Name: | Vorname: | Matr.Nr.:

Technische Universität München
Fakultät für Informatik
Prof. Dr. Dr. h.c. M. Broy

Wintersemester 2006/2007
27. Oktober 2006

EINFÜHRUNG IN DIE SOFTWARETECHNIK

Wiederholungsklausur

(Gesamtpunktezahl 120)

Aufgabe 1: Multiple-Choice-Fragen zu den Entwicklungsphasen (18 Punkte)

Geben Sie für jede der folgenden Aussagen durch Ankreuzen an, ob Sie der Aussage zustimmen oder nicht.
(Hinweis zur Punktebewertung einer Aussage: kein Kreuz = 0 Punkte, zwei Kreuze = -1 Punkt, ein richtiges Kreuz = +1 Punkt, ein falsches Kreuz = -1 Punkt)

Ja	Nein	Aussage
☒	☐	Transaktionsmonitore (z.B. UTM, CICS) dienen unter anderem zur Sequentialisierung paralleler Abläufe.
☐	☒	Die detaillierten Systemanforderungen werden im Systemfeinentwurf erstellt.
☐	☒	Das Pflichtenheft ist das Ergebnisdokument einer Systemstudie.
☒	☐	Bei der Validierung werden Systemanforderungen auf Adäquatheit überprüft.
☐	☒	Anforderungen an die Portierbarkeit sind Teil der funktionalen Anforderungen eines Systems.
☐	☒	In der Implementierung werden ER-Diagramme verwendet, um Testfälle zu beschreiben.
☐	☒	3-Schichten-Architekturen (3-Tier-Architekturen) sind nur für Client-Server-Hardware-Architekturen geeignet.
☒	☐	Die Weiterentwicklung von Legacy-Software ist häufig problematisch wegen ihrer Plattformabhängigkeit.
☒	☐	Design-by-Contract beschreibt Prozeduren (Methoden) durch Vor- und Nachbedingungen.
☒	☐	Middleware (z.B. CORBA, RMI) bezeichnet Software, die zum Beispiel Dienste für Nachrichtenaustausch, Datenzugriff, Prozeduraufrufe, etc. auf Client-Server-Systemen bereitstellt.
☒	☐	Zustandsübergangsdiagramme sind insbesondere gut geeignet zur Beschreibung des Ablaufverhaltens von Systemen.
☐	☒	Deploymentarchitektur beschreibt die Abbildung von Prozeduren (Methoden) auf Softwaremodule.
☒	☐	Statische Architekturaspekte lassen sich insbesondere durch Blockdiagramme gut beschreiben.
☐	☒	Design-Patterns sind vorgefertigte und ablauffähige Codeteile für bestimmte Problemklassen.
☒	☐	Frameworks sind vorgefertigte und meist nicht voll ablauffähige Codeteile.
☐	☒	Modul- und Systemintegrationstest sind immer auch Teile des Systemabnahmetests.
☐	☒	Programmtests ermöglichen den Nachweis der Fehlerfreiheit eines Programms.
☐	☒	Die Systemänderbarkeit resultiert aus der Qualität der funktionalen Systemanforderungen.

Aufgabe 2: Qualitätssicherung (30 Punkte)

