1. How to traverse ArrayList?

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

al.add(1);

al.add(3);

al.add(2);

Iterator it = al.iterator();

**while**(it.hasNext()){

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

}

**for**(Integer i:al){

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

}

al.forEach(a ->{

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

});

2. Traverse List in reverse order?

ListIterator it = al.listIterator(al.size());

**while**(it.hasPrevious()){

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

}

3. Sort ArrayList?

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

al.add(1);

al.add(3);

al.add(2);

Collections.*sort*(al); // reverse

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

4. Sort ArrayList in descending order?

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

al.add(1);

al.add(3);

al.add(2);

Collections.*sort*(al,Collections.*reverseOrder*());

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

5. Comapre Two ArrayList?

System.***out***.println(al.equals(al1));

6. Remove duplicate from ArrayList?

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

al.add(1);

al.add(3);

al.add(2);

al.add(2);

Set s= **new** LinkedHashSet(al);

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

Java 8

List list=al.stream().distinct().collect(Collectors.*toList*());

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

7. How to synchronized ArrayList?

Collections.sysncronizedList();

8. How to Sort Map? Using Key?

Map<Integer,String> hm= **new** LinkedHashMap<Integer,String>();

hm.put(1, "r");

hm.put(3, "m");

hm.put(2, "p");

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

TreeMap<Integer,String> tm = **new** TreeMap<Integer,String>(hm);

Set s= tm.entrySet();

Iterator it=s.iterator();

**while**(it.hasNext()){

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

}

9. Sort Map Key value (descending)?

hm.entrySet().stream().sorted(Map.Entry.*comparingByValue*(Comparator.*reverseOrder*())).forEach(System.***out***::println);

10.Comparable?

//comparable

package com.coll;

import java.util.ArrayList;

import java.util.Collections;

public class Student implements Comparable<Student>{

int rollNo;

String name;

public Student(int rollNo, String name) {

super();

this.rollNo = rollNo;

this.name = name;

}

@Override

public int compareTo(Student o) {

//return this.name.compareTo(o.name);

if(this.rollNo>o.rollNo){

return 1;

}else if(this.rollNo<o.rollNo){

return -1;

}else

return 0;

}

public static void main(String[] args) {

Student s1 = new Student(12,"R");

Student s2 = new Student(6,"A");

Student s3 = new Student(24,"N");

Student s4 = new Student(3,"P");

ArrayList<Student> al = new ArrayList<Student>();

al.add(s1);

al.add(s2);

al.add(s3);

al.add(s4);

System.out.println("Beofor");

for(Student student:al){

System.out.println(student.rollNo+" "+student.name);

}

System.out.println("After");

Collections.sort(al,Collections.reverseOrder());

for(Student student:al){

System.out.println(student.rollNo+" "+student.name);

}

}

}

11. Comparator?

public class Person {

int rollNo;

String name;

public Person(int rollNo, String name) {

super();

this.rollNo = rollNo;

this.name = name;

}

public static void main(String[] args) {

Person p1 = new Person(12, "R");

Person p2 = new Person(22, "N");

Person p3 = new Person(16, "A");

Person p4 = new Person(3, "Z");

ArrayList<Person> al = new ArrayList<Person>();

al.add(p1);

al.add(p2);

al.add(p3);

al.add(p4);

System.out.println("no");

Collections.sort(al, new RollNoSort());

for (Person p : al) {

System.out.println(p.rollNo + " " + p.name);

}

System.out.println("name");

Collections.sort(al, new NameSort().reversed());

for (Person p : al) {

System.out.println(p.rollNo + " " + p.name);

}

}

}

class RollNoSort implements Comparator<Person> {

@Override

public int compare(Person o1, Person o2) {

if (o1.rollNo > o2.rollNo) {

return 1;

} else if (o1.rollNo < o2.rollNo) {

return -1;

} else

return 0;

}

}

class NameSort implements Comparator<Person> {

@Override

public int compare(Person o1, Person o2)

return o1.name.compareTo(o2.name);

}

}class NameSort implements Comparator<Person>{

@Override

public int compare(Person o1, Person o2) {

return o1.name.compareTo(o2.name);

}

}

// sum of list

**public** **class** Test {

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

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

al.add(3);

al.add(1);

al.add(5);

al.add(2);

**int** sum = al.stream().mapToInt(Integer::intValue).sum();

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

}

}

**public** **class** Test {

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

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

al.add(3);

al.add(1);

al.add(5);

al.add(2);

**int** sum = al.stream().mapToInt(Integer::intValue).sum();

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

}

}

// age start from 2

ArrayList<Student> al = **new** ArrayList();

rollNo ,name,age

al.add(**new** Student(2,"rahul",25));

al.add(**new** Student(1,"mau",4));

al.add(**new** Student(5,"shlok",10));

al.add(**new** Student(3,"divya",22));

List<Student> collect = al.stream().filter(i->Integer.*toString*(i.age).startsWith("2")).collect(Collectors.*toList*());

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

//average /sum /min of age

ArrayList<Student> al = **new** ArrayList();

al.add(**new** Student(2,"rahul",25));

al.add(**new** Student(1,"mau",4));

al.add(**new** Student(5,"shlok",10));

al.add(**new** Student(3,"divya",22));

**double** asDouble = al.stream().mapToInt(Student::getAge).average().getAsDouble();

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

2. Duplicate entry remove

ArrayList<Student> al = **new** ArrayList();

al.add(**new** Student(2, "rahul", 25));

al.add(**new** Student(1, "mau", 4));

al.add(**new** Student(5, "shlok", 10));

al.add(**new** Student(3, "shlok", 22));

Set<String> seen = **new** HashSet<>();

Predicate<Student> p =i -> !seen.add(i.getName());

List<Student> collect = al.stream().filter(p).collect(Collectors.*toList*());

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

Salary Start from

Predicate<Employee> p = i->i.salary.toString().startsWith("5");

List<Employee> collect = al.stream().filter(p).collect(Collectors.*toList*());

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

Int to Integer :-

Predicate<Student> p = i->Integer.valueOf(i.age).toString().startsWith("2");

Spring mvc configuration

[Angular and Spring MVC CRUD Tutorial - Course Introduction - Part 0 (youtube.com)](https://www.youtube.com/watch?v=JUZYCNbZfCA&list=PLA7e3zmT6XQV79HeRtXFcPmFcAi_icKlh)