STREAM API

```
import java.util.*;

import java.util.stream.Collectors;

class Main {

    public static void main(String[] args) {

   // Steam API -find even number

  List<Integer> number=Arrays.asList(2,7,9,4,6,1,8,3,5);

  System.out.println(number.stream()

                        .filter(n->n%2==0)

                        .collect(Collectors.toList()));

}

}

OUTPUT:[2, 4, 6, 8]
```

NOTE: ***.collect(Collectors.toList()) is*** the terminal operation that takes the elements flowing through the stream and accumulates them into a List. In your example it turns the stream of even numbers into a List<Integer> (which println then prints).

```
import java.util.*;

import java.util.stream.Collectors;

class Main {

    public static void main(String[] args) {

  // Steam API -to find the maximum umber among the given array

  List<Integer> number=Arrays.asList(2,7,9,4,6,1,8,3,5);

  System.out.println(number.stream().max(Integer ::compare));

}

}

Ouput: Optional[9]
```

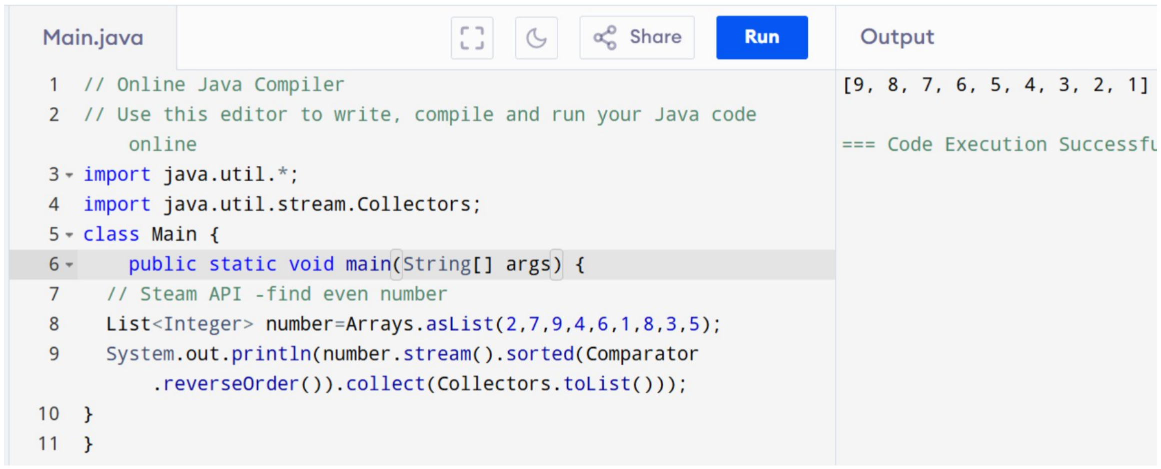
NOTE: *.max(Integer::compare)* → terminal operation that performs a reduction:

It uses the provided comparator (Integer::compare) to compare two elements.

It walks the stream and keeps the element that is "greater" according to the comparator.

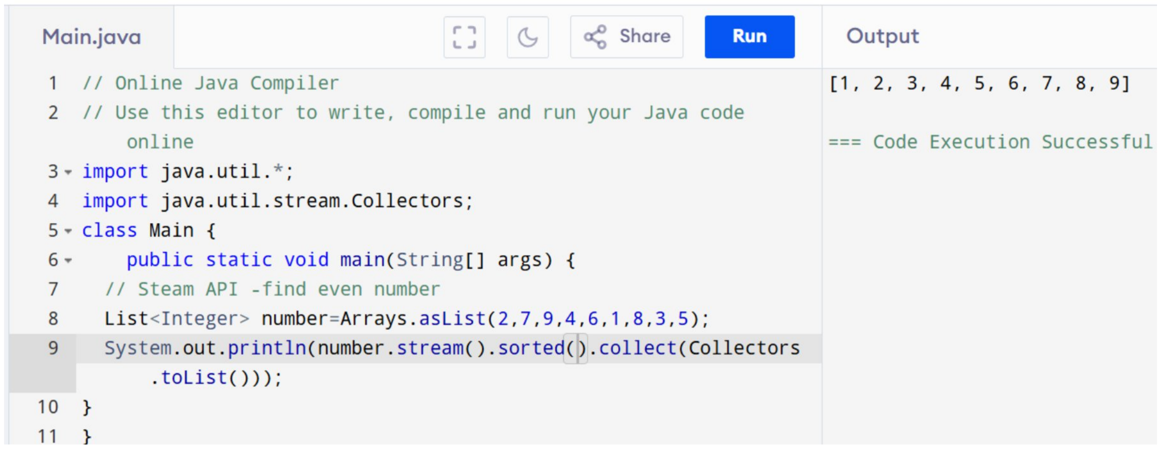
After processing all elements it returns the result wrapped in an Optional because a stream might be empty.

