Comparator vs Comparable in Java

During interviews, you can face this question differently, if you are giving a telephonic round then it's mostly fact-based i.e. you need to mention key points about both interfaces, while in the face to face interviews or during the written test, you might be asked to write code to sort objects using Comparator and Comparable e.g. sorting employee by name or branch. I have already discussed the second part of this question here and we will only see fact-based differences in this post.

1. Comparator interface is in `java.util` package, which implies it's a utility class, while Comparable interface is kept on `java.lang package`, which means it's essential for Java objects.

2. Based on syntax, one difference between Comparable and Comparator in Java is that former gives us `compareTo(Object toCompare)`, which accepts an object, which now uses Generics from Java 1.5 onwards, while `Comparator` defines `compare(Object obj1, Object obj2)` method for comparing two object.

3. Continuing from the previous **difference between Comparator vs Comparable** later is used to compare current object, represented by this keyword, with another object, while `Comparator` compares two arbitrary objects passed to `compare()` method in Java.

4. One of the key difference between `Comparator` and `Comparable` interface in Java is that, You can only have one `compareTo()` implementation for an object, while you can define multiple `Comparator` for comparing objects on different parameters e.g. for an `Employee` object, you can use compareTo() method to compare `Employees` on id, known as natural ordering, while multiple `compare()` method to order employee on age, salary, name and city. It's also a best practice to declare Comparator as nested static classes in Java, because they are closely associated with objects they compare.

5. Many Java classes, which make uses of `Comparator` and `Comparable` defaults to `Comparable` and provided overloaded method to work with arbitrary `Comparator instance` e.g. `Collections.sort()` method, which is used to sort Collection in Java has two implementations, one which sort object based on natural order i.e. by using `java.lang.Comparable` and other which accepts an implementation of `java.util.Comparator` interface.

6. One more key thing, which is not a difference but worth remembering is that both `compareTo()` and `compare()` method in Java must be consistent with equals() implementation i.e. if two methods are equal by equals() method then `compareTo()` and `compare()` must return zero. Failing to adhere to this guideline, your object may break invariants of Java collection classes which rely on `compare()` or `compareTo()` e.g. TreeSet and TreeMap.