Alp2 - Übungsblatt 8 (Aufgabe 1)

Bearbeitet von: Jasmine Cavael & Alexander Chmielus
Tutor: Fabian Halama
Tutorium 10 (Do. 16-18)

1a)
Sinn: Klassenmethoden können unabhängig von Objekten über die Klasse aufgerufen werden, daher eignen sie sich u.a. für Klassen ohne Attribute, oder wenn man nur Klassenvariablen benutzt.

Einschränkungen: Sie haben keinen Zugriff auf Instanzvariablen. Dementsprechend dürfen sie nur Klassenvariablen und lokale Variablen benutzen. Außerdem dürfen sie nur Klassenmethoden aufrufen.

b)
Man deklariert Variablen als private, wenn man möchte, dass sie nur in der jeweiligen Klasse sicher sind.
Ja, man kann sie als private deklarieren, wenn man sie zB. nur in der Klasse aufrufen lassen will.

c)
Wenn kein Konstruktor definiert wurde, können nur Klassenmethoden aufgerufen werden, da diese unabhängig von Objekten funktionieren. Sie können sowohl vererbt, als auch als Private deklariert werden.

d)
protected-Variablen sind in der Klasse, allen Unterklassen und im Package der Klasse sichtbar. Konstruktoren können ebenfalls als protected deklariert werden.

e)
Abstrakte Klassen sind Klassen, die mind. eine abstrakte Methode(d.h. Methoden, die keine Implementierung haben, aber eine Deklaration) besitzen. Unterklassen können dann diese Methoden implementieren. Variablen können in einer abstrakten Klasse deklariert werden.

f)
Man kann sowohl Objekte in einer abstrakten Klasse erzeugen, als auch Konstruktoren definieren.

g)
Ja, wenn die Klasse B eine generische Klasse ist und den Typ A auch verwenden kann. Ansonsten geht das nicht.

h)
Ja, natürlich.

i)
Automatisches Boxing und Unboxing sind Vorgänge zum "Ein-" und "Auspacken" von primitiven Datentypen in Referenztypen. Warum wurde das eingeführt? Schätze, um genau sowas über Wrapper-Klassen möglich zu machen.
Nachteile: Oft laufen die Vorgänge im Hintergrund, wodurch mehr Rechen- und Zeitaufwand besteht. Eine weiteres Problem ist das da:
Integer m = 1500;
Integer n = 1500;
System.out.println(m==n) --> false, da m und n unterschiedliche Integer-Objekte mit verschiedenen Speicherplätzen sind. (Sorry, konnte das nur mit einem Beispiel erklären.)

j)
Generische Klassen verwenden formalte Typ-Parameter, die wie normale Bezeichner in Java aufgebaut sind. Er wird nach dem Klassennamen so angegeben: <T>
Durch diese Klassen muss man weniger Casts verwenden um Typkonversionen und Anpassungen zu erhalten. Man kann so Unterklassen verschiedener Typen für die selbe Oberklasse erstellen und sie die gleichen Methoden aufrufen lassen.

k)
Darunter versteht man, dass Unterklassen Methoden der Oberklasse überschreiben. Dazu müssen sie die gleiche Signatur haben.

l)
Exceptions sind unerwartete Fälle, auf die das Programm reagieren muss, da es sonst gestört wird und, wenn es nicht reagieren kann, abstürzt.
Runtime-Exceptions sind sog. "unchecked exceptions", bei denen der Übersetzer, anders als bei Exceptions, keine explizite Ausnahmebehandlung erfordert.