System.out.println(max) prints the Optional (e.g. Optional[9]).



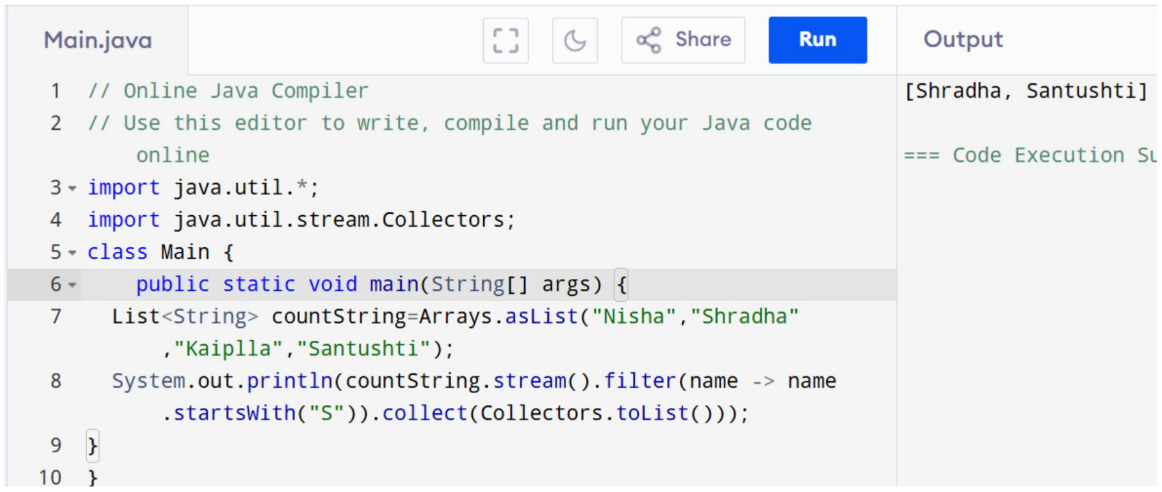
```java
// Online Java Compiler
// Use this editor to write, compile and run your Java code
    online
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
  // Steam API -find even number
  List<Integer> number=Arrays.asList(2,7,9,4,6,1,8,3,5);
  System.out.println(number.stream().sorted(Comparator
      .reverseOrder()).collect(Collectors.toList()));
}
}
```

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]

=== Code Execution Successfu
```



```java
// Online Java Compiler
// Use this editor to write, compile and run your Java code
    online
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
  // Steam API -find even number
  List<Integer> number=Arrays.asList(2,7,9,4,6,1,8,3,5);
  System.out.println(number.stream().sorted().collect(Collectors
      .toList()));
}
}
```

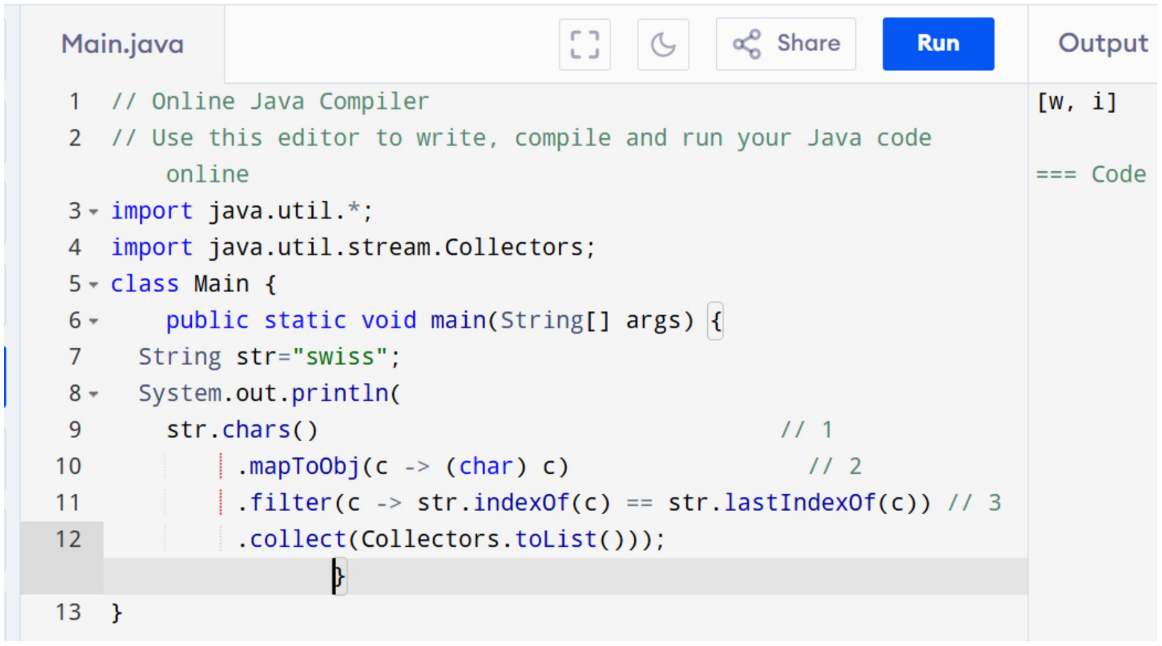
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]

=== Code Execution Successful
```


