Java

Compa >< rators

Tips

Tricks

Insights

Follow Sunit Ghosh For More...

Can this even happen?

If you are thinking, **Never...**

```java
import java.util.*;
import java.util.stream.*;

public class StringBuilderInHashMap {
    public static void main(String[] args) {
        List<StringBuilder> list = ...;
        StringBuilder sb = ...;
        Set<StringBuilder> set = new HashSet<>(list);
        set.add(sb);
        System.out.println(set.contains(sb)); // prints "true"
        sb.append("oops");
        System.out.println(set.contains(sb)); // prints "false"
    }
}
```

Then its time to discuss a few things with YOU

Follow Sunit Ghosh For More...

I am of course a functional interface

I model a method used to compare objects in the context, is the subject greater than this other one?

```
public interface Comparator<T> {
    int compare(T t1, T t2);
}

if a > b then compare(a, b) > 0
if a < b then compare(a, b) < 0
if a = b then compare(a, b) = 0
```

Also what could it mean for an object to be greater than another?

Follow Sunit Ghosh For More...

Taking String as example…

Implements a specific interface “Comparable”

```
public class String
implements Comparable<String> { }
```

```
var strings = List.of(
    "one", "two", "three", "four", "five");

var sortedStrings =
    strings.stream()
           .sorted()
           .toList();
```

Just sort this list by calling sorted() on a stream built on this list

Follow Sunit Ghosh For More...

```
var strings = Arrays.asList(
    "one", "two", "three", "four", "five");

var sortedStrings =
    strings.sorted(null);
```

or directly called the sorted method from the list interface and pass null to it

I am a functional interface

```java
public interface Comparator<T> {
    int compare(T t1, T t2);
}

Comparator<String> cmp =
 (s1, s2) -> s1.length() - s2.length();
```

Therefore I can work with lambdas as well!!

Avoid bugs by using JDK APIs

```
var ints =
    IntStream.range(0, 32)
        .mapToObj(index -> rand.nextInt(1000, 1100))
        .collect(Collectors.toList());

var sorted = ints.stream()
        .sorted(Integer::compare)
        .toList();
```

Wrapper classes have compare methods readymade for you

Follow Sunit Ghosh For More...

Adding more elements to a comparator is just about chaining calls

```
var cmp =
    Comparator
        .comparing(Person::lastName)
        .thenComparing(Person::firstName)
        .thenComparing(Person::age)
        .reversed();
```

Which preserves the readability of your code

sort them in the reverse order as well

you can create a comparator that will actually use the fact that your objects are comparable with the natural order method

```
Comparator<String> cmp =
    Comparator
        .naturalOrder()
        .reversed();
```

Reverse it if you like

if you have to deal with null values in the list you need to sort

```
var nullsAtTheEnd =
    Comparator.nullsLast(cmp);
var nullsAtTheBegining =
    Comparator.nullsFirst(cmp);
```

First or last?

A Recap

Use factory methods from Comparator to create your comparators

User factory methods from the wrapper classes to compare numbers

Use immutable objects

Follow

Sunit Ghosh

to get #interesting and latest #titbits on #java, #AiMl, #cloud technologies