The Stream API enables developers to create the parallel streams that can take advantage of multi-core architectures and enhance the performance of Java code.

**import** java.util.Arrays;

**import** java.util.List;

**public** **class** ListDemo {

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

{

List<Integer> list=Arrays.*asList*(1,2,3,4,5,6,7,98,9,8,9);

list.stream().forEach(System.***out***::print);

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

list.parallelStream().forEach(System.***out***::print);

}

}

Output:

123456798989

798345128996

Parallel Streams can be used in case of aggregate functions

Parallel Streams can be used if developers have performance implications with the Sequential Streams

-------flat Map---

flatMap() is the combination of a map and a flat operation i.e, it applies a function to elements as well as flatten them. flatMap() is used for both transformation and flattening

**import** java.util.Arrays;

**import** java.util.Collection;

**import** java.util.List;

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

**public** **class** ListDemo {

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

{

List odd=Arrays.*asList*(1,3,5);

List even=Arrays.*asList*(0,2,4);

List listoflist=Arrays.*asList*(odd,even);

System.***out***.println("before flattening"+listoflist);

List listoflist1 = (List) listoflist.stream().flatMap(list -> ((List) list).stream())

.collect(Collectors.*toList*());

System.***out***.println(listoflist1);

}

}

Output:.

before flattening[[1, 3, 5], [0, 2, 4]]

[1, 3, 5, 0, 2, 4]

---------peek() method---------------

Peek() is used to return the top of element in stack

**import** java.util.Arrays;

**import** java.util.Collection;

**import** java.util.List;

**import** java.util.Stack;

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

**public** **class** ListDemo {

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

{

Stack<String> stack1= **new** Stack<String>();

stack1.push("hi");

stack1.push("hello");

stack1.push("welcome");

System.***out***.println("intial stack::"+stack1);

System.***out***.println("the element at the top of stack::::"+stack1.peek());

}

}

Output:

intial stack::[hi, hello, welcome]

the element at the top of stack::::welcome

-----------distinct()------------------

distinct() is the method of **Stream** interface.

This methoduses hashCode() and equals() methods to get distinct elements

**import** java.util.Arrays;

**import** java.util.Collection;

**import** java.util.List;

**import** java.util.Stack;

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

**public** **class** ListDemo {

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

{

List list=Arrays.*asList*(1,2,3,4,4,43,2,1);

list.stream().distinct().forEach(System.***out***::println);

}

}

Output:

1

2

3

4

43

-----try with finally block----------

class JavaFinally

{

public static void main(String args[])

{

System.out.println(JavaFinally.myMethod());

}

public static int myMethod()

{

try {

return 112;

}

finally {

System.out.println("This is Finally block");

System.out.println("Finally block ran even after return statement");

}

}

}

Class Example

{

Public static void main(String args[])

{

System.out.print(Example.myMethod());

}

Public myMethod()

{

try

{

Return 89;

}

Finally

{

System.out.prtinln(“final Block”);

}

}

Output:

Final block

---------tostring() and equals()-------------------

Purpose of **Overriding toString**() Method in **Java**

If **we want** to represent an object of a class as a String, then **we** can use the **toString**() method which returns a textual representation of the object.

The String class overrides the **equals method** it inherited from the Object class and implemented logic to compare the two String objects character by character. The reason the **equals method** in the Object class does reference equality is because it does not know how to do anything else.

---------------string reverse with and without methods-------------

**package** product;

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

**public** **class** ExampleString {

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

{

/\*StringBuilder sb=new StringBuilder("maruthi");

sb.reverse();

System.out.println(sb);\*/

String svalue="maruthimarripati";

String R="";

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

R= svalue.charAt(i) + R;

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

}

}