Main.java

```java
// Online Java Compiler
// Use this editor to write, compile and run your Java code
    online
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
  List<String> countString=Arrays.asList("Nisha","Shradha"
      ,"Kaiplla","Santushti");
  System.out.println(countString.stream().filter(name -> name
      .startsWith("S")).collect(Collectors.toList()));
}
}
```

Output

```
[Shradha, Santushti]

=== Code Execution Su
```


Main.java

```java
// Online Java Compiler
// Use this editor to write, compile and run your Java code
    online
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
  String str="swiss";
  System.out.println(
    str.chars()                                    // 1
        .mapToObj(c -> (char) c)                   // 2
        .filter(c -> str.indexOf(c) == str.lastIndexOf(c)) // 3
        .collect(Collectors.toList()));
                }
}
```

Output

```
[w, i]

=== Code
```

3. `.filter(c -> input.indexOf(c) == input.lastIndexOf(c))`

- This filter keeps only characters that appear **exactly once** in the string.
  - `indexOf(c)` → gives the position of the **first occurrence**.
  - `lastIndexOf(c)` → gives the position of the **last occurrence**.
  - If they are equal → the character occurs only once.

  For `"swiss"`:

  - `'s'`: first index = 0, last index = 4 → not equal → repeated.
  - `'w'`: first index = 1, last index = 1 → equal → unique.
  - `'i'`: first index = 2, last index = 2 → equal → unique.
  - `'s'`: repeated → ignored.
  - `'s'`: repeated → ignored.

  After filtering, the stream is:

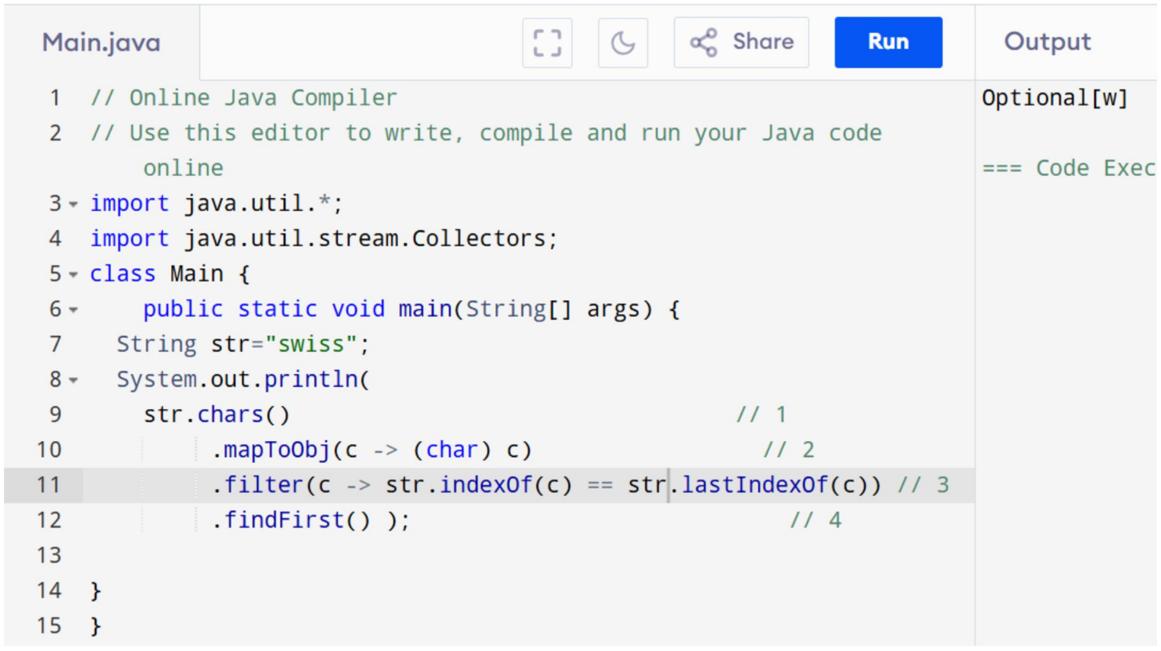
  ```css
  ['w', 'i']
  ```

`.mapToObj(c -> (char) c)`

- Converts each `int` code point into a `Character` object (autoboxed).

