Kontrollfragen A

1. Was für Vorteile bringt eine festgelegte Ordnung mit sich?
   Sie ermöglicht ein Einfaches vergleichen, sortieren. Dadurch lassen sich Daten auch schnell „finden“.

2. Denken Sie sich verschiedene Klassen. Wie steht es mit ihrer natürlichen Ordnung? Können Sie auch spezielle Ordnungen ausmachen?
   Natürliche Ordnung: -2 < -1 < 0 < 1 < 2…
   Spezielle Ordnung: „eigen Kreationen“ z.B.: 1, 7, 2, 0
   Tagesablauf: Zeit / Aufgabe
   Prioritäten: Wichtiges / Dringendes
   Handy-Vertrag

Kontrollfragen B

1. Durch welches Interface lässt sich eine natürliche Ordnung festlegen?
   Comparable<T>

2. Wie viele Methoden spezifiziert dieses Interface?
   Nur eine Methode:
   ```
   int compareTo(T other)
   ```

3. Wie ist der Rückgabewert von `compareTo()` zu interpretieren?
   Negativer Wert: „aktuelles Objekt ist kleiner als das andere Objekt (other)“
   0: „aktuelles Objekt ist gleich wie das andere Objekt (other)“
   Positiver Wert: „aktuelles Objekt ist grösser als das andere Objekt (other)“

4. Was ist im Zusammenhang mit `compareTo()` und `equals()` zu beachten?
   Gleichheit sollte analog behandelt werden (Gleiche Attribute auf die gleiche Art vergleichen).
   Andernfalls muss es ganz klar dokumentiert werden!

5. Wie muss man auf der letzten Folie `compareTo()` ändern, damit die Ballon-Texte umgekehrt ausgegeben werden?
   ```
   return other.text.compareTo(this.text);
   ```

Kontrollfragen C

1. Durch welches Interface kann man eine spezielle Ordnung erreichen?
   Comparator<T>

2. Die Methode `compare()` dieses Interfaces besitzt 2 Parameter. Was ist deren Bedeutung?
   Die zwei zu vergleichenden Objekte.
   Comparator ist eine eigene Klasse um eigene / spezielle Ordnung zu definieren. Mittels den Parametern werden die zu vergleichenden Objekte übergeben.

3. In der Klasse `Balloon` musste neu eine Methode `getSize()` implementiert werden. Weshalb?
   size ist private und kann somit von „aussen" nicht abgefragt werden.
   Damit Comparator den Wert für den Vergleiche verwenden kann, muss eine getter-Methode implementiert werden.

4. Verifizieren Sie die Terminal-Ausgaben nochmals genau.
   done.

5. Wozu dient die Methode `equals()` der Klasse `SizeUpComparator`?
   Comparator Klassen vergleichen
   Vergleich ob verschiedene Comparator Klassen auf die gleiche „Art" vergleichen / sortieren.

6. Wieso soll man ein Objekt zuerst aus einer geordneten Datenstruktur entfernen, bevor man es verändert?
   Damit die Ordnung weiterhin stimmt.
   Würde sie belassen und geändert könnte die Ordnung durcheinander kommen.
   Wir es Objekt entfernt, bearbeitet und neu eingefügt, wird es beim Einfügen an der richtigen Position eingefügt und somit stimmt die Ordnung weiterhin.