

---

**Programmieren in Java**
<http://proglang.informatik.uni-freiburg.de/teaching/java/2013/>


---

**Java-Übung Blatt 10 (Generics, Exceptions & I/O Streams, Serialisierung)**

2013-07-02

**Hinweise**

- Schreiben Sie Identifier *genau so*, wie sie auf dem Blatt stehen (inklusive Groß- und Kleinschreibung), nicht nur ungefähr.
- Identifier und Kommentare bitte auf *englisch*!
- Schreiben Sie *sinnvolle* Kommentare
- Laden Sie Ihre Lösungen mit subversion (svn) ins Übungssystem hoch. Den entsprechenden Pfad finden Sie online.
- Das Übungssystem kann überprüfen, ob Sie Ihr Quelltext den Anforderungen genügt und ob Sie alle Klassen erstellt haben, etc. Nutzen Sie dies!
- Sollte das Übungssystem Ihre Lösungen ablehnen, dann werden sie *nicht* korrigiert! Akzeptanzkriterien:
  - Compiliert erfolgreich
  - Checkstyle bringt keine Fehler
  - Alle Packages, Klassen, Interfaces, Methoden, Typen, Argumente sind exakt wie auf dem Übungsblatt gefordert.
- Ihr korrigierender Tutor wird die Korrektur Ihrer Abgabe in Ihr svn-Repository unter dem Namen `Feedback-<login>-ex<XX>.txt` comitten. (<login> ist dabei Ihr myAccount name und <XX> die Kennziffern des Übungsblatts)
- Zusätzlich können Sie Ihre Gesamtpunktzahl im Übungsportal einsehen.

Abgabe: Freitag, 12. Juli 2013, um 23.59 Uhr.

**Aufgabe 1 (Paare und Ordnung, 11 Punkte)**

Projekt: `ex10_1`. Package: `pairs`. In der Vorlesung wurden ja bereits als Beispiel für Generics ein Datentyp für sogenannte Paare gezeigt. Ein Paar ist ein Objekt, das genau zwei andere Objekte zusammenfasst. Im Unterschied zu einer Collection können die beiden Objekte aber unterschiedliche (auch nicht-verwandte) Typen haben.

1. (2 Punkte) Definieren Sie den Paar-Typ als eine generische Klasse `Pair`. Ihr Konstruktor erwartet zwei Objekte beliebigen Typs und man kann mit den Methoden `fst()` und `snd()` auf das Erste bzw. Zweite der übergebenen Objekte zugreifen. Implementieren Sie außerdem geeignete `hashCode` und `equals` Methoden.
2. (3 Punkte) Schreiben Sie eine Klasse `CompPair` die von `Pair` ableitet und zusätzlich das `Comparable` Interface implementiert. Stellen Sie sicher, dass Code wie der folgende compiliert:

---

```

1 package pairs;
2
3 interface Vehicle extends Comparable<Vehicle> {}
4 interface Car extends Vehicle {}
5 interface Train extends Vehicle {}
6
7 public class CompileMe {
8     public static void test(Car c1, Car c2, Train t1, Train t2) {
9         new CompPair<Car, Train>(c1, t1)
10             .compareTo(new CompPair<Car, Train>(c2, t2));
11     }
12 }
```

---

Beachten Sie außerdem die in der Vorlesung vorgestellten Einschränkungen und Konventionen für `Comparable`.

Definieren Sie auch einen Konstruktor für `CompPair` dem man gewöhnliche Pairs übergeben kann, sofern deren Komponenten vergleichbar sind.

Testen Sie in ihren Unit-Tests auch die korrekte „Zusammenarbeit“ mit den Collection Klassen; verwenden Sie dazu z.B. die Collection-Klasse `TreeSet` (welche Mengen aus geordneten Elementen implementiert) und überprüfen Sie, ob die typischen Mengen-Operationen wie erwartet funktionieren und ob der Iterator des `TreeSets` die Elemente in der richtigen Reihenfolge ausgibt.

