SYNTAX
TECHNOLOGIES

JAVA

Class 30

Agenda

ArrayList vs Array
LinkedList
Set Interface

User-defined class objects in Java ArrayList

```
class Student{
  int rollno;
  String name;
  int age;
  Student(int rollno,String name,int
age)
{
   this.rollno=rollno;
   this.name=name;
   this.age=age;
  }
}
```

```
import java.util.*;
public class TestCollection3{
 public static void main(String args[]){
  Student s1=new Student(101,"Sonoo",23);
  Student s2=new Student(102,"Ravi",21);
  Student s2=new Student(103,"Hanumat",25);

  ArrayList<Student> al=new ArrayList<Student>();
  al.add(s1);//adding Student class object
  al.add(s2);
  al.add(s3);

  Iterator itr=al.iterator();
  //traversing elements of ArrayList object
  while(itr.hasNext()) {
    Student st=(Student)itr.next();
    System.out.println(st.rollno+" "+st.name+" "+st.age);
  }
 }
}
```

ArrayList vs Arrays

1. ArrayList is the best alternative to Arrays in Java as ArrayList is Dynamic data structure and Arrays are the Static data structure.

2. Array can contain both primitives and objects but ArrayList can contain only object elements.

3. You can use length variable to calculate length of an array but size() method to calculate size of ArrayList.

4. Array use assignment operator to store elements but ArrayList use add() to insert elements.

5. Array can be multi dimensional , while ArrayList is always single dimensional but we can create arrays of arrays.

LinkedList

LinkedList is a class which implements the List interface of collection framework.

Like ArrayList, LinkedList also can contain duplicate elements and also maintain the insertion order.

We can declare the LinkedList same like ArrayList and perform operations like insertion and deletion are same like ArrayList.

The only difference in LinkedList and ArrayList is, LinkedList stores elements in a doubly-linked list data structure while ArrayList stores elements in a backing array.

ArrayList vs LinkedList

ArrayList	LinkedList
1) ArrayList internally uses **dynamic array** to store the elements.	LinkedList internally uses **doubly linked list** to store the elements.
2) Manipulation with ArrayList is **slow** because it internally uses array. If any element is removed from the array, all the bits are shifted in memory.	Manipulation with LinkedList is **faster** than ArrayList because it uses doubly linked list so no bit shifting is required in memory.
3) ArrayList class can **act as a list** only because it implements List only.	LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces.
4) ArrayList is **better for accessing** data.	LinkedList is **better for storing and removing** data.
5) ArrayList maintains indexes and element data	LinkedList maintains element data and two pointers for neighbor nodes hence the memory consumption is high in LinkedList comparatively.

Task

Create a class Insurance that will have an attribute as insuranceName and unimplemented behaviour as getQuote and cancelInsurance. Create 3 subclasses Car, Pet, Health. Car class has it's own attribute as carModel and Class Pet has petType attribute. Create 3 objects of the sub classes and store them in ArrayList. Using for loop/advanced for loop/ iterator access all methods of the class.

Create a Card class that will have interest rate field and card type

and a constructor that will initialize the fields.

Create 3 objects of different card and store them into LinkedList.

Using for loop/advanced for loop/ iterator access all methods of the class.

Set

Set is a Collection that can’t contain duplicate elements.

There are three main implementations of Set interface:

- HashSet
- LinkedHashSet
- TreeSet

Set

HashSet, class that implements Set interface. It does not allow duplicates and does not guarantee any insertion orders. It allows null elements.

LinkedHashSet, class that implements Set interface. It does not allow duplicates and orders its elements based on the order in which they were inserted.

TreeSet, is similar to HashSet except that it sorts the elements in the ascending order while HashSet doesn’t maintain any order.

Hashset in Java

- **HashSet** is the class, which implements the Set interface in Java.
- The HashSet does not add any additional methods beyond those found in the Set interface.
- **HashSet does not guarantee any insertion orders of the set** but it allows null elements.
- HashSet can be used in place of ArrayList to store the object if you require no duplicate and don’t care about insertion order.
- HashSet doesn’t allow duplicates. If you try to add a duplicate element in HashSet, the old value would be overwritten.