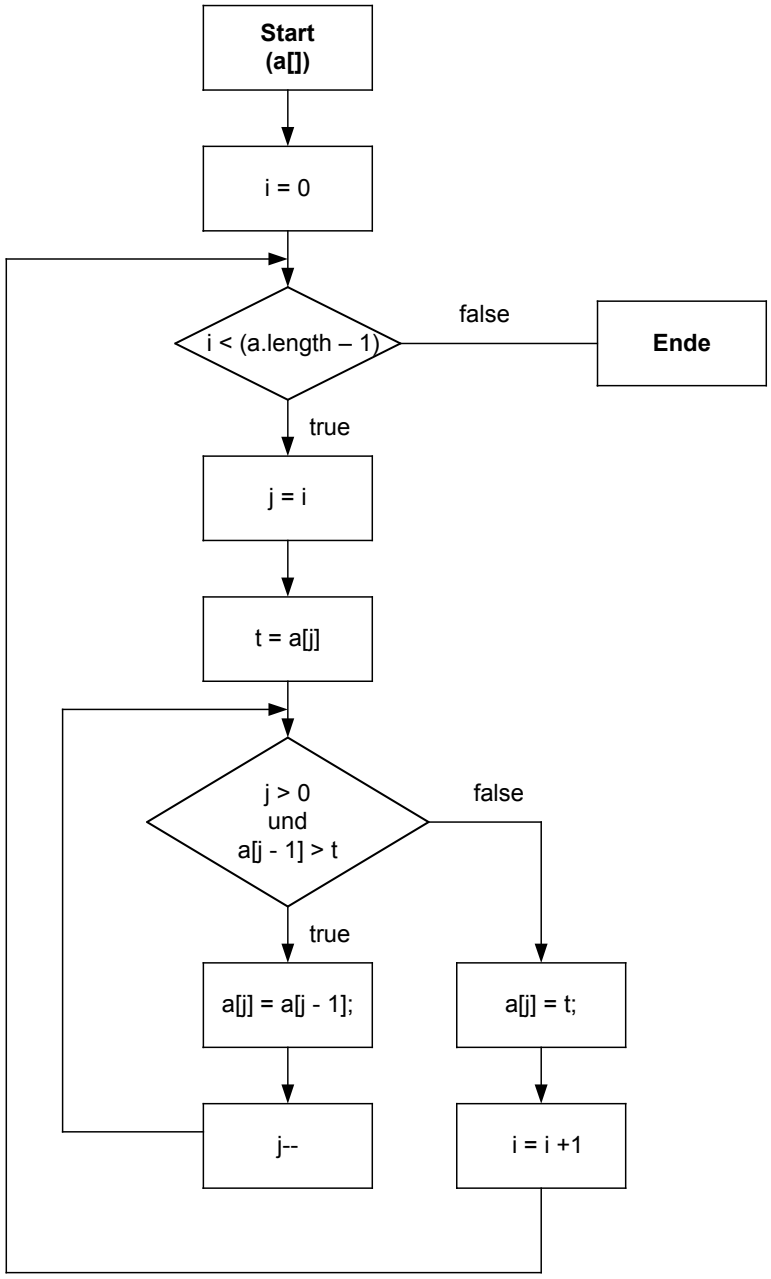
Das Zusatzblatt zur Angabe enthält Software-Quellcode, der *Sortieren durch Einfügen* implementiert. Dieser ist durch Inspektion zu überprüfen. Im Folgenden sind die Regeln der Inspektion angegeben.

RM1	(Dokumentation)	Jede Quellcode-Datei beginnt mit einem Kommentar, der den Dateinamen, Versionsinformation und Datum enthält.
RM2	(Dokumentation)	Deklarationen von Variablen werden kommentiert.
RM3	(Formatierung)	Zwischen dem Namen einer Methode und der Klammer der Parameterliste steht kein Leerzeichen.
RM4	(Formatierung)	Vor jeder Zeile, die einen Kommentar enthält, steht eine Leerzeile. Dies gilt nicht für die erste Zeile des Programms.
RM5	(Formatierung)	Nach einem Komma oder einem Semikolon steht ein Leerzeichen oder Zeilenumbruch.
RM6	(Bezeichner)	Klassennamen beginnen mit einem Großbuchstaben, Variablennamen und Methodennamen beginnen mit einem Kleinbuchstaben.

