

Abgabe vor der Vorlesung 08.11.05

Aufgabe 0 (Wiederholung) [optional]

Stellt sicher, dass ihr alle Begriffe aus der letzten und diesen Woche gut verstanden habt:

- Schnittstellen (Rechte, Pflichten, Auswirkungen) in Java, Modula, Haskell
- Typverträglichkeit (statischer/dynamischer Typ, Untertyp)
- Prozedurtyp
- Anonyme Klassen
- Abstrakte Fabrik als ein Entwurfsmuster für die Arbeit mit Schnittstellen
- Polymorphie
- Vererbung (((in)direkte)Unter-/Oberklasse, Spezialisierung/Generalisierung, Vererbungshierarchie)

Ein Weg mit einer solchen Menge an Definitionen und Begriffen umzugehen, ist sinnvolle Vor- und Nachbereitung der Vorlesung. Die Vorbereitung besteht normalerweise aus einem Durchgehen der Begriffe der letzten Vorlesungen, damit man nicht zu leicht abgehängt wird (15 Minuten sind genug). Die Nachbereitung dauert etwas länger (45 Minuten und mehr) und sollte eine eigene schriftliche Zusammenfassung der Begriffe enthalten.

Aufgabe 1 (Anonyme Klassen)

Anonyme Klassen (nachzulesen unter http://www.unix.org.ua/oreilly/java-ent/jnut/ch03_12.htm) sind ein wichtiger Teil von Java für die praktische Programmierung. Strukturieren Sie die Klasse `Main` so um, dass als `MouseListener` eine anonyme Klasse verwendet wird und nicht mehr die Fensterklasse `MouseListenerPanel` die Schnittstelle implementiert. Welchen Vorteil hat diese Umwandlung? Gibt es Nachteile?

Sie finden den Quelltext unter: <http://www.inf.fu-berlin.de/lehre/WS05/ALP3/material/U3.A1.jar>

Aufgabe 2 (Typklassen in Haskell, Schnittstellen in Java)

Eine vereinfachte Speicherverwaltung soll in Haskell und Java als abstrakter Datentyp implementiert werden. An die Speicherverwaltung werden Anfragen zum Reservieren (`allocate`) und Freigeben (`deallocate`) von (zusammenhängenden) Speicherbereichen gestellt. Beim Reservieren wird ein (zusammenhängender) Speicherbereich mit einer bestimmten Größe angefordert, die Speicherverwaltung versucht dann einen solchen Bereich zu finden, reserviert ihn und liefert dessen Startadresse zurück. Als Strategie zum Finden eines freien Speicherbe-

reichs soll "first fit" verwendet werden: Der Speicher wird, beginnend bei Adresse 0, nach einem genügend großen (evtl. größeren) freien Speicherbereich durchsucht. Wird ein solcher gefunden, wird darin ein Bereich der angeforderten Länge reserviert. Achtung: ggf. bleibt am Ende eine Lücke, die weiterhin als freier Speicher gilt.

Beim Freigeben wird die Speicherverwaltung aufgefordert, einen bestimmten Speicherbereich freizugeben (dies ist auch dann möglich, wenn der Speicherbereich, oder Teile davon, gar nicht reserviert waren).

Schnittstelle (Haskell):

```
class Memory m where
  allocate :: m -> Int -> (m, Int)
    -- bekommt Speicher und gewünschte Länge, liefert Paar aus
    -- modifiziertem Speicher und Startadresse des reservierten Bereichs
    -- bricht mit "error" ab, falls kein entsprechender Bereich
    -- vorhanden ist
  deallocate :: m -> Int -> Int -> m
    -- bekommt Speicher und Start- und Endadresse des freizugebenden
    -- Bereichs, liefert den modifizierten Speicher
    -- bricht mit "error" ab, falls ungültige Adresse(n) gegeben
```

Schnittstelle (Java):

```
interface Memory {
  int allocate(int length); // Wirft Ausnahme im Fehlerfall.
  void deallocate(int start, int end); // ditto.
}
```

Für die Implementierung bieten sich die zwei folgenden Datenstrukturen an. Entscheiden Sie sich jeweils für eine der beiden und begründen Sie kurz Ihre Wahl. In Haskell sollte ihre Implementierung zusätzlich die Typklasse `Show` implementieren und in Java die Methode `String toString()` überschreiben.

- a.) Als Modell dient eine Liste/ein Feld von Wahrheitswerten (Bool, boolean), wobei der n-te Wert dieser Liste/dieses Felds ausdrückt, ob der n-te Speicherblock reserviert oder verfügbar ist.

Beispiel:

Index/Adresse	0	1	2	3	4	5	6	7
Wert	false	true	true	false	false	true	true	true

- b.) Als zweite Möglichkeit soll der freie Speicher mit Hilfe einer Freispeicherliste verwaltet werden. Diese Liste enthält Paare von Start- und Endadressen der freien Speicherbereiche:

Freispeicherliste (passend zum Beispiel oben):

(0,0)	(3,4)
-------	-------

Achtung: Wird angrenzend an einen freien Speicherbereich weiterer Speicher freigegeben, entsteht ein zusammenhängender freier Bereich!

Abzugeben sind Auszüge aus Ihrer Implementierungen und Ausdrücke von Testläufen. Es wird erwartet, dass Sie Ihre Implementierung getestet und dokumentiert haben.

Aufgabe 3 (Typverträglichkeit bei Schnittstellen - Grundlagen)

Diese Aufgabe stellt sicher, dass ihr die Grundlagen der Typverträglichkeit bei Schnittstellen verstanden habt. Jeder sollte diese Aufgabe eigenständig und ohne Probleme lösen können.

```
Interface IA {
    void foo();
}
class A implements IA {
    void foo(){ System.out.println("A FOO"); }
}
class B implements IA {
    void foo(){ System.out.println("B FOO"); }
    void bar(){ System.out.println("B BAR"); }
}
class Main {
    public static void main(String[] args){
```

Anweisung:	Statisch Korrekt? Wenn ja, Dynamisch korrekt? Wenn nein, welcher Fehler tritt auf?	Ausgabe
A a = new A();		
a.foo();		
a.bar();		
B b = new B();		
b.foo();		
b.bar();		
IA ia = new IA();		
IA i = a;		
i.foo();		
i.bar();		
i = b;		

<code>i.foo();</code>		
<code>i.bar();</code>		
<code>b = i;</code>		
<code>b = (B)i;</code>		
<code>a = (A)i;</code>		

`}
}`

Aufgabe 4 (Typklassen in Haskell) [optional]

Erklären Sie die Implementierung `BatchedQueue` der Typklasse `Queue` (1.2.30). Legen Sie hierfür bei jeder Zeile dar, was mit ihr bewirkt wird. Führen Sie dann auf Papier einen Testlauf aus, der erst 4 Elemente der Schlange anfügt, dann 2 Elemente entfernt, zwei Element hinzufügt, 3 Elemente entfernt und schließlich wieder 2 hinzufügt.

Worin liegt der Vorteil der Implementierung `BatchedQueue` gegenüber der `Standardqueue`?

Aufgabe 5 (Java Wiederholung) [optional]

Unter <http://projects.mi.fu-berlin.de/w/bin/view/SS/JavaWiederholung> findet ihr zwei neue Aufgaben. Die erste konzentriert sich auf die Wiederholung von Feldern und sollte innerhalb von einer Stunde zu lösen sein. Schreibt mir bitte eine Email, wenn euch diese Aufgaben helfen und für euch interessant sind. Bei fehlendem Interesse, würde ich mir nämlich gerne die Mühe sparen.