3. (3 Punkte) Paare können im Rückgabetypp von sog. zip-Operationen verwendet werden: Eine zip-Funktion nimmt zwei `Collections` als Argument und gibt eine `Collektion` aus Paaren als Ergebnis zurück. Die Paare des Ergebnisses bestehen genau aus den kombinierten Elementen der Argument-`Collections`. Falls die Argument-`Collections` eine „Reihenfolge“ der Elemente vorgeben, soll diese im Ergebnis eingehalten werden. Falls die Argument-`Collections` ungleicher Länge sind, soll das Ergebnis die Länge des kürzeren Arguments haben. Beispiel in Pseudocode:

```
zip([1,2,3,4,5],["Hallo", "Welt", "wie", "geht's"])
==
[(1, "Hallo"), (2, "Welt"), (3, "wie"), (4, "geht's")]
```

Implementieren sie die zip-Funktion als generische, statische Methode `zip` der Klasse `pairs.Zip`.

4. (3 Punkte) Implementieren Sie ein alternative Version von `zip`, bei der man über einen zusätzlichen Parameter angeben kann, von welchem `Collection`-Typ das Ergebnis sein soll. Für diesen Zusatz-Parameter werden Sie ein Interface definieren müssen, das angibt wie man eine `Collection` von entsprechendem Typ baut. Nutzen Sie Generics damit dieser Typ auch statisch als Rückgabewert auftritt.

Das folgende Beispiel demonstriert, was mit der oben beschriebenen Funktion möglich sein soll (Pseudocode):

---

```
1 ArrayList<Pair<Integer,String>> zipResult1 =
2   zip(new ArrayListBuilder(), Arrays.asList(1,2,3), Arrays.asList("4","5","6"));
3 Set<Pair<Integer,String>> zipResult1 =
4   zip(new SetBuilder(), Arrays.asList(1,2,3), Arrays.asList("4","5","6"));
```

---

Testen Sie ihre Implementierungen mit JUnit. Die obigen Aufgaben können und sollen ohne die Verwendung von Typcasts gelöst werden.

## Aufgabe 2 (Serialisierung, 11 Punkte)

Projekt: `ex10_2`. Package: `monopoly`.

Im Template finden Sie den aktuellen Stand des in der Vorlesung entwickelten Spieles Monopoly. Da es sich bei Monopoly ja meistens um ein längeres Spiel handelt, fügen wir nun Funktionalität hinzu, die es erlaubt den Spielstand zu speichern.

Wir verwenden dazu die in der Vorlesung vorgestellte Serialisierung mit XStream. Eine Anleitung wie Sie XStream installieren und verwenden, finden Sie auf der Tools-Seite der Vorlesung<sup>1</sup>.

1. Schreiben Sie eine Klasse `GameState` welche alle Werte, die zur Rekonstruktion des Spielstandes notwendig sind, aus einem Game-Objekt und den referenzierten Objekten extrahiert und als entsprechende Instanzvariablen hält. Vermeiden Sie das Zwischenspeichern von Werten, die zur Rekonstruktion nicht erforderlich sind, z.B. Straßenpreise oder Farben.

Diese `GameState` Klasse sollte auf keinen Fall eine Referenz auf ein Game-Objekt enthalten, da dies bei der Serialisierung sonst auch mitgespeichert wird!

Implementieren Sie `equals` und `hashCode` für `GameState`.

2. Erweitern Sie `GameState` um eine Methode `createGame`.

---

<sup>1</sup><http://proglang.informatik.uni-freiburg.de/teaching/java/2013/tools.html>

---

```

1 /**
2  * Creates a game from the GameState object.
3  * @return The game created from the GameState.
4  */
5 public Game createGame();

```

---

Erzeugen Sie innerhalb dieser Methode ein neues Game-Objekt, welches in den entsprechenden Zustand wechselt. Erweitern Sie dazu die entsprechenden Klassen (Game, ...) um einen Konstruktor, der die Klasse mit den Werten aus einem GameState-Argument initialisiert.

3. Schreiben Sie eine Klasse PersistenceManager, welche den Spielstand (GameState) aus einem Spiel (Game) extrahiert und den GameState dann mit XStream in einen Writer speichert.