a) Überprüfen Sie durch Inspektion, ob die obigen Regeln für den Quellcode eingehalten wurden. Erstellen Sie eine Liste mit allen Verletzungen der Regeln. Geben Sie für jede Verletzung einer Regel die Zeilennummer, Regelnummer und Kommentar an, z.B. (07, RM5, kein Leerzeichen nach dem Komma). Schreiben Sie nicht in den Quellcode.
b) Geben Sie den Kontrollflussgraphen für die Methode „sort“ an.
c) Geben Sie einen Testfall für die Methode „sort“ an, der insgesamt eine vollständige Anweisungsüberdeckung leistet. Geben Sie außerdem die Werte des Feldes an, die
   1. laut Spezifikation nach Ausführung von „sort“ zu erwarten sind,
   2. sich nach Ausführung des Quelltextes von „sort“ ergeben.
d) Entspricht die Methode „sort“ ihrer Spezifikation, die durch die Nachbedingungen angeben ist? Geben Sie gegebenenfalls Korrekturen der Methode an.

Lösung:
Zu a)
4 P:
25, RM2, Deklaration nicht kommentiert.
16, RM3, Es darf kein Leerzeichen nach dem Methodennamen stehen.
07, RM4, Leerzeile fehlt.
16, RM6, der Name der Methode beginnt mit einem Großbuchstaben.

Zu b)
12 P:



Zu c)
3+1+5 P: Beispielhafter Testfall:

Eingabefeld	*Laut Spezifikation zu erwarten*	*Wirkung der Methode „sort“*
[3,5,4,1]	*[1,3,4,5]*	*[3,4,5,1]*

Zu d)
5 P: Nein. Der Ausdruck in der Zeile 19 müsste lauten: i < a.length

Aufgabe 3: Zustandsübergangsdiagramm (38 Punkte)

Modellieren Sie einen Getränkeautomaten. Hierfür ist folgende Funktionalität gegeben:

- Der Getränkeautomat bietet zwei Arten von Getränken: Wasser und Cola.
- Die beiden Getränke unterscheiden sich im Preis.
- Will jemand ein Getränk kaufen, so wählt er zunächst das gewünschte Getränk.
- Danach wirft er solange Münzen ein, bis der geforderte Betrag erreicht oder überschritten ist. Der Automat zeigt an, wie viel Geld eingeworfen wurde.
- Sobald genug Geld eingeworfen ist, werden das Getränk und das Wechselgeld ausgegeben.
- Bei Abbruch wird das gesamte bisher eingeworfene Geld zurückgegeben.

Beachten Sie hierbei folgendes:

- Der Automat hat nur eine begrenzte Anzahl an Waren jedes Typs.
- Geht eine Ware aus, kann diese nicht mehr gewählt werden. Die andere Ware bleibt weiter wählbar. Hierfür muss der Automat die Zahl der vorhandenen Waren speichern.

