Praktikum Programmiermethodik 2 (Technische Informatik)
WS 2015/2016, Hochschule für Angewandte Wissenschaften (HAW), Hamburg
Prof. Dr. Philipp Jenke, Kasperczyk-Borgmann

Für dieses Aufgabenblatt gelten die folgenden Regeln:

- Der Java Code Style (siehe EMIL) ist einzuhalten. Es werden keine Abgaben abgenommen, die diese Anforderungen nicht erfüllen.
- Alle Programme für dieses Aufgabenblatt müssen in dem Package `aufgabenblatt1` liegen.
- Pro Team muss ein gemeinsames Git-Repository für das Projekt vorhanden sein (z.B. auf GitHub, Bitbucket oder dem HAW-Informatik-Home-Verzeichnis).
- Der Code muss vollständig getestet sein.

Änderungshistorie:

- 16.10.2015: Hinweise zum Rückgabetyp von Methoden im Klassendiagramm und zum Ignorieren von Typ-Cast-Warnungen hinzugefügt.

Aufgabenblatt 1: Git, Wiederholung PM1/PT, XML/JSON, Generics

Aufgabe 1.1: Git
Lernziele: Umgang mit Git (Kommandozeilenparameter)
Aufgabe: Sie müssen in der Lage sein, in Ihrem Projekt mit Git (Kommandozeilen-Befehle) zu arbeiten: z.B. zwischen dem lokalen und dem entfernten Repository zu synchronisieren

Aufgabe 1.2: Wiederholung PM1/PT
Lernziele: Umgang mit dem `Collections`-Framework, Vergleichen in Java, Basisklasse `Object` verwenden
Aufgaben:

- Schreiben Sie eine Klasse `Student`. Ein Student besteht aus folgenden Informationen: Vorname, Nachname, Matrikelnummer (Ganzzahl), Liste von Prüfungsleistungen.
- Prüfungsleistungen sind ein eigener Typ `Pruefungsleistung`, der wiederum den Namen des Moduls und die Note beinhaltet.
- Machen Sie Studenten vergleichbar, indem Sie das Interface `Comparable<T>` implementieren. Sortiert werden soll anhand der Matrikelnummer.
- Manchmal ist auch der Vergleich nach Nachname, Vorname gewünscht. Setzen Sie dies durch einen geeigneten `Comparator<T>` um.

Aufgabe 1.3: XML
Lernziele: JSON-Dokumente lesen, XML-Dokumente lesen und schreiben, XML-DTD erstellen
Aufgabe:

- Gegeben ist folgendes JSON-Dokument:

```
{
    "name": "Raspberry Pi - Model b",
    "ip": "192.168.2.1",
    "sensoren":[
        {
            "ort": "Wohnzimmer",
            "typ": "Temperatur",
            "wert": 22.4
        },
        {
            "ort": "Küche",
            "wert": -1.2323
        },
        {
            "ort": "Badezimmer",
            "typ": "Luftfeuchtigkeit",
            "wert": 63.8
        }
    ]
}
```

- Erstellen Sie ein XML-Dokument, das die gleichen Informationen repräsentiert.
- Entwerfen Sie eine XML-DTD, die das XML-Dokument erfüllt und verwendet.
- Hinweis: Es gibt verschiedenen korrekte Lösungen.

Aufgabe 1.4: Generics

Lernziele: Erstellen generischer Typen, Arbeiten mit Typebounds und Wildcards,
Aufgabe:

- Schreiben Sie eine eigene Implementierung von `ArrayListe<T>`, die intern ein Array verwendet und folgende Schnittstelle bietet:

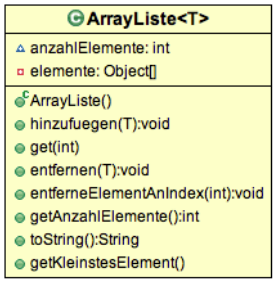

Abbildung 1: Klassendiagramm von `ArrayListe`. Die Methoden `get()` und `getKleinstesElement()` sollen Werte vom Typ `T` zurückliefern.

- Sollten sich durch erforderliche Type-Casts Compiler-Warnungen ergeben, ist das in dieser Aufgabe ausnahmsweise in Ordnung.
- Schreiben Sie eine statische Methode (außerhalb der Klasse `ArrayListe<T>`), die für eine beliebige Liste prüft, ob das erste Element (wenn es mindestens eins gibt) eine Zahl ist.
- Was können Sie mit den Elementen in der Klasse `ArrayListe<T>` machen, wenn Sie als UpperBound `Comparable<T>` verwenden. Wie müssen Sie dazu die Klasse verändern? Entwerfen Sie ein Beispiel und implementieren Sie es.
- Schreiben Sie eine Methode, die für eine `ArrayListe<T>` mit `Integer`-Werten die Summe der Zahlen berechnet und zurückliefert.