Außerdem soll diese Klasse auch eine XML-Beschreibung eines Zustands (GameState) mit XStream aus einem Writer lesen und mit createGame ein Spiel erzeugen welches zurückgegeben wird. Es soll folgendes Interface haben:

---

```

1 /**
2  * Saves the current state of the game to the writer.
3  *
4  * @param w The writer to write to.
5  */
6 public void saveGame(Game g, Writer w);
7
8 /**
9  * Restores a game from a state provided by a reader.
10 * @param r Reader to read state from.
11 * @return The game class with the new state.
12 */
13 public Game loadGame(Reader r);

```

---

Testen Sie Ihre Implementierung mit JUnit. Sie können wie in der Vorlesung vorgestellt einen StringReader bzw. StringWriter verwenden um Ihre Implementierung zu testen.

### Aufgabe 3 (Finally und Streams, 11)

Projekt: `ex10_3`. Package: `copy`.

Implementieren Sie ein Programm, welches alle Dateien aus einem Quell-Verzeichnis in ein Ziel-Verzeichnis kopiert. Das Programm soll interaktiv über einen Prompt Anweisungen entgegen nehmen. Der Fokus liegt bei dieser Aufgabe auf der korrekten Behandlung der Fehler, die dabei auftreten können.

Dies ist ein Beispiellauf des Programms, wenn es ohne Argumente aufgerufen wird (alles nach dem Prompt > bis zum Ende der Zeile sind Benutzereingaben):

```

Welcome. Type ":q" to quit.
SRC-DIR > /home/fennell/Desktop/tmp-files
DEST-DIR > /home/fennell/Desktop/tmp-files-dest
copying "/home/fennell/Desktop/tmp-files/1.txt", 00% (0% total)
copying "/home/fennell/Desktop/tmp-files/1.txt", 50% (25% total)
copying "/home/fennell/Desktop/tmp-files/X.txt", 00% (25% total)
Error: Permission denied!
copying "/home/fennell/Desktop/tmp-files/2.txt", 00% (50% total)
copying "/home/fennell/Desktop/tmp-files/2.txt", 99% (99% total)
Finished! Success: 2 files, Error: 1 files
SRC-DIR > :q
Bye!

```

Beachten Sie bei der Implementierung folgende Punkte:

- Alle für diese Aufgabe notwendigen Datei-Operationen werden von Objekten der Klasse `java.io.File` zur Verfügung gestellt.
- Verzeichnisse und andere Spezialdateien müssen nicht kopiert werden. Kopieren Sie die Standarddateien, die direkt im angegebenen Verzeichnis liegen.
- Wenn beim Kopieren einer Datei behandelbare Fehler auftreten, geben Sie eine sinnvolle Fehlermeldung aus und fahren Sie mit der nächsten Datei fort. Am Ende soll, wie oben angedeutet, die Anzahl der erfolgreichen und fehlgeschlagenen Datei-Kopiervorgänge ausgegeben und der Prompt angezeigt werden. Dann soll ein neuer Verzeichnis-Kopiervorgang gestartet werden können.
- Es sollen immer nur die Dateien geöffnet sein, aus denen gerade gelesen oder geschrieben wird.
- Während dem Kopieren soll der Fortschritt der einzelnen Datei sowie der Gesamtfortschritt, jeweils in Prozent von Bytes, nach und nach angezeigt werden.
- Ermöglichen Sie zu Testzwecken, dass der konkrete I/O-Stream, der zum Kopieren benutzt wird, sowie die Operationen auf dem Dateisystem durch Mock-Implementierungen ersetzt werden können. Dokumentieren Sie dies, so dass ihr Tutor versteht, wie er diese Mock-Implementierungen angeben kann. Es wird stark empfohlen, dass Sie auch eigenen JUnit-Tests schreiben. Um Zeitverzögerungen beim Kopieren zu simulieren, können sie die statische Methode `Thread.sleep(millis)` verwenden. (Die dabei mögliche `InterruptedException` muss nicht behandelt werden)
- Sie brauchen keine besonderen Vorkehrungen zu treffen um „ungewöhnliche“ und relative Dateinamen zu unterstützen. Entfernen Sie aber überflüssige Leerzeichen aus der Benutzereingabe mittels der `String` Methode `trim`.