1. Natural-Order in Java

Aufgabe: Zeigen Sie auf, wie Java die Anwendung der Natural-Order einer Klasse beim Sortieren anwendet.

Die Funktion `Comparator.naturalOrder()` gibt eine Comprator zurück, der das Interface `Comparable` verwendet.

```
public int compare(Comparable<Object> c1, Comparable<Object> c2) {
    return c1.compareTo(c2);
}
```

Im List.sort() wird dan ein Timsort gemacht, und die Elemente werden mithilfe von `compare` verglichen und sortiert. Un wie zuvor aufgezeigt, ist `compare` nur ein Comparator wrapper um die `compareTo` Methode der Klasse, die implementiert ist durch das Interface `Comparable`. Das Interface `Comparable` ist eine Java-Schnittstelle, die es Klassen ermöglicht, ihre natürliche Ordnung zu definieren.

2. Reverse-Order für Klassen

Aufgabe: Zeigen Sie auf, wie Java die Reverse-Order an eine Klasse anwendet.

Die Funktion `Comparator.reversed()` gibt einen Comparator zurück, der den existierenden Comparator umkehrt, in dem er einen neuen Comparator erstellt, der die `compare`-Methode des ursprünglichen Comparators umkehrt.

Als Beispiel nehmen wir mal `list.sort(Comparator.<OneTimeEvent>naturalOrder().reversed());` welches die Natürliche Ordnung der Klasse `OneTimeEvent` umkehrt.

Wie wir zuvor gesehen haben, wird `compare()` verwendet, um die natürliche Ordnung der Klasse zu erhalten, welche `compareTo()` auf der Klasse `OneTimeEvent` aufruft. Logischerweisse macht man `c1.compareTo(c2)` um die natürliche Ordnung zu erhalten. Java hat nun für Reverse-Order im ReversedComparator die `compare`-Methode umgekehrt, in dem sie `c2.compareTo(c1)` aufruft. Dadurch wird die natürliche Ordnung umgekehrt.

```
public int compare(Comparable<Object> c1, Comparable<Object> c2) {
    return c2.compareTo(c1);
}
```

Für die Reverse-Order gibt es auch eine statische Methode `Comparator.reverseOrder()`, die genau dasselbe macht wie `Comparator.naturalOrder().reversed()`, aber ohne einen existierenden Comparator zu benötigen. Ist als einfach ein shortcut.

3. Wie sortiert Java

Aufgabe: Tauchen Sie ein in das Sortieren von Java. Wie sortiert Java?

Java converted alle arten von Lists und weiteres in ein Array und verwendet dann die existierenden lösungen von Arrays `Arrays.sort()`, um die Elemente zu sortieren. Dabei gibt er der static Methode `Arrays.sort()` einen Comparator mit.

Wie bereits zuvor erwähnt, verwendet Java unteranderem einen Timsort Algorithmus, um die Elemente zu sortieren. Im sort Algorithmus wird der Comparator verwendet, um die Elemente zu vergleichen und zu sortieren.

Hier ein Beispiel aus dem Timsort code wie der Comparator verwendet wird: `c.compare(pivot, a[mid])` wir sehen also, das Java einfach die `compare`-Methode des Comparators aufruft, um die Elemente zu vergleichen.

