Serialisierung von Objekten und Zuständen

Carsten Gips (HSBI)

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

Motivation: Persistierung von Objekten und Spielzuständen

```
public class Studi {
    private final int credits = 42;
    private String name = "Hilde";
    ...
}
```

Wie kann ich Objekte speichern und wieder laden?

Serialisierung von Objekten

- Klassen müssen Marker-Interface Serializable implementieren
- Schreiben von Objekten (samt Zustand) in Streams

```
ObjectOutputStream: void writeObject(Object)
```

• Lesen und "Wiedererwecken" der Objekte aus Streams

```
ObjectInputStream: Object readObject()
```

Einfaches Beispiel

```
public class Studi implements Serializable {
   private final int credits = 42;
   private String name = "Hilde":
   public static void writeObject(Studi studi, String filename) {
        try (FileOutputStream fos = new FileOutputStream(filename);
            ObjectOutputStream oos = new ObjectOutputStream(fos)) {
           oos.writeObject(studi); oos.close();
        } catch (IOException ex) {}
    public static Studi readObject(String filename) {
        Studi studi = null;
        try (FileInputStream fis = new FileInputStream(filename);
           ObjectInputStream ois = new ObjectInputStream(fis)) {
           studi = (Studi) ois.readObject(); ois.close();
        } catch (IOException | ClassNotFoundException ex) {}
        return studi:
```

Bedingungen für Objekt-Serialisierung

- Klassen implementieren Marker-Interface Serializable
- Alle Attribute m

 üssen ebenfalls serialisierbar sein (oder Deklaration "transient")
- Alle primitiven Typen sind per Default serialisierbar
- Es wird automatisch rekursiv serialisiert, aber jedes Objekt nur einmal (bei Mehrfachreferenzierung)
- Serialisierbarkeit vererbt sich

Ausnahmen

- Als static deklarierte Attribute werden nicht serialisiert
- Als (transient) deklarierte Attribute werden nicht serialisiert
- Nicht serialisierbare Attribut-Typen führen zu NotSerializableException

Version-UID

```
static final long serialVersionUID = 42L;
```

- Dient zum Vergleich der serialisierten Version und der aktuellen Klasse
- Über IDE generieren oder manuell vergeben
- Wenn das Attribut fehlt, wird eine Art Checksumme von der Runtime-Umgebung berechnet (basierend auf diversen Eigenschaften der Klasse)

Demo: serial.SerializableStu

Wrap-Up

- Markerinterface Serializable schaltet Serialisierbarkeit frei
- Objekte schreiben: ObjectOutputStream: void writeObject(Object)
- Objekte lesen: ObjectInputStream: Object readObject()
- Wichtigste Eigenschaften:
 - Attribute müssen serialisierbar sein
 - transient und static Attribute werden nicht serialisiert
 - De-Serialisierung: KEIN Konstruktor-Aufruf!
 - Serialisierbarkeit vererbt sich
 - Objekt-Referenz-Graph wird automatisch beachtet

LICENSE



Unless otherwise noted, this work is licensed under CC BY-SA 4.0.