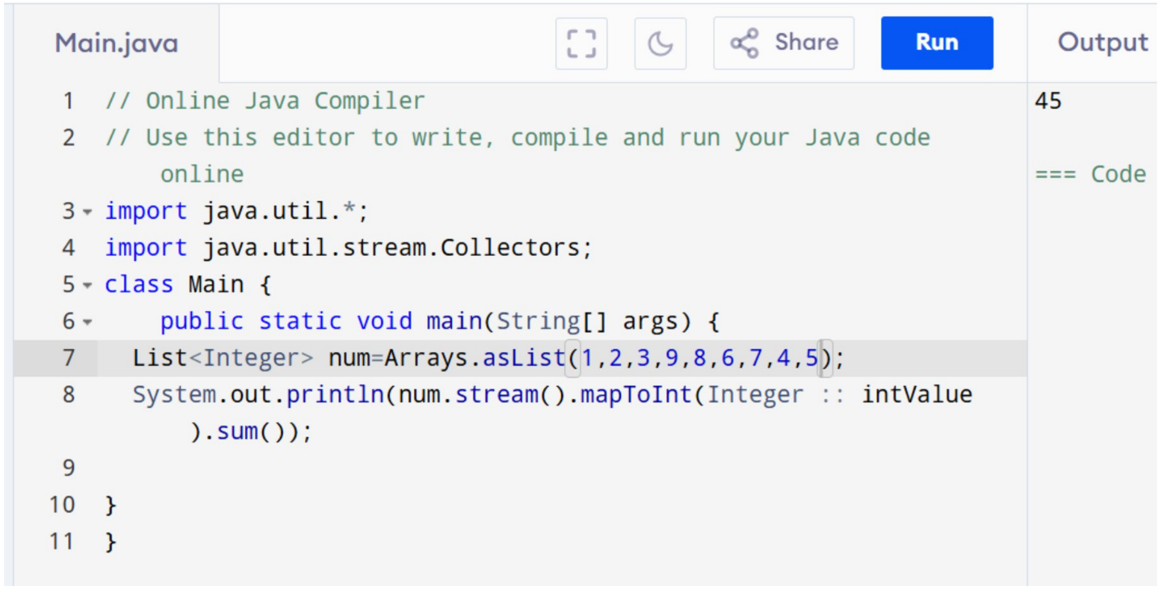
  So the stream becomes:

  ```css
  ['s', 'w', 'i', 's', 's']
  ```



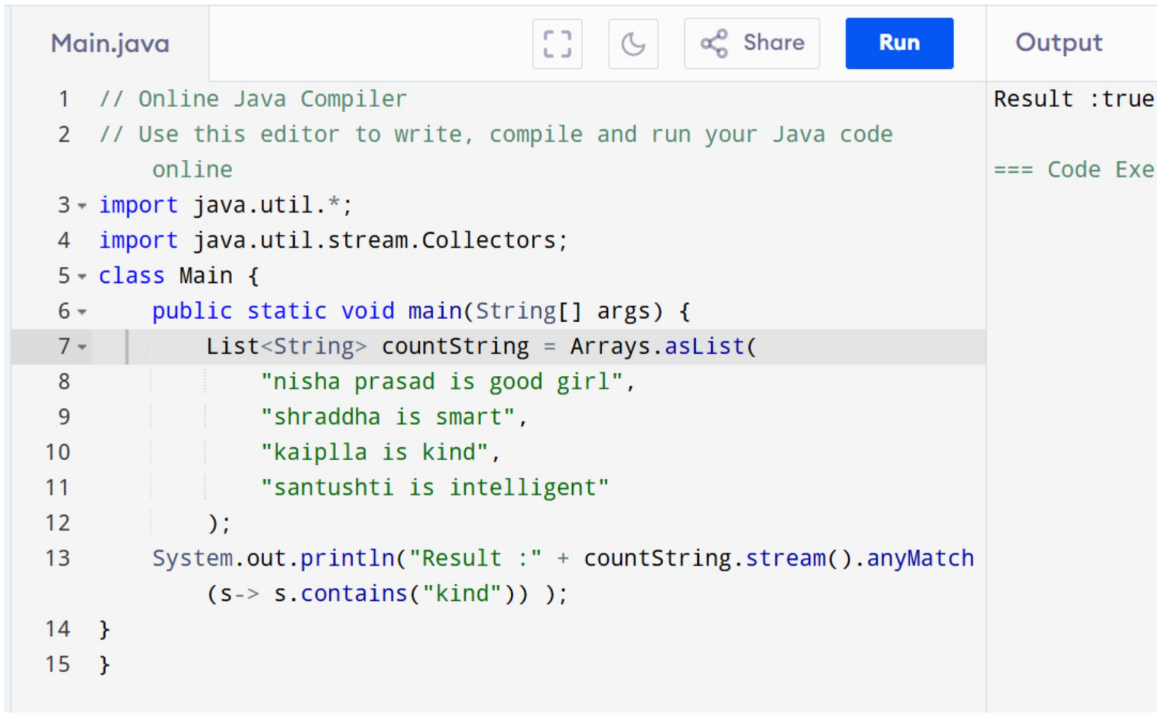
```java
// Online Java Compiler
// Use this editor to write, compile and run your Java code
    online
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
  String str="swiss";
  System.out.println(
    str.chars()                                    // 1
        .mapToObj(c -> (char) c)                     // 2
        .filter(c -> str.indexOf(c) == str.lastIndexOf(c)) // 3
        .findFirst() );                                // 4

}
}
```

```
Optional[w]

=== Code Exec
```



