Stream is a interface present in java.util.stream.Stream
Returns the count of elements in this stream.

```
import java.util.stream.Stream;

public class Eg1 {

public static void main(String[] args) {

Stream<Integer> s = Stream.of(1, 2, 3, 4, 5, 6);
System.out.println(s.count()); // 6

}
}
```

Stream<T> filter(Predicate<? super T> predicate);

public static<T> Stream<T> of(T... values)
Returns a sequential ordered stream whose elements are the specified values.

public boolean isPresent() present in Optional.class
A container object which may or may not contain a non-null value.
If a value is present,**isPresent() will return true** and **get() will return the value.**
Return true if there is a value present, otherwise false.

public boolean startsWith(String prefix){}
Tests if this string starts with the specified prefix

Optional<T> findFirst();
Returns an Optional describing the first element of this stream, or an empty Optional if the stream is empty.

public T get(){}
If a value is present, returns otherwise

```
import java.util.Optional;
import java.util.stream.Stream;

public class Eg2 {

public static void main(String[] args) {

Stream<String> name = Stream.of("Mani", "Hari", "Mahesh", "Vikas");
Optional<String> findFirst = name.filter(i -> i.startsWith("Z")).findFirst();

if (findFirst.isPresent()) {
System.out.println(findFirst.get());
}else {
System.out.println("No Name Checked");
}

}
}
```

```java
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class Eg3 {

public static void main(String[] args) {
List<String> l1 = new ArrayList<>();
l1.add("Pradeep");
l1.add("Praneeth");
l1.add("Mahesh");
l1.add("Vikas");
System.out.println(l1); // [Pradeep, Praneeth, Mahesh, Vikas]

List<String> l2 = l1.stream().map(s -> s.toUpperCase()).collect(Collectors.toList());
System.out.println(l2); // [PRADEEP, PRANEETH, MAHESH, VIKAS]

}
}
```

```
import java.util.ArrayList;
import java.util.Optional;
import java.util.stream.Stream;

public class Eg4 {
public static void main(String[] args) {

ArrayList<Integer> al = new ArrayList<>();
al.add(1);
al.add(3);
al.add(2);
al.add(5);
al.add(4);
System.out.println("Orginal List: " + al); // Orginal List: [1, 3, 2, 5, 4]
Stream<Integer> myStream = al.stream();

Optional<Integer> minVal = myStream.min(Integer::compare);
if (minVal.isPresent()) {
System.out.println("Minimum Value: " + minVal.get()); // Minimum Value: 1
}

myStream = al.stream();
Optional<Integer> maxVal = myStream.max(Integer::compare);
if (maxVal.isPresent()) {
System.out.println("Minimum Value: " + maxVal.get());//Minimum Value: 5
}
}
}
```

sorted() returns a stream consisting of the elements of this stream, sorted according to natural order

```
import java.util.ArrayList;
import java.util.stream.Stream;

public class Eg5 {
public static void main(String[] args) {

ArrayList<Integer> al = new ArrayList<>();
al.add(1);
al.add(3);
al.add(2);
al.add(5);
al.add(4);
Stream<Integer> sortedStream = al.stream().sorted();
System.out.println("Natural Sorting"); // Natural Sorting
sortedStream.forEach((n) -> System.out.print(n)); // 12345
}
}
```