If we want to process a group of objects from collection then we should use streams.

The stream is an interface present in java.util.Stream. Once we got the stream, by using that we can process the objects of collection. Every Collection implements the Stream interface.

Syntax:

Stream s=Collectionobject.stream();

The objects can be processed in 2 ways:

1. Configuration
2. Processing.

**1.Configuration**:- we can configure either by using filtering mechanism or by using map mechanism.

**1.1 Filtering:-** we filter the elements from collection based on some Boolean condition filter() method of stream interface.

Syntax:

Public Stream filter(Predicate <T> t);

**1.2 Mapping:-** If we want to create a separate new object, for every object present in the collection based on our requirement then we should go for map() method of stream interface.

Syntax:

Public Stream map(Function <T> f);

**2.processing:-**

Elements in collection are processed by

a.collect() method.

b.count() method.

c.sorted() method.

d.min() and max() methods.

e.forEach() method.

f. toArray() method.

**2.1. collect():-** This method collects the elements from the stream and adding to specified collection indicated by argument.

Example:-1

import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

import java.util.function.Predicate;

import java.util.stream.Collectors;

import java.util.stream.Stream;

class suku{

public static void main(String args[])

{

ArrayList<Integer> a1=new ArrayList();

a1.add(10);

a1.add(20);

a1.add(30);

a1.add(40);

a1.add(50);

System.out.println("Collection:"+a1);

Stream s=a1.stream();

Predicate<Integer> p=(i)->{

if(i>20){

return true;

}

else {

return false;

}};

List <Integer> a2=(List<Integer>) s.filter(p).collect(Collectors.toList());

System.out.println(a2);

}

}

Output:-

Collection:[10, 20, 30, 40, 50]

[30, 40, 50]

Example:2

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.function.Function;

**import** java.util.stream.Collectors;

**import** java.util.stream.Stream;

**class** suku{

**public** **static** **void** main(String args[])

{

ArrayList<Integer> a1=**new** ArrayList();

a1.add(10);

a1.add(20);

a1.add(30);

a1.add(40);

a1.add(50);

System.***out***.println("Collection:"+a1);

Stream s=a1.stream();

Function<Integer,Integer>p=(i)->{

**if**(i>20){

**return** i+10;

}

**else** {

**return** i+20;

}

};

List <Integer> a2=(List<Integer>) s.map(p).collect(Collectors.*toList*());

System.***out***.println(a2);

}

}

Output:

Collection:[10, 20, 30, 40, 50]

[30, 40, 40, 50, 60]

**2.2.count():-** This method returns the no.of elements in Stream.

Syntax:

Public integer count();

Example:

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.function.Function;

**import** java.util.stream.Collectors;

**import** java.util.stream.Stream;

**class** suku{

**public** **static** **void** main(String args[])

{

ArrayList<Integer> a1=**new** ArrayList();

a1.add(10);

a1.add(20);

a1.add(30);

a1.add(40);

a1.add(50);

System.***out***.println("Collection:"+a1);

Stream s=a1.stream();

Function<Integer,Integer>p=(i)->{

**if**(i>20){

**return** i+10;

}

**else** {

**return** i+20;

}

};

s=s.map(p);

System.***out***.println("No.of elements in Stream:"+s.count());

}

}

Output:-

Collection:[10, 20, 30, 40, 50]

No.of elements in Stream:5

**3.sorted:-** if we sort the elements present inside stream then we should go for sorted() method.

Public Stream sorted():Default sorting order

Public Stream sorted(Comparator c): customized sorting order.