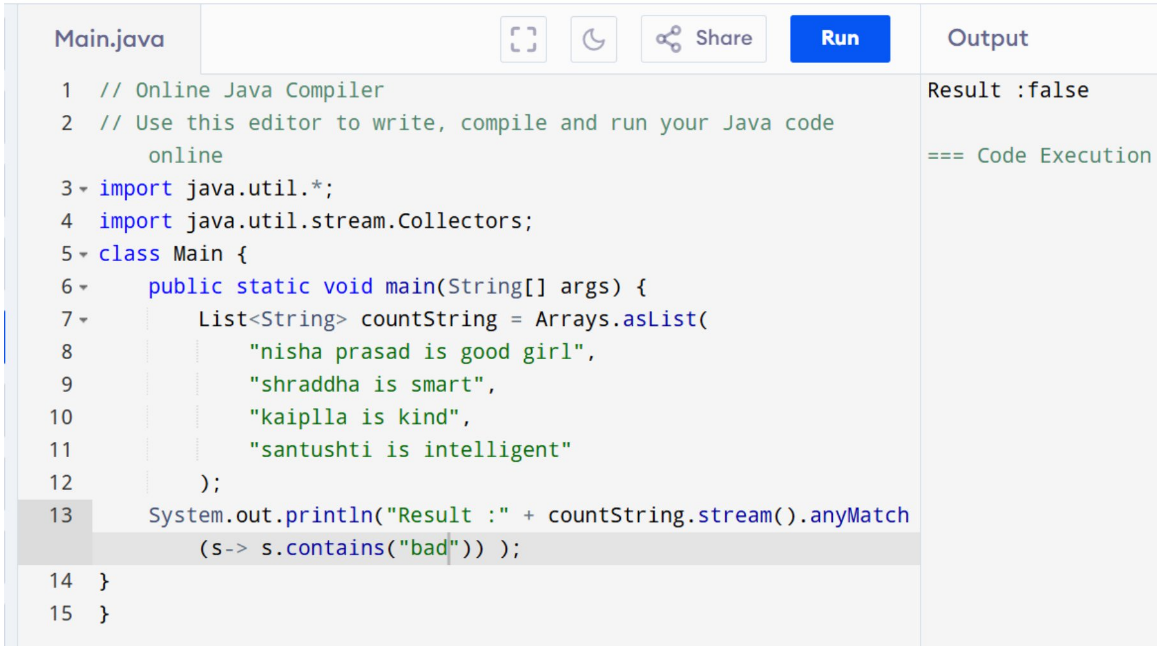
```java
// Online Java Compiler
// Use this editor to write, compile and run your Java code
    online
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
  List<Integer> num=Arrays.asList(1,2,3,9,8,6,7,4,5);
  System.out.println(num.stream().mapToInt(Integer :: intValue
      ).sum());

}
}
```

```
45

=== Code
```



