Übung 6 zur OOS SS16 Bearbeitung bis 28.04.2016

Aufgabe 21: (Serialisierung bei zyklischen Verweisen)

a) Definieren Sie 2 Klassen, die jeweils ein Attribut haben, das eine Referenz auf ein Objekt der anderen Klasse ist. Erzeugen und initialisieren Sie von jeder der beiden obigen Klassen jeweils ein Objekt, so dass die beiden erzeugten Objekte zyklisch aufeinander verweisen.

b) Serialisieren Sie das 1. dieser beiden Objekte, schreiben Sie es auf einen `ObjectOutputStream` und schließen Sie den Stream.

c) Erzeugen Sie für die Datei des `ObjectOutputStreams` aus b) einen `ObjectInputStream` und deserialisieren Sie das Objekt auf diesem Stream.

d) Zeigen Sie, dass durch die Deserialisierung genau 2 Objekte erzeugt wurden, deren Adressen sich von denen der beiden Objekte, die serialisiert wurden, unterscheiden und zyklisch aufeinander verweisen.

Aufgabe 22: (Serialisierung von mehreren Objekten und primitiven Typen)

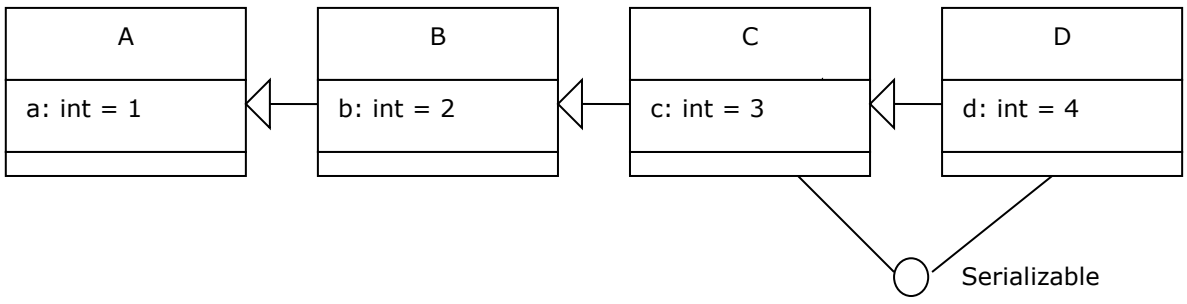
Schreiben Sie ein Java-Programm, das zunächst einen `int`-Wert, dann ein `String`- und schließlich zwei `Zeit`-Objekte serialisiert und auf einen `FileOutputStream` schreibt und diesen als Datei persistent macht.

Aufgabe 23: (Deserialisierung von mehreren Objekten und primitiven Typen)

Schreiben Sie ein Java-Programm, das die Datei aus Aufgabe 22 deserialisiert und die erhaltenen Werte ausgibt.

Aufgabe 26: (Konstruktoren bei Deserialisierung)

Gegeben sei folgendes UML-Klassendiagramm:



a) Implementieren Sie die Klassen `A – D` in Java. Implementieren Sie dabei jeweils den Standardkonstruktor so, dass er eine Meldung ausgibt.

Übung 6 zur OOS SS16 **Bearbeitung bis 28.04.2016**

b) Schreiben Sie in Klasse `D` ein `main`-Programm, das ein Objekt der Klasse `D` erzeugt und dessen Attribute auf verschiedene Werte setzt, die sich alle von den Initialwerten unterscheiden.

c) Implementieren Sie für die Klasse `D` die Standardmethode `toString()`, die alle Attributwerte von `D` ausgibt, und geben Sie damit das Objekt aus b) aus.

d) Serialisieren Sie das Objekt aus b).

e) Deserialisieren Sie die Datei aus d) zum Objekt mit Referenz `dNeu`.

f) Welche Werte haben die Attribute von `dNeu` und welche Konstruktoren werden dabei aufgerufen?

g) Überprüfen Sie Ihre Behauptung aus f), indem Sie `dNeu` ausgeben.

Aufgabe 42: (Aufzählungstypen)

Implementieren Sie eine Klasse `Ampel`, die eine Ampelsteuerung liefert. Gehen Sie hierzu wie folgt vor:

Beschreiben Sie die möglichen Farben der Ampel als Aufzählungstyp.

Der Initialzustand ist rot.

Implementieren Sie eine Methode `schalte`, die ausgehend vom aktuellen Zustand der Ampel den Nachfolgezustand liefert.

Aufgabe 43: (variable Parameterzahl, for each Schleife)

Implementieren Sie eine statische Methode, die eine variable Anzahl von Strings als Parameter erwartet und die Konkatenation (Aneinanderreihung) der Strings liefert. Wenn weniger als ein Argument vorhanden ist, werfen Sie folgende Exception `IllegalArgumentException`.

Verwenden Sie hierzu eine foreach-Schleife.

Aufgabe 52: (Überlagerung von equals)

Implementieren Sie die `equals`-Methode der Klasse `Auto` durch Überlagerung und erläutern Sie, in welchen Fällen diese Implementierung ausgeführt wird und in welchen Fällen die Implementierung von `equals` durch Überladung auf Folie „allg. Konzepte 13“ ausgeführt wird.

Aufgabe 53: (UML-Diagramm zu Praktikum 2)

Stellen Sie die Zusammenhänge zwischen den Klassen und Interfaces aus dem Versuch 2 des Praktikums in einem UML-Diagramm dar.

Übung 6 zur OOS SS16 Bearbeitung bis 28.04.2016

Design-Aufgabe 1:

a) Entwerfen Sie ein Design in UML, so dass die Datenstruktur eines beliebigen nicht leeren Baumes mit beliebigen Objekten als Knoteneinträgen implementiert wird.
Hierbei soll einem Nutzer dieser Datenstruktur die interne Struktur verborgen bleiben. Insbesondere soll der Nutzer nicht unterscheiden müssen, ob es sich um ein Blatt oder um einen Teilbaum handelt.
Der Nutzer soll also eine Referenz auf einen Baum erhalten, ohne ihn selbst erzeugen zu können.
Zur Verwendung des Baumes sollen dem Nutzer die folgenden Methoden zur Verfügung gestellt werden:
- `anzahlKnoten`, die die Anzahl der Knoten des Baumes ausgibt.
- `präfix`, die die Einträge der Knoten des Baumes in Präfixordnung liefert.

Entwerfen Sie so, dass jederzeit neue Methoden integriert werden können.

b) Geben Sie eine Implementierung Ihres Designs aus a) an.

Aufgabe 54: (Lineare Liste erweitern)

Erweitern Sie Ihre Implementierung der linearen Liste aus Aufgabe 41 um die Überlagerung der Standardmethoden `clone` und `equals`.

Implementieren Sie in einem anderen Paket eine Klasse, die obige Implementierung der linearen Liste verwendet und die folgenden statischen Methoden implementiert:

`append`: konkateniert 2 Listen

`attach`: hängt ein Objekt rechts an eine Liste

`reverse`: spiegelt eine Liste

`palindrom`: überprüft, ob eine Liste ein Palindrom ist.