**4package** com.te.collection;

**import** java.util.ArrayList;

**import** java.util.Collection;

**public** **class** Coll {

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

Collection<Integer> collection = **new** ArrayList<Integer>();

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

System.***out***.println("--------add---------");

collection.add(10);

collection.add(20);

collection.add(30);

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

Collection<Integer> c = **new** ArrayList<Integer>();

c.add(40);

c.add(50);

c.add(100);

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

System.***out***.println("--------addAll-------");

System.***out***.println(collection.addAll(c));

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

System.***out***.println("--------isEmpty-------");

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

System.***out***.println("----remove----");

System.***out***.println(collection.remove(30));

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

List<Integer> sublist = collection.subList(0, 3);

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

System.***out***.println(collection.indexOf(20));

System.***out***.println(collection.lastIndexOf(40));

}

}

Comparable Interface:

Java Comparable interface is used to order the objects of the user-defined class. This interface is found in java.lang package and contains only one method named compareTo(Object). It provides a single sorting sequence only, i.e., you can sort the elements on the basis of single data member only. For example, it may be rollno, name, age or anything else.

compareTo(Object obj) method

**public int compareTo(Object obj):** It is used to compare the current object with the specified object. It returns

* positive integer, if the current object is greater than the specified object.
* negative integer, if the current object is less than the specified object.
* zero, if the current object is equal to the specified object.

**package** com.te.collection;

**public** **class** Person **implements** Comparable<Person> {

**int** id;

String name;

**public** Person(**int** id, String name) {

**super**();

**this**.id = id;

**this**.name = name;

}

@Override

**public** **int** compareTo(Person p) {

**return** **this**.id - p.id;

or

**return** **this**.hashCode() – p.hashCode(); (because in

hashcode method

we are returning

id.)

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

}

@Override

**public** String toString() {

**return** "Person [id=" + id + ", name=" + name + "]";

}

@Override

**public** **int** hashCode() {

**return** **this**.id;

}

}

**package** com.te.collection;

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** ExOfComparableType {

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

ArrayList<Person> array = **new** ArrayList<Person>();

array.add(**new** Person(10, "cde"));

array.add(**new** Person(40, "Abc"));

array.add(**new** Person(80, "xyz"));

array.add(**new** Person(8, "oop"));

System.***out***.println("---- Before Sorting----");

**for** (Person person : array) {

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

}

System.***out***.println("---- After Sorting----");

Collections.*sort*(array);

**for** (Person person : array) {

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

}

}

}

Comparator interface:

* used to compare two objects
* compare() with two arguments

Compare() returns

* +ve if 1st > 2nd
* -ve if 1st < 2nd
* 0 if both are equal

**package** com.te.collection;

**import** java.util.Comparator;

**public** **class** Student **implements** Comparator<Student> {

**int** id;

String name;

**public** Student() {

}

**public** Student(**int** id, String name) {

**super**();

**this**.id = id;

**this**.name = name;

}

**public** **int** compare(Student s1, Student s2) {

// By ID

**return** s1.id - s2.id;

// Alphabetically

// return s1.name.compareTo(s2.name);

// length of name

// return s1.name.length()-(s2.name.length());

}

@Override

**public** String toString() {

**return** "Student [id=" + id + ", name=" + name + "]";

}

}

**package** com.te.collection;

**import** java.util.\*;

**public** **class** LinkedListExOfComparator {

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

LinkedList<Student> linkedlist = **new** LinkedList<Student>();

linkedlist.add(**new** Student(1, "nilim"));

linkedlist.add(**new** Student(2, "naveen"));

linkedlist.add(**new** Student(4, "aakash"));

linkedlist.add(**new** Student(10, "bunny"));

linkedlist.add(**new** Student(3, "Aamar"));

Collections.*sort*(linkedlist, **new** Student());

**for** (Student student : linkedlist) {

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

}

}

}

START FORM HERE:

ITERATOR:

Iterator interface is used to iterate elements from Iterable types.

🡪 Three methods:

🡪next():Object or <E> 🡪 is used to iterate elements from the underlying collection

🡪hasNext(): Boolean🡪 returns true if the next elements is available for iteration else returns false.

🡪 remove():void🡪 removes the last iterated element.

Iterable interface provides a factory method iterator(); which returns and Iterator type object.

Iterator is uni-directional.