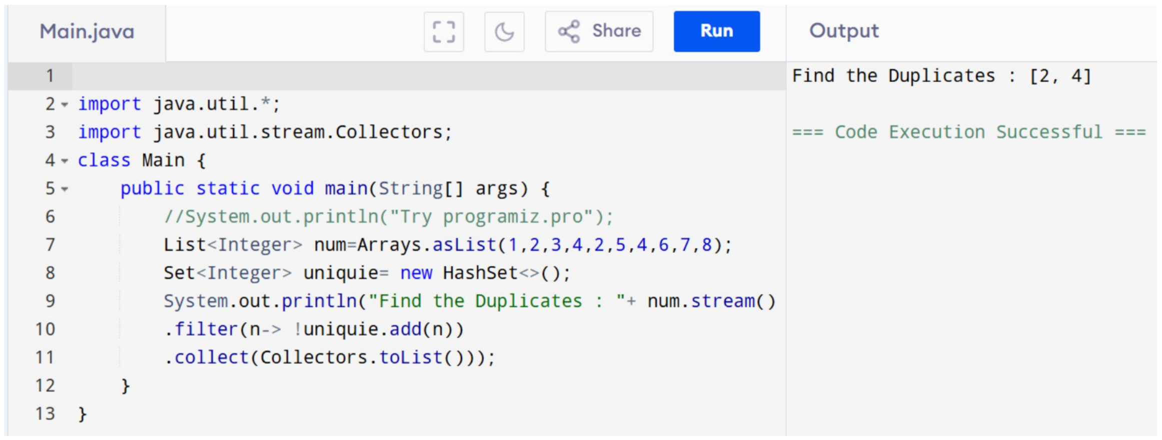
```java
// Online Java Compiler
// Use this editor to write, compile and run your Java code
    online
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
        List<String> countString = Arrays.asList(
            "nisha prasad is good girl",
            "shraddha is smart",
            "kaiplla is kind",
            "santushti is intelligent"
        );
    System.out.println("Result :" + countString.stream().anyMatch
        (s-> s.contains("kind")) );
}
}
```

```
Result :true

=== Code Exe
```



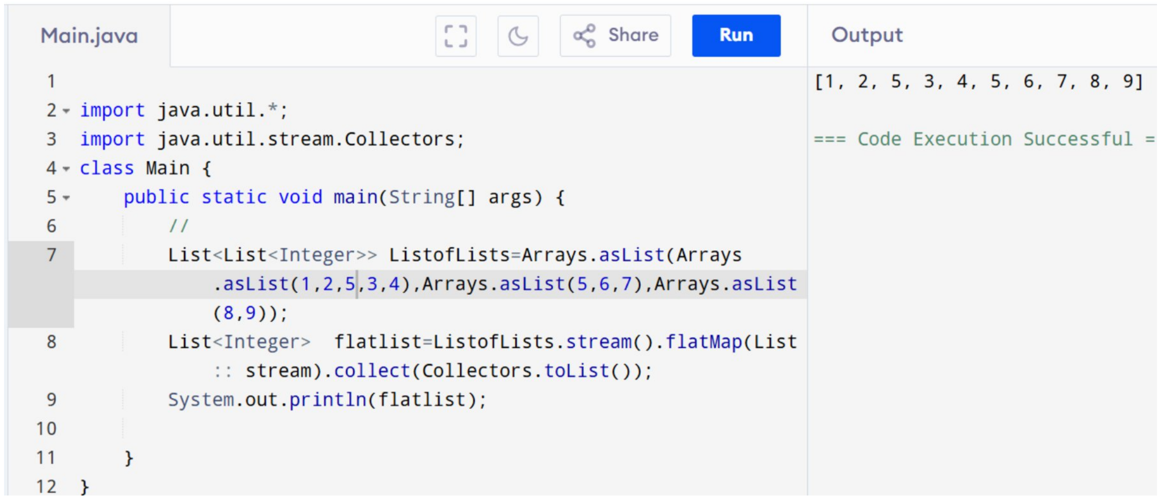
```java
// Online Java Compiler
// Use this editor to write, compile and run your Java code
    online
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
        List<String> countString = Arrays.asList(
            "nisha prasad is good girl",
            "shraddha is smart",
            "kaiplla is kind",
            "santushti is intelligent"
        );
    System.out.println("Result :" + countString.stream().anyMatch
        (s-> s.contains("bad")) );
}
}
```

```
Result :false

=== Code Execution
```



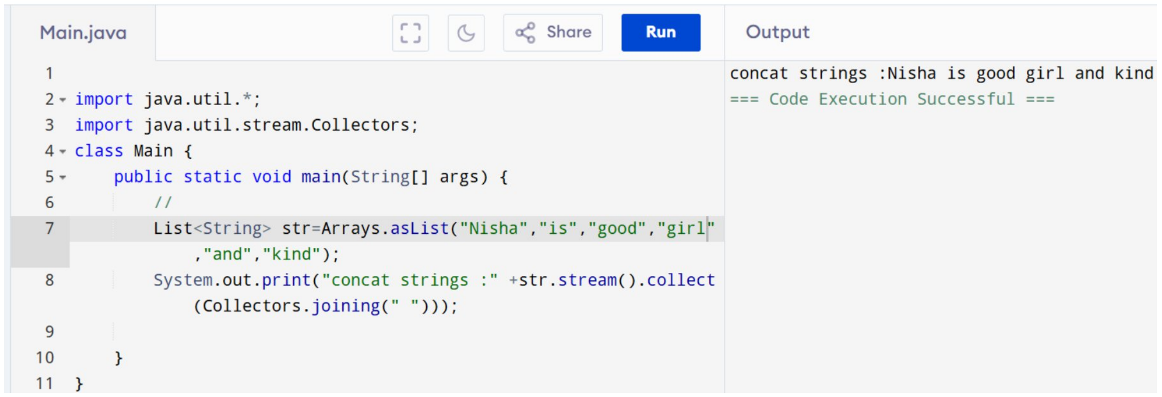
```java

import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
        //System.out.println("Try programiz.pro");
        List<Integer> num=Arrays.asList(1,2,3,4,2,5,4,6,7,8);
        Set<Integer> uniquie= new HashSet<>();
        System.out.println("Find the Duplicates : "+ num.stream()
        .filter(n-> !uniquie.add(n))
        .collect(Collectors.toList()));
    }
}
```

```
Find the Duplicates : [2, 4]

=== Code Execution Successful ===
```



