DAY-67

-------

COLLECTION

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

Collection is an object, it able to represent a group of other objects.

Q] In java applications, to represent a group of other elements we already have arrays then what is the need of collections?

OR

Q] What are the diff b/w Array and Collection?

Ans:

---->

1.Arrays are having fixed size in nature.Incase of arrays we are able to add the elements upto the specified size only, we are unable to add the

elements over its size, if we are trying to add elements over its size then JVM will rise an exception like "ArrayIndexOutOfBoundsException"

EXAMPLE:1

----------

class Demo1

{

public static void main(String[] args)

{

Demo1[] d = new Demo1[3];

d[0] = new Demo1();

d[1] = new Demo1();

d[2] = new Demo1();

d[3] = new Demo1(); // ArrayIndexOutOfBoundsException

}

}

output:

-----------

java.lang.ArrayIndexOutOfBoundsException: 3

-->Collection are dynamically growable nature,even if we add the elements over its size then the JV< will not rise an exception.

EXAMPLE-2

---------

import java.util.\*;

class Demo1

{

public static void main(String[] args)

{

ArrayList al = new ArrayList(3);

al.add(new Demo1());

al.add(new Demo1());

al.add(new Demo1());

al.add(new Demo1());

}

}

2.In java, by default Arrays are able to allow homogenous elements if we are trying to add the elements which are not same array data type then

compiler will rise an error like "incompatible types"

EXAMPLE:3

----------

import java.util.\*;

class sample

{

}

class Demo1

{

public static void main(String[] args)

{

Demo1[] d = new Demo1[2];

d[0] = new Demo1();

d[0] = new sample();

}

}

output:

--------

incompatible types: sample cannot be converted to Demo1

d[0] = new sample();

In java, Collection are able to allow heterogenous elements even we add different types of elements compiler will not rise an error

EXAMPLE:4

----------

import java.util.\*;

class India

{

}

class People

{

}

class Demo1

{

public static void main(String[] args)

{

ArrayList al = new ArrayList(4);

al.add(new Demo1());

al.add(new Demo1());

al.add(new India());

al.add(new People());

}

}

3.Arrays are not having any predefiend methods to perform searching and sorting operation over the elements, in case of arrays to perform seraching

and sorting operations develeopers need to provide thier own logic.

-->In collection predefiend methods or predefiend collections to perform seraching and sorting operation over the elements.

example:In collections, TreeSet was provide to perform sorting order.

Example:5

----------

import java.util.\*;

class Demo1

{

public static void main(String[] args)

{

TreeSet ts = new TreeSet();

ts.add("z");

ts.add("f");

ts.add("s");

ts.add("c");

System.out.println(ts);

}

}

output:

-------

[c, f, s, z]

4.Arrays are able to allow only one type of elements so arrays are able to improve the typedness in java application and they are to perform typesafe

operations.

-->Collections are able to allow diff types of elements so collection are able to reduce the typedness in java application and they are unable to perform

typesafe operations.

5.If we know the no of elements in advance at the time of writing java applications then Arrays are better to use in java applications and they will

provide very good performance in java applications, but, Arrays are not flexible to design applications.

In java applications, Collections are able to provide less performance, but, They will provide flexibility to design applications

'o' repersent Collection objects in java applications , JAVA has provided predefined classes and interfaces in the form of java.util package

called as "Collection Framework".

Q)What are the classes and interfaces are existed in java.util package to repersent Collections?

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

--- Diagram---

pic1

pic2

Collection:

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

--> It is an interface provided by JAVA along with JDK1.2 version.

--> It able to represent a group of individual elements as single unit.

--> It has provided the following methods common to every implementation class.

1.public boolean add(Object obj)

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

-->This method is able to add the specified element to Collection object. If the specified element is added successfully then add(-) method will

return "true" value. If the specified element is not addedd successfully then add() method will return "false" value.

//example-6

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

import java.util.\*;

class Demo2

{

public static void main(String[] args)

{

HashSet hs=new HashSet();

System.out.println(hs.add("A"));

hs.add("B");

hs.add("C");

hs.add("D");

System.out.println(hs);

System.out.println(hs.add("A"));

System.out.println(hs);

}

}

OutPut:

-------

true

[A,B,C,D]

false

[A,B,C,D]

2.public boolean addAll(Collection c)

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

-->This method can be used to add all the elements of the specified Collection to the present

Collection object. If addition operation is success then addAll(-) method will return "true" value, if

addition operation is failure then addAll() method will return "false" value.

//example-7

-----------

import java.util.\*;

class Demo2

{

public static void main(String[] args)

{

HashSet hs=new HashSet();

hs.add("A");

hs.add("B");

hs.add("C");

hs.add("D");

System.out.println(hs);

HashSet hs1=new HashSet();

System.out.println(hs1.addAll(hs));

System.out.println(hs1);

System.out.println(hs1.addAll(hs));

System.out.println(hs1);

}

}

Output:

-------

[A, B, C, D]

true

[A, B, C, D]

false

[A, B, C, D]

3.public boolean remove(Object obj)

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

--> This method can be used to remove the specified element from the Collection object. If remove operation is success then remove() method

will return true value, if remove operation is failure then remove() method will return false value.

