

Name:

Matr.Nr.:

Ziel-Repository: <svn://www.ssw.uni-linz.ac.at/2009S/PSW2/<kMatrNr>/branches/ex3/>

Abzugeben: Quellcode, auf Papier und elektronisch.

Abgabefrist: 14. 12. 2009

Tutor:

Punkte:

Maximal: 100 Punkte

Reflection, XML, Serialization Objekt-Serialisierungs-API

In Java gibt es unterschiedliche Möglichkeiten Objekte zu Speichern:

- Objektserialisierung (Serializable, ObjectOutputStream, ObjectInputStream),
- XML-Bean-Serialisierung (XMLEncoder, XMLDecoder),
- manuelles Sichern der Daten (zB in Dateien eventuell als XML oder in Datenbanken) und
- die Verwendung der Java Architecture for XML Binding (JAXB).

Mit JAXB kann man Objekte in XML-Dokumente schreiben. Dazu muss man die zu serialisierenden Klassen annotieren und in einem JAXBContext bekannt machen. Mit Annotationen kann man die XML-Erzeugung beeinflussen, z.B. Properties umbenennen.

Aufgabe

In dieser Übung müssen Sie eine eigene Serialisierungs-API (*Anubis*) schreiben. Ziel ist, dass sie sich mit Reflection, XML, Serialisierung und API-Design auseinandersetzen.

Anubis soll folgende Eigenschaften haben:

- Primitive Datentypen (+String) können serialisiert werden.
- Komplexe Datentypen müssen annotiert werden, damit sie serialisiert werden können.
- Komplexe Datentypen müssen einen parameterlosen Konstruktor haben, oder eine Factory zur Erzeugung eines Standardobjekts muss registriert werden.
- Das Backend muss austauschbar sein; implementieren sie ein XML-, und ein CSV-Backend
- Nur direkt in der Klasse deklarierte Daten müssen serialisiert werden, Vererbung kann vernachlässigt werden
- Implementieren Sie JUnit-Tests zum Test des Systems

Punkte

Implementieren sie ihr Programm sauber strukturiert, damit sie keine Punkte durch „Kleinigkeiten“ verlieren.

- Schreiben von Objekten (30 Punkte)
- Lesen von Objekten (30 Punkte)
- XML Backend (15 Punkte)
- CSV Backend (10 Punkte)
- JUnit-Tests (15 Punkte)

Architektur

In diesem Abschnitt schlage ich eine Schnittstelle für API-Nutzer und Backend-Implementierer vor. Beschreiben und begründen Sie Abweichungen von der vorgeschlagenen API, damit sich der Tutor in ihrem Code schnell zurechtfindet.

Schnittstelle für Nutzer:

```
class AnubisWriter
```

- static AnubisWriter create(AnubisSink sink);
erzeugt einen neuen AnubisWriter mit gegebenem Backend
- void write(Object source)
schreibt das gegebene Objekt

```
class AnubisReader
```

- static AnubisReader create(AnubisSource source)
erzeugt einen AnubisReader mit der gegebenen Quelle
- Object read()
liest das serialisierte Objekte ein
- <T> void setFactory(Class<T> forClass, AnubisFactory<T> factory)
registriert eine Factory für Standardobjekte der gegebenen Klasse

```
interface AnubisFactory<T>
```

- T create()
erzeugt ein Standardobjekt der Klasse T

```
@interface Anubis
```

Annotation zum Markieren von Feldern und Methoden (Properties) die serialisiert werden sollen.

Schnittstelle für Backend-Implementierer:

```
class AnubisElement
```

verwaltet einen Teil eines Objekts (z.B. ein Property)

```
interface AnubisSink
```

- void write(AnubisElement)
schreibt ein Element in das Backend

```
interface AnubisSource implements Iterator<AnubisElement>
```

- AnubisElement read()
liest ein Element

Beispiel

In diesem Abschnitt sehen sie wie eine Klasse annotiert werden muss, damit sie mit Anubis serialisiert werden kann. Und eine mögliche XML-Ausgabe für ein Objekt der Klasse.

```
@Anubis
public class Car {
    @Anubis
    private int maxSpeed;
    private int speed;
    @Anubis
    private Person owner;

    public Car(int maxSpeed, Person owner) {
        this.maxSpeed = maxSpeed;
        this.owner = owner;
    }
    @Anubis(setter="accelerate")
    public int getSpeed() {
        return speed;
    }
    public void accelerate(int deltaSpeed) {
        speed += deltaSpeed;
    }
}

public CarFactory implements AnubisFactory<Car> {
    public Car create() {
        return new Car(0, null);
    }
}
```

```
Car c = new Car(222, null);
c.accelerate(123);
```

```
<anubis>
  <element id="1" kind="CLASS" access="" name="" target=""
    type="at.jku.ssw.anubis.test.Car" value="">
  </element>
  <element id="2" kind="PRIMITIVE" access="FIELD" name="maxSpeed"
    target="1" type="int" value="&quot;222&quot; ">
  </element>
  <element id="3" kind="CLASS" access="FIELD" name="owner"
    target="1" type="" value="">
  </element>
  <element id="4" kind="PRIMITIVE" access="METHOD" name="accelerate"
    target="1" type="int" value="&quot;123&quot; ">
  </element>
</anubis>
```