Example:-

**import** java.util.ArrayList;

**import** java.util.Comparator;

**import** java.util.List;

**import** java.util.function.Function;

**import** java.util.stream.Collectors;

**import** java.util.stream.Stream;

**class** suku{

**public** **static** **void** main(String args[])

{

ArrayList<Integer> a1=**new** ArrayList();

a1.add(10);

a1.add(50);

a1.add(30);

a1.add(40);

a1.add(20);

System.***out***.println("Collection:"+a1);

Function<Integer,Integer>p=(i)->{

**if**(i>20){

**return** i;

}

**else** {

**return** i;

}

};

Comparator <Integer> c=(a,b)->{

**if**((Integer)a>(Integer)b) {

**return** -1;

}

**else**

{

**return** 1;

}

};

Stream <Integer> s=a1.stream();

s=s.map(p).sorted(c);

List<Integer> l=(List<Integer>)s.collect(Collectors.*toList*());

System.***out***.println(l);

}

}

Output:

---------

Collection:[10, 50, 30, 40, 20]

[50, 40, 30, 20, 10]

**4.min() and max():-**

Syntax:

Min(Comparator c)

Max(Comparator c);

These methods returns the max and min value according to comparator.

Example:-

**import** java.util.ArrayList;

**import** java.util.Comparator;

**import** java.util.List;

**import** java.util.function.Function;

**import** java.util.stream.Collectors;

**import** java.util.stream.Stream;

**class** suku{

**public** **static** **void** main(String args[])

{

ArrayList<Integer> a1=**new** ArrayList();

a1.add(10);

a1.add(50);

a1.add(30);

a1.add(40);

a1.add(20);

System.***out***.println("Collection:"+a1);

Function<Integer,Integer>p=(i)->{

**if**(i>20){

**return** i;

}

**else** {

**return** i;

}

};

Comparator <Integer> c=(a,b)->{

**if**((Integer)a>(Integer)b) {

**return** 1;

}

**else**

{

**return** -1;

}

};

Comparator <Integer> d=(a,b)->{

**if**((Integer)a>(Integer)b) {

**return** 1;

}

**else**

{

**return** -1;

}

};

Stream <Integer> s=a1.stream();

s=s.map(p);

Stream <Integer> v=a1.stream();

v=v.map(p);

System.***out***.println("Max Number:"+s.max(c));

System.***out***.println("Min Number:"+v.min(d));

}

}

Output:

--------

Collection:[10, 50, 30, 40, 20]

Max Number:Optional[50]

Min Number:Optional[10]

**5.toArray():-** we use this method to copy the elements present in stream into array.

Example:-

**import** java.util.ArrayList;

**import** java.util.Comparator;

**import** java.util.List;

**import** java.util.function.Function;

**import** java.util.stream.Collectors;

**import** java.util.stream.Stream;

**class** suku{

**public** **static** **void** main(String args[])

{

ArrayList<Integer> a1=**new** ArrayList();

a1.add(10);

a1.add(50);

a1.add(30);

a1.add(40);

a1.add(20);

System.***out***.println("Collection:"+a1);

Function<Integer,Integer>p=(i)->{

**if**(i>20){

**return** i;

}

**else** {

**return** i;

}

};

Comparator <Integer> c=(a,b)->{

**if**((Integer)a>(Integer)b) {

**return** 1;

}

**else**

{

**return** -1;

}

};

Stream <Integer> s=a1.stream();

s=s.map(p);

Integer [] i=(Integer[]) s.toArray(Integer[]::**new**);

System.***out***.println("Array:");

**for**(**int** j=0;j<i.length;j++) {

System.***out***.println(i[j]);

}

}

}

Output:

Collection:[10, 50, 30, 40, 20]

Array:

10

50

30

40

20

**6.of():-** we can also apply a stream for group of values and for arrays.

Example:

**import** java.util.List;

**import** java.util.stream.Collectors;

**import** java.util.stream.Stream;

**class** suku{

**public** **static** **void** main(String args[])

{

Stream <Integer>s= Stream.*of*(10,20,30);

System.***out***.println(s);

List<Integer> l=(List<Integer>)s.collect(Collectors.*toList*());

System.***out***.println("Created Stream:"+l);

}

}

Output:

--------

java.util.stream.ReferencePipeline$Head@5caf905d

Created Stream:[10, 20, 30]