ListIterator interface:

🡪 used to iterate elements only from list type

🡪 is bi—directional

🡪 List interface provides a factory method listIterator() which returns a ListIterator type object

9 methods:

🡪 next(): object

🡪 hasNext():boolean

🡪 previous(): object

🡪hasPrevious():Boolean

🡪add(object e):void

🡪set(object e):void

🡪nextIndex ():int

🡪previousIndex ():int

🡪remove ():void

ARRAYLIST IMPL:

package com.tyss.mycollection.common;

import java.util.Iterator;

public class MyArray<E> implements Iterable<E> {

Object[] array;

int arraysize;

private int position;

public MyArray(int arraysize) {

array = new Object[arraysize];

}

private void increaseCapacity() {

Object[] temp = new Object[array.length + 3];

System.arraycopy(array, 0, temp, 0, array.length - 1);

array = temp;

}

public void add(int e) {

if (position >= array.length - 1)

increaseCapacity();

array[position] = e;

position++;

}

public Object get(int index) {

return array[index];

}

public void remove(int index) {

for (int i = index + 1; i < array.length; i++) {

array[i - 1] = array[i];

}

array[position] = null;

position--;

}

public int size() {

return position;

}

@Override

public String toString() {

if (size() == 0)

return "[]";

String s = "[" + array[0];

for (int i = 1; i < array.length; i++) {

s += "," + array[i];

}

s += "]";

return s;

}

@Override

public Iterator<E> iterator() {

return new MyItr();

}

private class MyItr implements Iterator<E> {

int index;

@Override

public boolean hasNext() {

return (index < position) ? true : false;

}

@Override

public E next() {

@SuppressWarnings("unchecked")

E e = (E) array[index++];

return e;

}

}

}

SET

Difference between list and set:

🡪 in set duplicates are not allowed

🡪 not an indexed collection.

HashSet:

Order not guaranteed.

LinkedHashSet:

According to insertion order

TreeSet:

According to sorted order using comparable or comparator inpl.

// This is one way to sort HashSet by using ArrayList

package com.te.collection;

public class EmployeeHashset implements Comparable<EmployeeHashset> {

int eid;

String ename;

String edept;

double esalary;

public EmployeeHashset(int eid, String ename, String edept, double esalary) {

super();

this.eid = eid;

this.ename = ename;

this.edept = edept;

this.esalary = esalary;

}

@Override

public String toString() {

return " [eid=" + eid + ", ename=" + ename + ", edept=" + edept + ", esalary=" + esalary + "]";

}

@Override

public int hashCode() {

return this.eid;

}

// @Override

// public boolean equals(Object obj) {

// EmployeeHashset employeeHashset= (EmployeeHashset) obj;

// return this.eid== employeeHashset.eid;

// }

@Override

public int compareTo(EmployeeHashset o) {

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

return this.hashCode()- o.hashCode();

}

}

//Example of sorting a HashSet using ArrayList

package com.te.collection;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashSet;

import java.util.Iterator;

public class HashSetImpl {

public static void main(String[] args) {

HashSet<EmployeeHashset> employeeHashset = new HashSet<EmployeeHashset>();

// EmployeeHashset employeeHashset1 = new EmployeeHashset(106, "Naveen", "Development", 20000);

// EmployeeHashset employeeHashset2 = new EmployeeHashset(102, "Nilim", "Development", 25000);

// EmployeeHashset employeeHashset3 = new EmployeeHashset(108, "Akash", "Testing", 18000.20);

// EmployeeHashset employeeHashset4 = new EmployeeHashset(106, "Bunny", "Trainer", 50000.20);

// EmployeeHashset employeeHashset5 = new EmployeeHashset(104, "Kishore", "Manager", 70000.20);

employeeHashset.add(new EmployeeHashset(106, "Vaveen", "Development", 20000));

employeeHashset.add(new EmployeeHashset(102, "Nilim", "Development", 45000));

employeeHashset.add(new EmployeeHashset(101, "Cilim", "Development", 35000));

employeeHashset.add(new EmployeeHashset(107, "Ailim", "Development", 65000));

employeeHashset.add(new EmployeeHashset(103, "Zilim", "Development", 15000));

Iterator iterator = employeeHashset.iterator();

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

ArrayList<EmployeeHashset> list = new ArrayList<EmployeeHashset>(employeeHashset);

Collections.sort(list);

System.out.println("After Sorting");

Iterator<EmployeeHashset> iterator1 = list.iterator();

while (iterator1.hasNext()) {

System.out.println(iterator1.next());