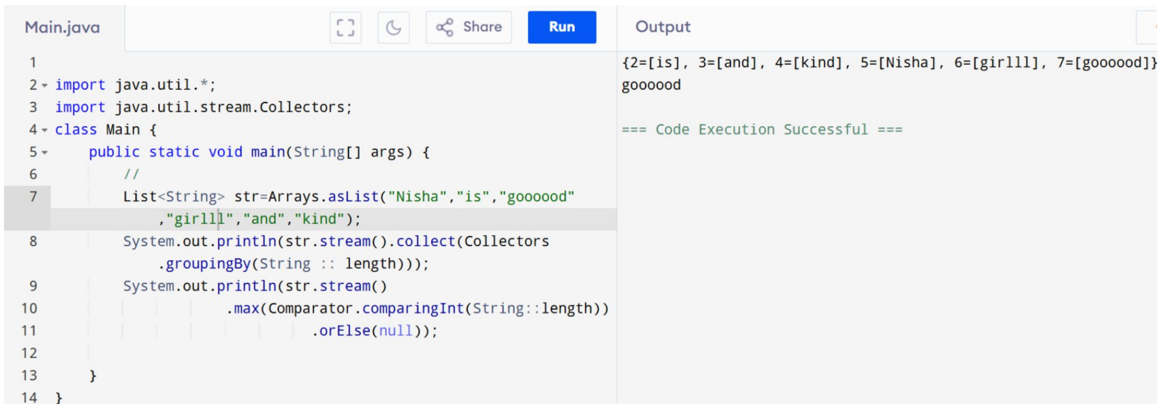
```java

import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
        //
        List<List<Integer>> ListofLists=Arrays.asList(Arrays
            .asList(1,2,5,3,4),Arrays.asList(5,6,7),Arrays.asList
            (8,9));
        List<Integer>  flatlist=ListofLists.stream().flatMap(List
            :: stream).collect(Collectors.toList());
        System.out.println(flatlist);

    }
}
```

```
[1, 2, 5, 3, 4, 5, 6, 7, 8, 9]

=== Code Execution Successful =
```



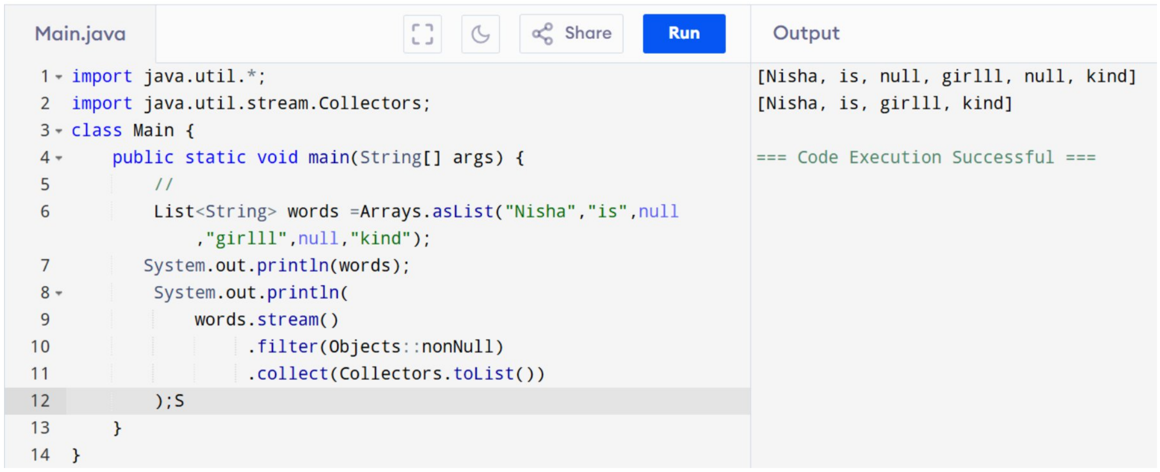
```java

import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
        //
        List<String> str=Arrays.asList("Nisha","is","good","girl"
            ,"and","kind");
        System.out.print("concat strings :" +str.stream().collect
            (Collectors.joining(" ")));

    }
}
```

```
concat strings :Nisha is good girl and kind
=== Code Execution Successful ===
```



