```java
package Lambda_Functional_Programming;

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

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

        /*
            1) Lambda (Functional Programming) Java 8 ile kullanılmaya başlanmıştır.
               Ana amacı Fonksiyonel programlamayı kolaylaştırarak kod yazmayı daha sade, basit ve esnek hale getirmektir.
            2) Functional Programming'de "Ne yapılacak" (What to do) üzerine yoğunlaşılır.
            3) Functional Programming, Arrays ve Collections ile kullanılır.
            4) Functional Programming kapsamında "Lambda Expession" kullanılabilir ama önerilmez.
             "Lambda Expession" yerine "Method Reference" tercih edilir.
            5) "Method Reference" kullanımı "Class Name :: Method Name"


        * stream() methodu collection'dan elementleri akışa dahil etmek için ve methodlara ulaşmak için kullanılır.

        * map() : yapacağımız bir işlem tüm elemanlara uygulayacaksak kullanıyoruz.
        yani elemanların değerleri değişecekse map() methodu kullanılır.
        Örneğin Her elemanı ikiyle çarpacaksak map(t-> t*2)


        * collect(Collectors.toList) : Her bir elemanı alıp listin içine atmak için kullanılır. Elemanları collection'a çevirir.

        * filter() : Herhangi bir şart olduğunda, istenen koşula göre filtreleme yapmak için kullanılır.
            * filter(t->t>5)

        * distinct() : tekrar eden elemanları 1 kere almak için kullanılır

        * reduce() : tüm elemanları istenen şarta uygun işlemler sonucunda tek elemana indirgemek için kullanılır.
            * reduce(0,(t,u)->t+u)
            * reduce((t,u)->t+u).get()

        * forEach() : Herbir elemana birşeyler yapmak için kullanılır. Örneğin yazdırmak.

         */

        List<Integer> list = new ArrayList<>();

        list.add(8);
        list.add(9);
        list.add(131);
        list.add(10);
        list.add(9);
        list.add(10);
```

```
        list.add(2);
        list.add(8);

        List<Integer> yeniListe = new ArrayList<>();
      // list.stream().filter(t-> t<10).forEach(t -> System.out.print(t+"
"));


        // todo listeye atama yap
        yeniListe=list.stream().filter(t->
t<11).collect(Collectors.toList());

        // todo şart
        list.stream().filter(t->t%2==0).forEach(t-> System.out.print(t+"
"));
        System.out.println();

        // todo karelerini al
        list.stream().filter(t->t%2!=0).map(t->t*t).forEach(t->
System.out.print(t+" "));
        System.out.println();

        // todo max
        Integer max = list.stream().distinct().reduce(Integer.MIN_VALUE,
(t,u)-> t>u ? t : u );

        // todo sıralama
        Integer min = list.stream().distinct().filter(t->t%2==0).filter(t-
>t>7).sorted(Comparator.reverseOrder()).reduce(Integer.MAX_VALUE,(t,u)->u);
//ters sıralama

list.stream().sorted(Comparator.comparing(Integer::intValue)).forEach(Utils
::ayniSatirdaBosluklaYazdir);//şu şarta göre sıralama

        // todo Matth classını kullan
        Integer toplam =
list.stream().distinct().filter(Utils::ciftElemaniSec).map(Utils::karesinAl
).reduce(Math::addExact).get();

        // Math::addExact    toplama
        // Math::multiplyExact    çarpma
        // Math::max    Maximum

        // todo String metodlarını kullan
        //map(String::toUpperCase)
        // list.replaceAll(String::toUpperCase)

        // todo belli şarta göre sil
        // list.removeIf(t->t.length()>5);
        // list.removeIf(t->t.subString(2,5));

        // todo belli şartlara uyup uymadığını kontrol
        // list.stream().allMatch(t->t.length()<12);
        // list.stream().noneMatch(t->t.startsWith("X") ||
t.startsWith("x"));
        // list.stream().anyMatch(t->t.endsWith("r"));

        // todo tamsayılarda belli aralıklarda işlem yap
        // IntStream.rangeClosed(7,100).reduce(Math::addExact).getAsInt();

        // todo herhangi bir değeri atlamak için
```

```
        //
list.stream().sorted(Comparator.comparing(Courses::getAverageScore)).skip(x
).collect(Collectors.toList());


//.....................................................................
..............................

        List<String> liste = new ArrayList<>();
        liste.add("Ali");
        liste.add("Ali");
        liste.add("Mark");
        liste.add("Amanda");
        liste.add("Christopher");
        liste.add("Jackson");
        liste.add("Mariano");
        liste.add("Alberto");
        liste.add("Tucker");
        liste.add("Benjamin");

        System.out.println();
        neIstersen(liste);

    }

    public static void neIstersen(List<String> liste){

        liste.stream().filter(t->t.startsWith("A")).forEach(t->
System.out.print (t+" "));
    }
}
```