Java coding challenge

- It is **REQUIRED** to have Unit Tests for the solution.
- Implement your solution **as Clean Code** as possible.

1. Create new project
2. Implement only one class `Range` to present a range of elements (having natural order). To create a `Range` instance, simply give it a `lowerbound` and a `upperbound`.

#1 Numbers
Class `Range` can be used with `int`

- `Range` must be *immutable*, once created, there is no way to change its `lowerbound` and `upperbound`.
- `Range` must provide a *static factory method* namely `of(int, int)` to create a new instance.
- It is not allowed to create a `Range` with `lowerbound > upperbound`.
- The method `contains(x)` must returns `true` only if `lowerbound <= x <=upperbound`.

Example:

```
Range validAgesForHighSchool = Range.of(16, 18);


validAgesForHighSchool(5); // false
validAgesForHighSchool(17); // true
```

#2 Type of Range
Mathmatically, a `Range` can be `open`, `closed`, `openClosed` or `closedOpen`.

- (5, 7) //open range excludes both bounds
- [5, 7] // closed range includes both bounds
- (5, 7] //open closed excludes lowerbound but includes upperbound
- [5, 7) //closed open includes lowerbound but excludes upperbound

Example:

```
Range open = Range.open(5, 7);
open.contains(5); //false

Range closed = Range.closed(5, 7);
closed.contains(5); // true

Range openClosed = Range.openClosed(5, 7);
openClosed.contains(5); // false
openClosed.contains(7); // true

Range closedOpen = Range.closedOpen(5, 7);
closedOpen.contains(5); // true;
closedOpen.contains(7); // false;
```

#3 **Make it generic with all `Comparable<T>` types**
Extends the `Range` such that it can supports any types implementing `Comparable` interface.

Example:

```
Range text = Range.open("abc", "xyz");

Range decimals = Range.open(BigDecimal.valueOf("1.123"), BigDe
cimal.valueOf("1.23456789"));

Range dates = Range.closed(LocalDate.of(2022, Month.SEPTEMBER,
01), LocalDate.of(2022, Month.SEPTEMBER, 30)));
```