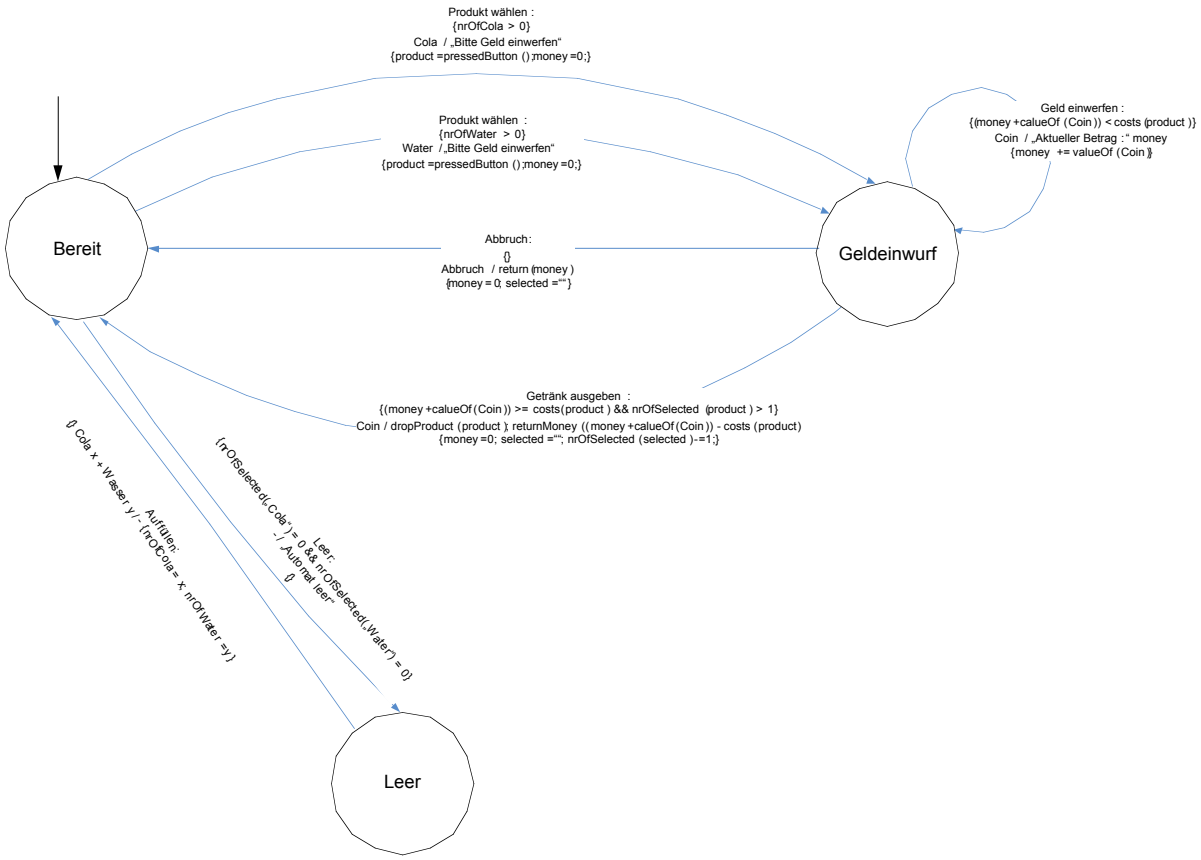
Als Einschränkung für den Automaten gilt, dass dieser mit den drei Zuständen „Bereit“, „Geldeinwurf“ und „Leer“ auskommen muss. Der Zustand „Leer“ wird nur erreicht, wenn beide Waren nicht mehr angeboten werden können.

Gegeben sind außerdem folgende Funktionen, die Zugriff auf die in Aufgabe a) zu benennenden Attribute ermöglichen:

int valueOfCoin(Coin coin) liefert den Wert einer eingeworfenen Münze,
int costs(String product) liefert den Preis einer gewählten Ware,
Integer nrOfProduct(String product) liefert eine Referenz auf die Anzahl der verfügbaren Waren eines Typs,
String pressedButton() liefert den Namen des durch Knopfdruck gewählten Produktes,
void returnMoney(int money) veranlasst die Auszahlung eines übergebenen Betrages,
void dropProduct(String product) veranlasst die Warenausgabe.

a) Nennen Sie alle Zustandsattribute, die der Automat benötigt. Beschreiben Sie deren Zweck kurz in Prosa. (4P je 1)
[NAT: nrOfCola, NAT: nrOfWater, NAT: money, STRING: product]
b) Benennen Sie alle Ein- und Ausgaben für den Automaten. Beschreiben Sie deren Zweck kurz in Prosa. (8P je 1)
[Geld, Produktwahl, Cola, Wasser][Geld, Cola, Wasser, eingeworfener Betrag]
c) Beschreiben Sie alle Attribute so in einer Tabelle, dass ersichtlich ist, welche Werte in den drei Zuständen jeweils zulässig sind. (4P je Attribut 1)
d) Zeichnen Sie den Automat. Verwenden Sie hierbei unbedingt die in der Vorlesung und Übung gezeigte Syntax mit Vor- und Nachbedingung. (22P 3/3/3/2/5/2 + 1/2/1)

	nrOfCola	nrOfWater	money	product
Bereit	>=0	>=0	0	„“
Geldeinwurf	>=0	>=0	>=0	„cola“ || „water“
Leer	0	0	0	„“