Methods in Hashset

- **boolean add(Element e)**: It adds the element e to the list.
- **void clear():** It removes all the elements from the list.
- **boolean contains(Object o):** It checks whether the specified Object o is present in the list or not. If the object has been found it returns true else false.
- **boolean isEmpty():** Returns true if there is no element present in the Set.
- **Iterator iterator():** Used to return an iterator over the element in the set.
- **int size():** It gives the number of elements of a Set.
- **Boolean remove (Object o):** It removes the specified Object from the Set.
- **Object clone():** This method returns a shallow copy of the HashSet.

NOTE: A Set doesn't provide any method for data retrieval.

```
import java.util.HashSet;
public class HashSetExample {
   public static void main(String args[]) {
HashSet<String> hset = new HashSet<String>();

     // Adding elements to the HashSet
     hset.add("Apple");
     hset.add("Mango");
     hset.add("Grapes");
     hset.add("Orange");
     hset.add("Fig");
     //Addition of duplicate elements
     hset.add("Apple");
     hset.add("Mango");
     //Addition of null values
     hset.add(null);
     hset.add(null);

     //Displaying HashSet elements
     System.out.println(hset);
    }
}
```

```
import java.util.HashSet;
import java.util.Iterator;

class IterateHashSet{
  public static void main(String[] args) {
    // Create a HashSet
    HashSet<String> hset = new HashSet<>();

    //add elements to HashSet
    hset.add("Chaitanya");
    hset.add("Rahul");
    hset.add("Tim");
    hset.add("Rick");
    hset.add("Harry");

    Iterator<String> it = hset.iterator();
    while(it.hasNext()){
      System.out.println(it.next());
    }
  }
}
```

LinkedHashSet in Java

- The LinkedHashSet class is the simplest implementation of the Set interface.
- The LinkedHashSet does not add any additional methods beyond those found in the Set interface.
- LinkedHashSet maintains the insertion order.
- Elements get sorted in the same sequence in which they have been added to the Set.
- The LinkedHashSet achieves good performance by using a hash to store the Object in the Set. The hash allows fast lookup.

LinkedHashSet in Java

```
3 import java.util.LinkedHashSet;
4 import java.util.Set;
5
6 public class LinkedHashSetDemo {
7
8     public static void main(String[] args) {
9
10        Set<String> veggiesLset=new LinkedHashSet<>();
11        veggiesLset.add("Cucumber");
12        veggiesLset.add("Pepper");
13        veggiesLset.add("Onion");
14        veggiesLset.add("Pumpkin");
15
16        System.out.println(veggiesLset);
```

Console

<terminated> LinkedHashSetDemo [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Dec 12, 2019, 5:53:47 PM)

```
[Cucumber, Pepper, Onion, Pumpkin]
```

TreeSet in Java

- The TreeSet class is the simplest implementation of the Set interface.
- The TreeSet does not add any additional methods beyond those found in the Set interface.
- TreeSet is similar to HashSet except that it sorts the elements in the ascending order while HashSet doesn’t maintain any order.
- TreeSet don’t allows null.

TreeSet in Java

```
 6 public class TreeSetDemo {
 7
 8     public static void main(String[] args) {
 9
10         Set<Integer> numbersTset = new TreeSet<>();
11         numbersTset.add(190);
12         numbersTset.add(10);
13         numbersTset.add(20);
14         numbersTset.add(200);
15
16         System.out.println(numbersTset);
17     }
18 }
```

Console

<terminated> TreeSetDemo [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Dec 12, 2019, 5:57:03 PM)

```
[10, 20, 190, 200]
```

Task

How can you remove all duplicates from ArrayList?

```
List<String> aList=new ArrayList<>();
aList.add("John");
aList.add("Jane");
aList.add("James");
aList.add("Jasmine");
aList.add("Jane");
aList.add("James");
```

Task

1. Create a Set collection in which you need to add names of the countries. In this set we want all objects to be sorted in alphabetical order. Using 2 different ways retrieve all elements from set.
2. Create a Set of cities in which you want to make sure that insertion order is maintained. Then remove any city that starts with “A”;
3. Create a Set collection that will hold Objects of Student Type. In this set we do not care about the insertion order. Each student object should have name and studentID. Display name of each student.