}

}

}

//Example of sorting a LinkedHashSet using TreeSet

package com.te.collection;

import java.util.Comparator;

public class LinkedHashSetByName implements Comparator<EmployeeHashset> {

@Override

public int compare(EmployeeHashset o1, EmployeeHashset o2) {

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

}

}

//Example of sorting a LinkedHashSet using TreeSet

package com.te.collection;

import java.util.Iterator;

import java.util.LinkedHashSet;

import java.util.TreeSet;

public class HashSetImpl {

public static void main(String[] args) {

LinkedHashSet<EmployeeHashset> employeeHashset = new LinkedHashSet<EmployeeHashset>();

employeeHashset.add(new EmployeeHashset(106, "Vaveen",

"Development", 20000));

employeeHashset.add(new EmployeeHashset(102, "Nilim",

"Development", 45000));

employeeHashset.add(new EmployeeHashset(101, "Cilim", "Development", 35000));

employeeHashset.add(new EmployeeHashset(107, "Ailim", "Development", 65000));

employeeHashset.add(new EmployeeHashset(103, "Zilim", "Development", 15000));

LinkedHashSetByName byName = new LinkedHashSetByName();

TreeSet<EmployeeHashset> treeSet = new TreeSet<EmployeeHashset>(byName);

treeSet.add(new EmployeeHashset(106, "Vaveen", "Development", 20000));

treeSet.add(new EmployeeHashset(102, "Nilim", "Development", 45000));

treeSet.add(new EmployeeHashset(101, "Cilim", "Development", 35000));

treeSet.add(new EmployeeHashset(107, "Ailim", "Development", 65000));

treeSet.add(new EmployeeHashset(103, "Zilim", "Development", 15000));

Iterator iterator = treeSet.iterator();

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

// ArrayList<EmployeeHashset> list = new ArrayList<EmployeeHashset>(employeeHashset);

// Collections.sort(list);

// System.out.println("After Sorting");

// Iterator<EmployeeHashset> iterator1 = list.iterator();

// while (iterator1.hasNext()) {

// System.out.println(iterator1.next());

// }

}

}

SORTED SET METHODS:

package com.te.collection;

import java.util.SortedSet;

import java.util.TreeSet;

public class SetEx {

public static void main(String[] args) {

TreeSet set = new TreeSet();

set.add(10);

set.add(40);

set.add(30);

set.add(20);

set.add(60);

set.add(40);

System.out.println("Smallest Element: " + set.first());

System.out.println("Largest element : " + set.last());

System.out.println("---------------");

SortedSet set2 = set.headSet(60);

for (Object object : set2) {

System.out.println(object + " ");

}

System.out.println("---------------");

SortedSet set3 = set.tailSet(20);

for (Object object : set3) {

System.out.println(object + " ");

}

System.out.println("---------------");

SortedSet set4 = set.subSet(20, 60);

for (Object object : set4) {

System.out.println(object + " ");

}

}

}

Smallest Element: 10

Largest element : 60

---------------

10

20

30

40

---------------

20

30

40

60

---------------

20

30

40

[10:52 am, 23/06/2021] Nilim Sarma: package com.te.setexamples;

import java.util.LinkedHashSet;

import java.util.TreeSet;

public class LinkedHashSetImpl {

public static void main(String[] args) {

LinkedHashSet<Student> hashSet = new LinkedHashSet<Student>();

Student student1 = new Student(10, "Sam", 17);

Student student2 = new Student(16, "Jam", 19);

Student student3 = new Student(9, "Pam", 20);

Student student4 = new Student(43, "Mam", 16);

Student student5 = new Student(23, "Cam", 10);

hashSet.add(student1);

hashSet.add(student2);

hashSet.add(student3);

hashSet.add(student4);

hashSet.add(student5);

System.out.println(hashSet);

Student student = new Student();

TreeSet<Student> set = new TreeSet<Student>(student);

set.add(student1);

set.add(st…

package com.te.setexamples;

import java.util.Comparator;

public class Student implements Comparator<Student> {

int id;

String name;

int age;

public Student() {

super();

}

public Student(int id, String name, int age) {

super();

this.id = id;

this.name = name;

this.age = age;

}

@Override

public int compare(Student o1, Student o2) {

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

}

@Override

public String toString() {

return "Student [id=" + id + ", name=" + name + ", age=" + age + "]";

}

}