Java Notes 3

Table of Contents

[1 Java Notes 2](#_Toc104097679)

[1.1 Java natural ordering 2](#_Toc104097680)

[1.2 Boxed Values, unboxed Values and Equality 3](#_Toc104097681)

1. Java Notes
   1. Java natural ordering

A **List l** may be sorted as follows.

Table

Description automatically generated Collections.sort(l);

If the List consists of String elements, it will be sorted into alphabetical order. If it consists of Date elements, it will be sorted into chronological order. How does this happen? String and Date both implement the Comparable interface.

Comparable implementations provide a ***natural ordering*** for a class, which allows objects of that class to be sorted automatically. The following table summarizes some of the more important Java platform classes that implement Comparable.

Elements that can be compared to one another are called   
***mutually comparable***. Although elements of different types may be mutually comparable, none of the classes listed here permit interclass comparison.

Text

Description automatically generated

* This is the values which should be returned, depending on the condition

You can sort a collection which implemented **Comparable**, or you can pass in your **Comparator** to provide / override the sorting logic.

Text

Description automatically generated

Strings are compared **Lexicographically**:





* If a Type does not implement Comparable<T> then Collections.sort will not compile
  1. Boxed Values, unboxed Values and Equality

**Boxed values** (**Integer**, **Long**, **Double**, **Character**) should not be compared with **==** because you're comparing them as references, not as values.

* == is the reference comparison operator. It compares the memory addresses of the objects
* If two variables point at different objects, they will not == each other, even if the objects represent the same value.
* When comparing primitives with **==**, there are no memory addresses in play. It's a simple instruction to compare whether a value is equal to another value

**TODO**: modifying a collection while iterating over it. What are the best practices to do so?