerfen Sie das Diagramm so, daß **nur die Aktivitäten sequentiell ablaufen, für die**

**das wirklich erforderlich ist, d.h. parallelisieren Sie so viele Aktivitäten wie möglich!**

b) Modellieren Sie ein **Zustandsdiagramm**, welches das Verhalten einer **digitalen**

**Stoppuhr** mit drei Einstell-Druckknöpfen darstellt. Nach dem Einlegen der Batterie

befindet sich die Uhr im Normalzustand

– **Knopf 1** erlaubt es, die Zeitmessung zu starten (Stoppuhr starten), die Zeitmessung

anzuhalten (Stoppuhr anhalten) und eine angehaltene Zeitmessung fortzusetzen

– **Knopf 2** ermöglicht es, die Anzeige der Stoppuhr bei der Zeitmessung anzuhalten bzw.

fortzusetzen (Achtung, auch nach dem Anhalten der Anzeige läuft die Zeitmessung

weiter)

– **Knopf 3** setzt die Stoppuhr in den Normalzustand (0:00, keine Zeitmessung) zurück

Es wird davon ausgegangen, daß die Knöpfe nicht gleichzeitig gedrückt werden können.

Ein Endzustand braucht nicht modelliert zu werden.

**Aufgabe 4: Refactoring**

1. Schreiben Sie kurz die Grundidee **(d.h. Ziel und Vorgehen)** des Refactoring!

b) Verbessern Sie das nachfolgende Konstrukt durch eine geeignete Refactoring-

Maßnahme. Geben Sie hierzu den geänderten **Source-Code** der zahleGehalt()-

Methode **und** das geänderte **Klassenmodell** an. Achten Sie auf die korrekten Parameter!

**Mitarbeiter** int zahleGehalt() {

Manager: int switch (getTyp()) {

Arbeiter: int case Mitarbeiter.ARBEITER

Typ: int return \_monatsGehalt;

case Mitarbeiter.MANAGER

zahleGehalt() return \_monatsGehalt + bonus;

default:

throw new Exception (…

**Aufgabe 5: Modellierung**

Entwerfen Sie ein Klassendiagramm für ein Software-System zur Bestellung von Handys,

das den folgenden Sachverhalt abbildet:

– ein Kunde kann ein oder mehrere Handys bestellen

– jedes Handy ist von einem bestimmten Typ (z.B. Nokia 12, Siemens Sx etc.)

– für jedes Handy kann eine Menge von Zubehörteilen ausgewählt werden (z.B.

Handytasche, bunte Handyschalen, Headset etc.)

– jedes Zubehörteil ist von einem bestimmten Typ

a) Entwerfen Sie ein **Klassenmodell** mit allen **Beziehungen** und **Kardinalitäten** und

**benennen** Sie jede Beziehung mit einem geeigneten Namen

c) Skizzieren Sie in Java oder Pseudocode die Methode, die für ein Handy ein Zubehörteil

hinzufügt und dabei überprüft, ob der **Typ des Zubehörteils** für den **Handy-Typ** erlaubt

ist. Geben Sie hierbei an, in welcher Klasse welche der verwendeten Methoden deklariert

ist!

**Aufgabe 6: Test (Äquivalenzklassen/ Grenzwertanalyse)**

Die folgende Methode dient zum Setzen eines Wochentags und soll getestet werden:

void setzeWochentag(short aktuellerTag);

Für den Eingabeparameter muss gelten: 1 <= aktuellerTag <= 31, ansonsten ist die

Eingabe ungültig.

1. Bestimmen Sie die **Äquivalenzklassen** für die obige Methode und geben Sie für

jede gefundene Äquivalenzklasse einen **Testfall** an.

2. Führen Sie eine **Grenzwertanalyse** durch und geben Sie die dabei gefundenen

**Testfälle** an.

3. Beschreiben Sie was ein **Grenzwert** ist und geben Sie 3 Beispiele für typische

**Grenzwertfehler** an.