//example-8

-----------

import java.util.\*;

class Demo2

{

public static void main(String[] args)

{

ArrayList al=new ArrayList();

al.add("A");

al.add("B");

al.add("C");

al.add("D");

System.out.println(al);

System.out.println(al.remove("C"));

System.out.println(al);

System.out.println(al.remove("C"));

System.out.println(al);

}

}

OutPut:

--------

[A,B,C,D]

true

[A,B,D]

false

[A,B,D]

4.public boolean removeAll(Collection c)

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

-->This method can be used to remove all the elements of the specified Collection from the present

Collection object. If remove operation is success then removeAll() method will return true value. If

remove operation is not success then removeAll() method will return false value.

//example-9

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

import java.util.\*;

class Demo2

{

public static void main(String[] args)

{

ArrayList al=new ArrayList();

al.add("A");

al.add("B");

al.add("C");

al.add("D");

al.add("E");

al.add("F");

System.out.println(al);

ArrayList al1=new ArrayList();

al1.add("B");

al1.add("C");

al1.add("D");

System.out.println(al1);

System.out.println(al.removeAll(al1));

System.out.println(al);

System.out.println(al.removeAll(al1));

System.out.println(al);

}

}

output:

-------

[A, B, C, D, E, F]

[B, C, D]

true

[A, E, F]

false

[A, E, F]

5.public boolean contains(Object obj)

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

--> This method will check whether the specified element is existed or not in the Collection object.

If the specified element is existed then this method will return "true" value . If the specified element

is not existed then this method will return "false" value.

//Example-10

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

import java.util.\*;

class Demo2

{

public static void main(String[] args)

{

ArrayList al=new ArrayList();

al.add("A");

al.add("B");

al.add("C");

al.add("D");

al.add("E");

al.add("F");

System.out.println(al);

System.out.println(al.contains("B"));

System.out.println(al.contains("X"));

}

}

Output:

--------

[A,B,C,D,E,F]

true

false

6.public boolean containsAll(Collection c)

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

-->This method will check whether all the elements of the specified Collection are available or not

in the present Collection object. If all the elements are existed then containsAll() method will return

true value, if atleast one element is not existed then containsAll() method will return false value.

//Example-11

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

import java.util.\*;

class Demo1

{

public static void main(String[] args)

{

ArrayList al=new ArrayList();

al.add("A");

al.add("B");

al.add("C");

al.add("D");

al.add("E");

al.add("F");

System.out.println(al);

ArrayList al1=new ArrayList();

al1.add("B");

al1.add("C");

al1.add("D");

System.out.println(al.containsAll(al1));

al1.add("X");

al1.add("Y");

System.out.println(al.containsAll(al1));

System.out.println(al);

System.out.println(al1);

}

}

output:

--------

[A, B, C, D, E, F]

true

false

[A, B, C, D, E, F]

[B, C, D, X, Y]

7.public boolean retainAll(Collection c)

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

-->This method will remove all the elements from the present Collection object except the elements

which are existed in the specified Collection object. if atleast one element is removed then

retainAll() method will return true value. If no elements are removed then retainsAll() method will

return false value.

//Example-12

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

import java.util.\*;

class Demo2

{

public static void main(String[] args)

{

ArrayList al=new ArrayList();

al.add("A");

al.add("B");

al.add("C");

al.add("D");

al.add("E");

al.add("F");

System.out.println(al);

ArrayList al1=new ArrayList();

al1.add("B");

al1.add("C");

al1.add("D");

System.out.println(al1);

System.out.println(al.retainAll(al1));

System.out.println(al);

}

}

OutPut:

--------

[A,B,C,D,E,F]

[B,C,D]

true

[B,C,D]

8.public int size()

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

--> This method can be used to return an integer value representing the no of elements which are existed in the Collection object.

9.public void clear()

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

--> This method can be used to remove all elements from Collection objectt.

10.public boolean isEmpty()

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

--> This method can be used to check whether Collection objectt is empty or not.If the Collection

object is empty then isEmpty() method will return "true" value. If the Collection object is not empty

then isEmpty() method will return "false" value.

11.public Object[] toArray()

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

--> This method will return all the elements of the Collection object in the form of Object[].

//example-12

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

import java.util.\*;

class Demo2

{

public static void main(String[] args)

{

ArrayList al=new ArrayList();

al.add("A");

al.add("B");

al.add("C");

al.add("D");

al.add("E");

al.add("F");

System.out.println(al);

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

Object[] obj=al.toArray();

for(Object o: obj)

{

System.out.print(o+" ");

}

System.out.println();

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

al.clear();

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

System.out.println(al);

}

}

OutPut:

-------

[A,B,C,D,E,F]

6

A B C D E F

false

true

[]