Prof. Dr. Franzen/Prof. Dr. Wüst
FB MNI
24. Januar 2002

Klausur zu Programmieren 3 (Java), WS 01/02

Füllen Sie bitte zunächst dieses Deckblatt vollständig aus und unterschreiben es.

Name,Vorname:

Matrikelnummer:

Semester:

Ich bestätige die Richtigkeit der gemachten Angaben.

Unterschrift:

Bearbeiten Sie die Aufgaben bitte auf den folgenden Blättern. Verwenden Sie bei Bedarf auch die Rückseiten. Jede Aufgabe sollte allerdings nur auf dem Blatt bearbeitet werden, auf dem sich die Aufgabenstellung befindet.

Bearbeitungsdauer: 110 Min.

Viel Erfolg!

Aufgabe	Max. Punkte	Erreichte Punkte
1	18	
2	16	
3	14	
4	14	
5	38	
Summe	**100**	

Aufgabe 1 (Programmstruktur)

(a) Erläutern Sie das Konzept des Java Bytecodes: Wie entsteht er, wie wird er ausgeführt und wieso werden Java-Programme durch ihn plattformunabhängig?

(3)

(b) Die Methode `foo()` sei in der Toplevel-Klasse `C` definiert, die zum Paket `p.q.r` gehört. Wie heißt die Byte-Code-Datei, in der sich der zu `foo()` gehörige Byte-Code befindet? In welchem Verzeichnis muss sich diese Byte-Code-Datei im Datei-System befinden? Welchen Einfluss hat die Umgebungsvariable `CLASSPATH`?

(3)

(c) Was bewirkt eine **`import`**-Anweisung?

(2)

(d) Was versteht man unter einer anonymen Klasse, einer lokalen Klasse und einer Elementklasse?

(4)

(e) Das Datenfeld `x` sei innerhalb der Klasse `C` im Paket `p` ohne eines der Sichtbarkeitsattribute (**`public`**, **`private`** oder **`protected`**) definiert:

```
package p;
class C{int x;}
```

Erläutern Sie, an welchen Stellen eines Programmes dieser Name sichtbar ist und wo nicht!

(3)

(f) Geben Sie die kürzestmögliche Klassendefinition an!

(3)

Aufgabe 2 (Typen, Speicherverwaltung)

(a) In Java werden Referenztypen und primitive Typen unterschieden. Erläutern Sie den Unterschied anhand der Zuweisung `x = y` und anhand des booleschen Ausdrucks `x == y`!

(4)

(b) Welche Arten von Referenztypen werden in Java unterschieden?

(2)

(c) Zählen Sie alle primitiven Typen von Java und den Speicherbedarf für ihre Werte auf!

(4)

(d) Ist es möglich,dass der für ein Objekt benötigte Speicher nicht auf dem Heap angelegt wird? Begründen Sie!

(2)

(e) Welchen Einfluss hat ein Java Programm auf die Freigabe von Speicherplatz auf dem Heap?

(2)

(f) Sind Felder (d.h. Arrays) auch Objekte? Kann also auch auf ein Feld die Methode `toString()` angewendet werden?

(2)

Aufgabe 3 (Threads)

(a) Zeichnen Sie ein Zustandsübergangsdiagramm für einen Java Thread. Tragen Sie insbesondere die Übergänge ein, die durch Aufrufe der Methoden `start()`, `yield()`, `sleep()`, `wait()` und `notify()` bewirkt werden.

(8)

(b) Was unterscheidet einen Java Thread von einem Betriebssystem-Prozess?

(2)

(c) Wann endet ein Thread?

(2)

(d) Wie lassen sich in Java kritische Abschnitte spezifizieren?

(2)

Aufgabe 4 (Vererbung)

(a)Ist folgende Klassendefinition zulässig? Begründen Sie!

```
class C extends D, F {/*Klassenkörper folgt hier.....*/}
```

(2)

(b)Ist folgende Klassendefinition zulässig? Begründen Sie!

```
class C implements I, J, K {/*Klassenkörper folgt hier.....*/}
```

(2)

(c)Es sei B eine Unterklasse von A. Wie wird gewährleistet, dass bei Erzeugung eines B-Objektes auch ein Konstruktor von A aufgerufen wird. Wie lautet die Regel, wenn B überhaupt keinen Konstruktor definiert?

(4)

(d) Welche Konsequenz hat es, wenn eine Klassendefinition das Schlüsselwort `final` verwendet? Was bedeutet das gleiche Schlüsselwort bei einer Methodendefinition?

(3)

(g)In Java ist nur einfache Vererbung für Klassen zulässig. Welchen Vorteil bietet dies beim Binden einer Methode?

(3)

Aufgabe 5

Gegegeben sei die folgende Schnittstelle für einen Stapel (Stack):

```
interface Stapel{

//legt elem auf dem Stapel ab
public void push(Object elem);

//liefert das oberste Element des Stapels und entfernt es
public Object pop() throws java.util.EmptyStackException

//gibt an, ob der Stapel leer ist oder nicht
public boolean istLeer();

//liefert das oberste Element des Stapels, ohne es zu entfernen
public Object peek();


}
```

(a) Definieren Sie eine Klasse `StapelImp`, die diese Schnittstelle implementiert. Die Klasse soll einen parameterlosen Konstruktor besitzen. Die Klasse `java.util.EmptyStackException` soll als vordefiniert angesehen werden. Hinweis: Besonders einfach wird es, wenn Sie die Klasse `StapelImp` als .Erweiterung der Klasse `java.util.Vector` anlegen.

(24)

(b) Schreiben Sie ein Testprogramm, das die Zeichenketten `"unten"`, `"mitte"`, `"oben"` ablegt und wieder entfernt. Weiterhin soll das Programm prüfen, ob die Ausnahme `java.util.EmptyStackException` und tatsächlich auftritt!

(14)