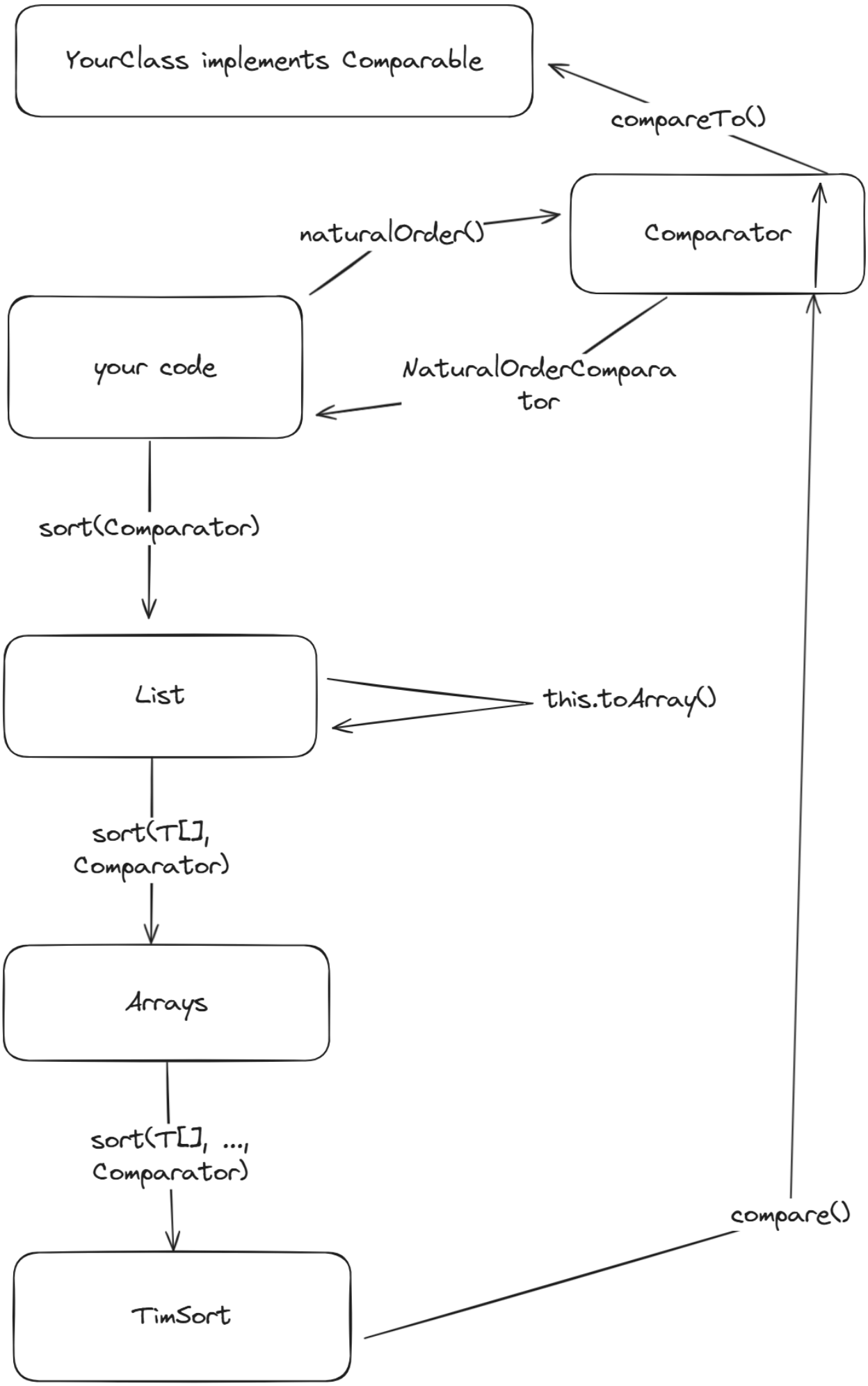
4. Sortieren in Java

Aufgabe: Zeigen Sie auf, wie Java das Sortieren löst.

Das meiste wurde bereits in den vorherigen Aufgaben erklärt. Java macht in der `List.sort()` Methode nichts anderes, also sich selber zu einem Array zu konvertieren und dann die `Arrays.sort()` Methode aufzurufen, mit dem gegebenen Comparator.

Folgendes Diagram zeigt, wie Java das Sortieren löst. Es ist etwas vereinfacht. Heisst ich habe gewisse Wrapper Methoden Calls weggelassen, oder sehr technische Sorting Algorithmen details nicht aufgezeigt. Es geht hierbei hauptsächlich um den groben Überblick, wie Java das Sortieren löst, mithilfe von Comparators.

```
// Code zum Diagram
events.sort(Comparator.naturalOrder());
```


YourClass implements Comparable
compareTo()
naturalOrder()
Comparator
your code
NaturalOrderComparator
sort(Comparator)
List
this.toArray()
sort(T[], Comparator)
Arrays
sort(T[], ..., Comparator)
compare()
TimSort