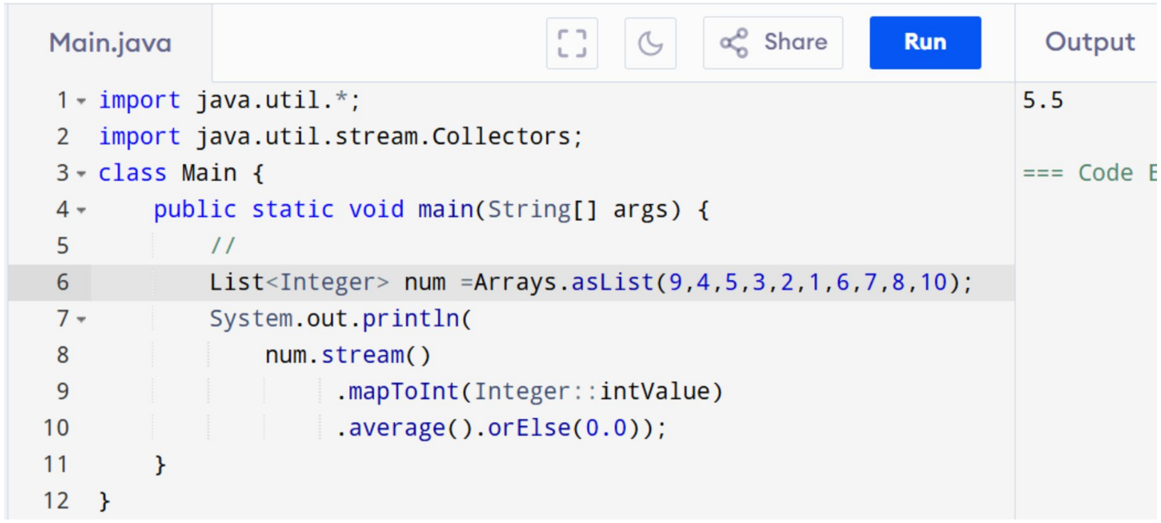
```java

import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
        //
        List<String> str=Arrays.asList("Nisha","is","goooood"
            ,"girlll","and","kind");
        System.out.println(str.stream().collect(Collectors
            .groupingBy(String :: length)));
        System.out.println(str.stream()
                    .max(Comparator.comparingInt(String::length))
                            .orElse(null));

    }
}
```

```
{2=[is], 3=[and], 4=[kind], 5=[Nisha], 6=[girlll], 7=[goooood]}
goooood

=== Code Execution Successful ===
```



```
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
        //
        List<String> words =Arrays.asList("Nisha","is",null
            ,"girlll",null,"kind");
       System.out.println(words);
        System.out.println(
            words.stream()
                .filter(Objects::nonNull)
                .collect(Collectors.toList())
        );S
    }
}
```

```
[Nisha, is, null, girlll, null, kind]
[Nisha, is, girlll, kind]

=== Code Execution Successful ===
```



```
import java.util.*;
import java.util.stream.Collectors;
class Main {
    public static void main(String[] args) {
        //
        List<Integer> num =Arrays.asList(9,4,5,3,2,1,6,7,8,10);
        System.out.println(
            num.stream()
                .mapToInt(Integer::intValue)
                .average().orElse(0.0));
    }
}
```

```
5.5

=== Code E
```

```
import java.util.*;

import java.util.stream.Collectors;

class Main {

    public static void main(String[] args) {

      int[] arr={1,9,8,6,5,4,3,7,2};

      List<Integer> array=Arrays.stream(arr)

                          .boxed()

                          .collect(Collectors.toList());

        int maximum = array.stream()

                      .max(Integer ::compareTo)

                      .get();

       int secondmax =array.stream()

                      .distinct()
```

```
                    .sorted(Comparator.reverseOrder())
                    .skip(1)
                    .findFirst()
                    .get();
        int minimum = array.stream()
                    .min(Integer ::compareTo)
                    .get();
        int secondmin=array.stream()
                    .distinct()
                    .sorted()
                    .skip(1)
                    .findFirst()
                    .get();
        int pro1= maximum*secondmax;
        int pro2= minimum*secondmin;
        int difference=pro1-pro2;
        System.out.println(difference);

    }
}
OUTPUT: 70
```