1. Stack Peek Method in Java?

   The java.util.Stack.peek() method in Java is used to retrieve or fetch the first element of the Stack or the element present at the top of the Stack. The element retrieved does not get deleted or removed from the Stack.

   Ex:

```
Stack<String> STACK = new Stack<String>();

STACK.push("Welcome");

System.out.println("The element at the top of the"

    + " stack is: " + STACK.peek())
```

2. Distinct Method in java

   The Stream API provides the *distinct()* method that returns different elements of a list based on the *equals()* method of the *Object* class.
   distinct() uses hashCode() and equals() methods to get distinct elements.

```
List<String> list = Arrays.asList("AA", "BB", "CC", "BB", "CC", "AA", "AA");
long l = list.stream().distinct().count();
System.out.println("No. of distinct elements:"+l);
String output = list.stream().distinct().collect(Collectors.joining(","));
System.out.println(output);
```

3. Filter Method

The filter method, as its name suggests, **filters elements based upon a condition** you gave it. For example, if your list contains numbers and you only want numbers, you can use the filter method to only select a number that is fully divisible by two.

The map(Function mapper) method takes a Function, technically speaking, an object of java.util.function.Function interface. This function is then applied to each element of Stream to convert it into the type you want.

Because we need to convert a String to an Integer, we can pass either the Integer.parseInt() or Integer.valueOf() method to the map() function.

```
List<String> numbers = Arrays.asList("1", "2", "3", "4", "5", "6");
System.out.println("original list: " + numbers);
List<Integer> even = numbers.stream()
                    .map(s -> Integer.valueOf(s))
                    .filter(number -> number % 2 == 0)
                    .collect(Collectors.toList());//which will accumulate all even numbers into a List and return.
```

4. Parallel Stream

```
    private long countPrimes(int max) {
   return range(1, max).parallel().filter(this::isPrime).count();
                                        }
```

we have the method **countPrimes** that counts the number of prime numbers between 1 and our max. A stream of numbers is created by a range method. The stream is then switched to parallel mode; numbers that are not primes are filtered out and the remaining numbers are counted.
parallelization is just a matter of calling the parallel() method. When we do that, the stream is split into multiple chunks, with each chunk processed independently and with the result summarized at the end

5. Difference b/w Parallel Stream and Stream?

```
      for (int i = 0; i < 1000; i++) {
    list.add(i);
  }
  list.stream().forEach(System.out::println);
 }
}
```

You will notice that this program will output the numbers from 0 to 999 sequentially, in the order in which they are in the list. If we change stream() to parallelStream() this is not the case anymore (at least on my computer): all number are written, but in a different order. So, apparently, parallelStream() indeed uses multiple threads.

6. MIN and MAX Methods

We use the **min() and max() methods** to find the min & max value in streams. These methods are used for finding min & max values in different types of streams such as stream of chars, strings, dates.\

```
List<LocalDate> dates = Arrays.asList(LocalDate.now(),
                                      LocalDate.now().minusDays(9),
                                      LocalDate.now().minusMonths(2),
                                      LocalDate.now().minusDays(30));

      //getting max date
      LocalDate maxdate = dates.stream()
     .max( Comparator.comparing( LocalDate::toEpochDay ) )
     .get();

      //getting min date
      LocalDate mindate = dates.stream()
     .min( Comparator.comparing( LocalDate::toEpochDay ) )
     .get();
```

7. Difference between ADD and SET methods in arraylist?

set replaces the element at the given index. add inserts the element at the givenindex and moves all elements ahead of it one position. set replaces the element at the given index. add inserts the element at the given index and moves all elements ahead of it one position

EX :-

```
ArrayList<Integer>    arrlist = new ArrayList<Integer>();

    int i = arrlist.set(3, 30);

    Before operation : [1, 2, 3, 4, 5]
    After operation : [1, 2, 3, 30, 5]
    Replaced element : 4

    int i = arrlist.set(3, 30);

    Before operation : [1, 2, 3, 4, 5]
   After operation : [1, 2, 3, 30,4, 5]
```

8. Difference between comparator and comparable?

   Comparable provides a single sorting sequence. In other words, we can sort the collection on the basis of a single element such as id, name, and price. Comparable affects the original class**.** Comparable provides compareTo() method to sort elements.

   The Comparator provides multiple sorting sequences. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc. omparator doesn't affect the original class**.** Comparator provides compare() method to sort elements.