```
package ArrayList;

import java.util.ArrayList;
import java.util.Vector;

public class SynchronizationOverhead {
    public static void main(String[] args)  {
        Vector<Integer> vector = new Vector<>();
        ArrayList<Integer> arrayList = new ArrayList<>();

        long startTime,endTime;

        //Measuring Vector Performance
        startTime = System.nanoTime();
        for(int i = 0; i < 100000 ; i++){
            vector.add(i);
        }
        endTime = System.nanoTime();
        System.out.println("Time taken by Vector : "+
(endTime-startTime) + " ns");

System.out.println("==============================================
=========================");
        //Measuring the ArrayList Performance
        startTime=System.nanoTime();
        for(int i = 0; i < 100000 ; i++){
            arrayList.add(i);
        }
        endTime=System.nanoTime();
        System.out.println("Time taken by ArrayList : "+
(endTime-startTime) + " ns");
    }
}
```

O/P

```
Time taken by Vector: 14456100 ns

=================================================================

Time taken by ArrayList: 8491000 ns
```

- To make Arraylist Synchronized

```
List<Integer> synchronizedList = Collections.SynchronizedList(new ArrayList<>());
```

Set Interface:

- Sets store unique elements and do not allow duplicates.
- Two common implementations are HashSet and LinkedHashSet.

1. HashSet

- HashSet is a class that implements the set interface.
- It allows null values and stores elements in unordered way.

Characteristics:

- Does not maintain any order of elements.
- Best suited for quick adding, removing and contains operations.

```java
package Set;

import java.util.HashSet;

public class HashSetExample {
    public static void main(String[] args) {
        //Creating Hashset
        HashSet<String> fruits = new HashSet<>();

        //Adding elements to the hashset
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Orange");
        fruits.add("Grapes");

        //Displaying the HashSet
        System.out.println("HashSet :" + fruits);

        // Checking the element exist or not
        System.out.println("Does hashSet contains Grapes? : "
+fruits.contains("Grapes"));

        //Removing an element
        fruits.remove("Grapes");
        System.out.println("After removal HashSet :" +
fruits);

        //To check the size of the set
        System.out.println("Size of HashSet :" +
fruits.size());

        // To clear all the elements from the set
        fruits.clear();
        System.out.println("Is HashSet empty : "+
fruits.isEmpty());

    }
}
```

O/P:

```
HashSet :[Apple, Grapes, Orange, Banana]

Does hashSet contains Grapes? : true

After removal HashSet :[Apple, Orange, Banana]

Size of HashSet :3

Is HashSet empty : true
```

HashSet (Java SE 21 & JDK 21)

LinkedHashSet:

LinkedHasSet is similar to HashSet but maintains the doubly-linked list across all elements.

- Characteristics:

- Maintains the insertion order
- Slightly slower than HashSet because it maintains the order. (Load Factor*)

```
- import java.util.LinkedHashSet;

  public class LinkedHashSetExample {
      public static void main(String[] args) {
          // creating LinkedHashSet
          LinkedHashSet<String> cities = new
  LinkedHashSet<>();

          //Adding elements to the LinkedHashSet
          cities.add("Sambhajinagar");
          cities.add("Pune");
          cities.add("Nagpur");
          cities.add("Amravati");

          //Displaying the LHS
          System.out.println("LinkedHashSet : "+ cities);

          //To check LHS contain an element or not
          System.out.println("Does it contain Mumbai : "+
  cities.contains("Mumbai"));
      }
  }
```

LinkedHashSet (Java SE 21 & JDK 21)