Prof. Dr. Franzen
FB MNI
23. Januar 2003

Klausur zu Programmieren 3 (Java), WS 02/03

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

Name,Vorname:	
Matrikelnummer:	
Semester, Studiengang (I, BI oder MI):	

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	11	
6	47	
Summe	**120**	
Bonus HÜ		

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)Warum kann man in einer Klassenmethode (also einer Methode, die mit dem Schlüsselwort **`static`** vereinbart wurde) nicht die Objektreferenz **`this`** verwenden?

(2)

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

(4)

(e)Dürfen zwei Attribute einer Klasse den gleichen Namen besitzen? Dürfen zwei Methoden einer Klasse den gleichen Namen besitzen? Begründen Sie Ihre Antwort!

(3)

(f)Zählen Sie alle Arten von Bezeichnern auf, die in einem Java Programm vereinbart werden können!

(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)Warum ist ein Laufzeitsystem mit automatischer Speicherbereinigung für eine Echtzeitanwendung ungeeignet?

(2)

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

(4)

(d)Unter welchen Umständen wird die Methode `finalize()` der Klasse `Object` aufgerufen?

(4)

(e)Kann die Methode `Object.wait()` auch auf ein Array angewendet werden? Begründen Sie!

(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. Welchen Zustand nimmt der Thread nach Ablauf der Methode `run()` ein, welchen direkt nach der Erzeugung?

(8)

(b)Auf ein Objekt der Klasse `java.lang.String` sollen mehrere Threads zugreifen können. Was ist zu tun? Begründen Sie!

(2)

(c)Innerhalb eines Methodenrumpfes wird die Methode `wait()` aufgerufen. Bei Ausführung des Programmes wird an dieser Stelle aber eine `IllegalMonitorStateException` ausgelöst. Woran mag das liegen?

(2)

(d)Wie lassen sich in Java kritische Abschnitte spezifizieren?Welche Objekte können als Sperre fungieren?

(2)

Aufgabe 4 (Vererbung)

(a) Die Methode `A.meth(X,Y)` wird in der Klasse `B` überschrieben (s.u.). In welchem Zusammenhang müssen die Klassen `AException` und `BException` stehen? Begründen Sie!

```
class A {...
      void meth(X x, Y y)throws AException{..}
}
class B extends A{...
      void meth(X x, Y y)throws BException{...}
}
```

(3)

(b)Eine Methode `B.meth(...)` werde von einer Methode `A.meth(...)` überschrieben. Wie verhält es sich mit den Vor- und Nachbedingungen der beiden Methoden, dürfen sie abgeschwächt oder verschärft werden? Begründen Sie Ihre Antwort mit dem Ersetzungsprinzip!

(4)

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

(3)

Aufgabe 5 (Schnittstellen)

(a)Im Paket `java.lang` ist die Schnittstelle `Runnable` deklariert. Kann man mit

```
Runnable r = new Runnable();
```

ein `Runnable` Objekt erzeugen? Begründen Sie!

(2)

(b)Können in einer Schnittstellendefinition Attribute eines Objektes definiert werden?

(1)

(c)Erläutern Sie den Unterschied zwischen einer Schnittstelle und einer abstrakten Klasse!

(2)

(d)Kann eine Klasse mehrere Schnittstellen implementieren? Anders ausgedrückt, ist

```
class A implements I1, I2{...}
```

zulässig?

(1)

(e) Die Klasse `Thread` besitzt u.a. einen Konstruktor `Thread(Runnable)`, dessen Parameter den Schnittstellentyp `Runnable` besitzen muss. Was bewirkt dieser Konstruktor? An welchen Stellen kann bei Verwendung dieses Konstruktors die Methode `run()` implementiert werden?Vergleichen Sie mit der Schnittstelle `ActionListener` und ihrer Verwendung bei der Registrierung an der Ereignisquelle mittels `addActionListener(..)`.

(5)

Aufgabe 6 (Implementierung eines Beispiels)

Gegegeben sei die folgende Schnittstelle für eine nach dem FIFO-Prinzip organisierte Warteschlange unbeschränkter Größe:

```
interface Schlange {

// Fügt ein von null verschiedens Objekt am Ende der Schlange ein.
public void einfügen (Object obj) throws IllegalArgumentException;

// Entfernt das Objekt am Anfang der Schlange.
// Liefert null, wenn die Schlange leer ist.
public Object entfernen ();

// Liefert true, falls Schlange leer, false sonst.
public boolean istLeer();

// Liefert Länge der Schlange.
public int gibLänge();

// Darstellung als String: beginnt mit "<",
// gefolgt von einer String-Darstellung für jedes Element,
// jede getrennt durch Leerräume, endet mit ">"
public String toString();
}
```

(a)Schreiben Sie ein Test-Programm zum Test der Implementierungen in (c) und (d): Darin soll eine Schlange `q` mit den 4 Elementen `1`, `4`, `'a'`, `"Element"` erzeugt werden, dann soll `q` auf die Standardausgabe mittels der Methode `toString()` ausgegeben werden und schliesslich soll es `q` vollständig leeren.
Hinweis: Beachten Sie, dass `int`- und `char`-Werte nicht vom Typ `Object` sind! Abhilfe?

(10)

(b) Schreiben Sie eine Klasse `VerketteteSchlange`, die die Schnittstelle `Schlange` implementiert und die Elemente miteinander verkettet.
Hinweis: Führen Sie eine geschachtelte Top-Level Klasse `Element` mit Attributen `inhalt`, `nachfolger` ... ein.

(20)

(c) Schreiben Sie eine Klasse `VectorSchlange`, die die Schnittstelle `Schlange` implementiert und sich dabei abstützt auf die Klasse `Vector` des Paketes `java.util` mit den Methoden

```
void addElement(Object),
Object remove(int),
int size(),
Object elementAt(int) und
boolean isEmpty().
```

(11)

(d)Wie können Sie Ihre Implementierungen "thread-sicher" machen, d.h. den nebenläufigen Zugriff auf einen Stapel durch mehrere Threads. Beachten Sie bei der Antwort zur Klasse `VectorSchlange` den folgenden Satz der API-Dokumentation der Klasse java.util.Vector: "Vector is synchronized".

(6)