```
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;
import java.util.function.Function;
import java.util.stream.Collectors;

public class LambdasStreams {

    public static void main(String... args) {

        List<String> cities = Arrays.asList("Regensburg", "Basel", "Munich", "Bonn", "Hamburg", "Munich", "Berlin");

        // print distinct list of cities on console
        cities.stream()
                .distinct()
                .forEach(System.out::println);

        // print first 3 cities in list
        System.out.println("--");
        cities.stream()
                .limit(3)
                .forEach(System.out::println);

        // store in boolean variable whether all city names have all at least 6 characters
        boolean nameLengthAtLeast6Chars = cities.stream().allMatch( city -> city.length() > 5);
        System.out.println("--");
        System.out.println("All names have length of at least 6 chars: " + nameLengthAtLeast6Chars);

        // store list of distinct city names in descending order of name's length
        List<String> distinctSorted = cities.stream()
                .distinct()
                .sorted( (s1,s2) -> -Integer.compare(s1.length(), s2.length()))
                .collect(Collectors.toList());
        System.out.println("--");
        distinctSorted.forEach(System.out::println);
```

```
// store list of city names in CAPITAL LETTERS in new TreeSet
Set<String> inCapitalLetters = cities.stream()
        .sorted()
        .map( s -> s.toUpperCase() )
        .collect(Collectors.toCollection(TreeSet::new));
System.out.println("--");
inCapitalLetters.forEach(System.out::println);

// find first city name in natural order of given length len and if present and store name in String
// variable or store String "no city name of length ..."
// (use terminal operation that returns Optional<T> object and continue using this object)
System.out.println("--");
int len = 11;
String firstOfGivenLength = cities.stream()
        .sorted()
        .filter( s -> s.length() == len )
        .findFirst()
        .orElse("no city name of length " + len);
System.out.println(firstOfGivenLength);

// print name of one city with longest name
System.out.println("--");
cities.stream()
        .sorted( (s1,s2) -> -Integer.compare(s1.length(), s2.length()))
        .findFirst()
        .ifPresent(System.out::println);

// store length of longest city name
System.out.println("--");
int lengthOfLongesName = cities.stream()
        .map( s -> s.length() )
        .reduce(0, Integer::max);
System.out.println("length of longest name: " + lengthOfLongesName);


// reduce list of names to String of their initals
System.out.println("--");
String initials = cities.stream()
        .map( s -> s.charAt(0) )
        .reduce("", (c1,c2) -> c1+c2, (c1,c2) -> c1+c2);
System.out.println("Initials: " + initials);
```

```
        // compute total string length over all names
        System.out.println("--");
        int totalLength = cities.stream()
                .mapToInt(String::length)
                .sum();
        System.out.println("Total string length over all name: " + totalLength);

        // Store Map<Character,Long> of number of cities grouped by their initials
        Map<Character,Long> frequencyOfInitials = cities.stream()
                .map( s -> s.charAt(0) )
                .collect(Collectors.groupingBy(Function.identity(), Collectors.counting()));

        // as above but do not store but print directly to console
        System.out.println("--");
        cities.stream()
                .map( s -> s.charAt(0) )
                .collect(Collectors.groupingBy(Function.identity(), Collectors.counting()))
                .entrySet()
                .forEach( entrySet -> System.out.println(entrySet.getKey() + ": " + entrySet.getValue()));


        // as above but print map sorted by value
        System.out.println("--");
        cities.stream()
                .map( s -> s.charAt(0) )
                .collect(Collectors.groupingBy(Function.identity(), Collectors.counting()))
                .entrySet()
                .stream()
                .sorted(Comparator.comparingLong(Map.Entry::getValue))
                .forEach( entrySet -> System.out.println(entrySet.getKey() + ": " + entrySet.getValue()));

        // count number of letters in city names and print table to console sorted by key
        System.out.println("--");
        cities.stream()
                .map( s -> s.split("") )
                .flatMap(Arrays::stream)
                .sorted()
                .collect(Collectors.groupingBy(Function.identity(), Collectors.counting()))
                .entrySet()
                .forEach( entrySet -> System.out.println(entrySet.getKey() + ": " + entrySet.getValue()));

    }
}
```