Aufgabe 4: Systementwicklung (34 Punkte)

Sie wurden beauftragt ein neues Bibliotheksverwaltungssystem zu entwickeln. Dieses System soll den Bücherbestand verwalten sowie Ausleih- und Vormerkungsfunktionalität bieten.
Rahmendaten zum Projekt: Von Seiten des Auftraggebers wird großer Wert auf besonders hohe Sicherheitseigenschaften des Systems gelegt. Im Rahmen des Projekts sollen von Anfang an Prototypen den Projektfortschritt dokumentieren. Zur Entwicklung dieses Systems steht Ihnen ein Team von sechs Entwicklern zur Verfügung. Das System muss innerhalb von 8 Monaten fertig gestellt sein.

a) Nennen Sie drei Stakeholder des Bibliotheksverwaltungssystems und jeweils zwei Anwendungsfälle

Lösung [3+6 P

- Bibliothekar
  - Bücher eintragen, löschen, aktualisieren
  - Bücher verleihen, zurücknehmen
  - Mahngebühren berechnen
  - Neue Benutzer eintragen, Benutzer löschen
- Bibliotheksbenutzer
  - Bücher vormerken, Vormerkung stornieren
  - Mahngebühren einsehen,
  - Erinnerung an Buchrückgabe
  - Bücher suchen
- Administrator
  - Eintragen und Löschen von Bibliothekaren
  - Performance-Überwachung

]

b) Welches der Vorgehensmodelle V-Modell 97, V-Modell XT, Spiralmodell, Extreme Programming würden Sie unter den geschilderten Umständen für die Entwicklung dieses Systems anwenden? Treffen Sie eine eindeutige Entscheidung und begründen Sie diese mit drei schlüssigen Argumenten.

Lösung [9 P
z.B.: XP wegen

- Anforderungen instabil
- Kleines Team (Dokumentations-Overhead vermeiden)
- Kurze Projektlaufzeit

]

c) Das zu entwickelnde Softwaresystem sollte die Qualitätseigenschaften *Benutzbarkeit*, *Zuverlässigkeit* und *Wartbarkeit* erfüllen. Geben Sie zu jeder dieser Qualitätseigenschaften je zwei Qualitätssicherungsmaßnahmen an.

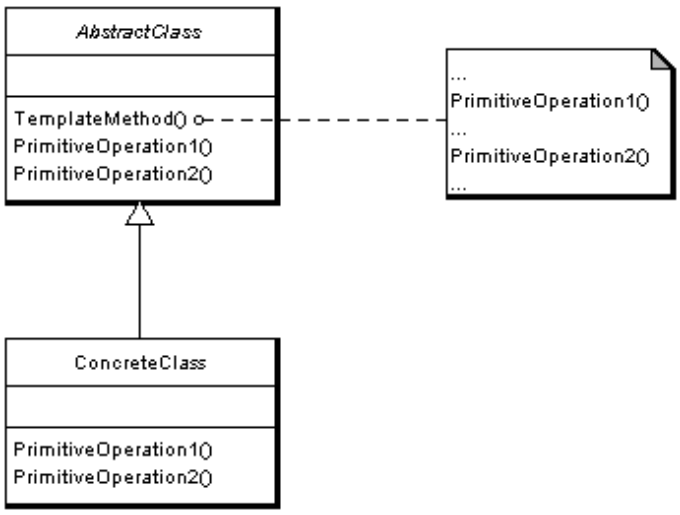
Lösung [6 P

- Benutzbarkeit — Interviews und Reviews mit Nutzern, Prototyping
- Zuverlässigkeit — Testen, Formale Verifikation
- Wartbarkeit — Coding Standards, Code Reviews, Verwendung von Mustern, Metriken

