Collections

Es ist sicherzustellen, dass stets Typstellvertreter gesetzt sind. Dazu hilft es, die Compileroption `-Xlint:unchecked` einzuschalten. In Netbeans:
Rechtklick auf das Projekt `-> Properties -> Build -> Compiling -> Additional Compiler Options`.
Der Compilertyp einer `Collection` sollte stets ein Interface sein.

1 PhonebookEntry

Empfohlene Reihenfolge: `PhonebookEntry` anlegen, Methoden nur soweit implementieren, dass sie kompilieren und dann zuerst den Test schreiben. Gegeben ist das Fragment einer Javaklasse für Telefonbucheinträge:

```
public class PhonebookEntry {
    private String number;
    private String name;
}
```

Zwei Telefonbucheinträge gelten als gleich, wenn Sie die gleiche Nummer haben (die Namen sind dabei unwesentlich).

- Der Konstruktor sollte Telefonnummern akzeptieren, welche entweder nur aus Zahlen bestehen oder mit `+L` beginnt wobei `L`$\neq$`0`. Andere Nummern sollten mit einer `IllegalArgumentException` abgelehnt werden.
- Implementiere entsprechende Methoden `equals` und `hashCode`.
- Implementiere in der Klasse eine natürliche Sortierreihenfolge so, dass aufsteigend nach den Telefonnummern sortiert wird.
- Lege eine geeignetes Datenstruktur vom Typ `PhonebookEntry` an, schreibe Testdaten hinein und sortiere
  - nach der natürlichen Reihenfolge ihrer Elemente
  - in der umgekehrten natürlichen Sortierreihenfolge (hilfreiche Methode `reverseOrder` in `java.util Collections`; kein Unit-Testing nötig)
  - alphabetisch nach Namen und bei gleichen Namen absteigend nach der Telefonnummer(kein Unit-Testing nötig)

Gib für jede Sortierreihenfolge die Ausgabe an.

2 Test

Entwickle einen Unit-Test für `PhonebookEntry`.

3 Student

Implementiere eine Klasse `Student`, die im Konstruktor den Familiennamen, den Vornamen die Matrikelnummer[1] eines Studenten erhält.
Sollte ein Student mit gleicher Matrikelnummer bereits existieren, so ist im Konstruktor eine

[1] Eindeutige Nummer jedes Studenten, ähnlich wie die Aufnahmenummer

Exception zu werfen. Zu diesem Zweck müssen alle existierenden Matrikelnummern in einer geeigneten statischen Collection der Klasse Student gespeichert sein.
Mit folgendem Programmfragment, welches mit der Klasse funktionieren soll, kann man mehrere Studenten (zuerst nach dem Nachnamen und danach nach dem Vornamen) sortiert ausgeben:

```
Student s1 = new Student("Muster", "Thomas", 123456);
Student s2 = new Student("Herbert", "Franz", 111111);
Student s3 = new Student("Malt", "David R.", 323232);
new Student("", "", 111111); //Exception
// Studenten in geeigneter Collection speichern
// Studenten ausgeben
```

Ergänzen Sie auch das oben angegebene Programmfragment. Erwartete Ausgabe:

```
Herbert, Franz (111111)
Malt, David R. (323232)
Muster, Thomas (123456)
```