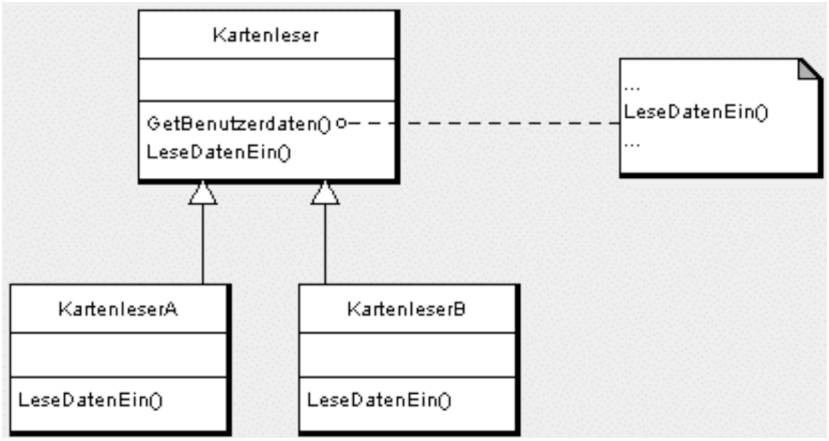
]

d) Das von Ihnen zu entwickelnde System soll ein derzeit in Betrieb befindliches System ablösen. Das bisherige System verwendet ein Chipkartensystem zur Identifikation der Bibliotheksbenutzer. Im Zuge der Systemumstellung soll ein neues Kartensystem eingeführt werden. Um nicht sofort alle alten Karten ersetzen zu müssen, soll das neue Bibliotheksverwaltungssystem die alten Karten weiterhin unterstützen. Zeichnen Sie ein Klassendiagramm, um unter Verwendung des Patterns *TemplateMethod* (s.u.) diese Anforderung zu erfüllen.

Intention des *TemplateMethod*-Patterns: Definiere das Gerüst eines Algorithmus in einer Methode wobei die Implementierung einiger Schritte in Unterklassen ausgelagert wird. Template Method lässt die Unterklassen bestimmte Schritte in einem Algorithmus redefinieren, ohne die Struktur des Algorithmus zu verändern.



Lösung [10 P



]

Dieses Blatt bitte nicht beschriften!

```
00 /*
01  * InsertionSorter.java
02  * Version 1.0, 27.09.2006
03  *
04  * Eine Implementierung des Verfahrens Sortieren durch Einfügen
05  */
06 public class InsertionSorter {
07   /**
08    * Die folgenden Methode sortiert ein Eingabefeld, gegeben durch
09    * den Parameter a
10    *
11    * Nachbedingung:
12    * Das Eingabefeld a ist linear geordnet, d.h.
13    * für alle k gilt: ( 1 <= k < a.length ) ==> ( a[k-1] <= a[k] )
14    *
15    */
16   public static void Sort  (int[] a) {
17
18     // i ist der Index für das zu sortierende Element
19     for (int i = 0; i < (a.length - 1); i = i + 1) {
20
21       // Das Feld ist bis zu dem Element a[i-1] bereits aufsteigend sortiert
22       // Ab dem Element a[i] ist es noch unsortiert
23
24       int j = i;      // Der Index für den Vergleich mit der unteren Hälfte
25       int t = a[j];
26
27       // Das Element a[i] wird mit a[i-1], a[i-2], etc. verglichen.
28       // Wenn ein Element a[j] mit a[j] <= a[i] gefunden wird,
29       // wird a[i] hinter diesem eingefügt.
30       // Wird kein solches Element gefunden,
31       // wird a[i] an den Anfang des Feldes gesetzt.
32       while (j > 0 && a[j - 1] > t) {
33         a[j] = a[j - 1];
34         j--;
35       }
36
37       a[j] = t;
38     }
39   }
40 }
```