Java CopyOnWriteArraySet class

Java CopyOnWriteArraySet is a thread-safe variant of HashSet which uses a underlying `CopyOnWriteArrayList` for all of its operations.

Similar to CopyOnWriteArrayList, it's immutable snapshot style iterator method uses a reference to the state of the array (inside the backing list) at the point that the iterator was created. This helps in usecases when traversal operations vastly outnumber set update operations and we do not want to synchronize the traversals and still want thread safety while updating the set.

1. CopyOnWriteArraySet Hierarchy

The `CopyOnWriteArraySet` class extends `AbstractSet` class and implements `Serializable` interface.

```
public class CopyOnWriteArraySet<E>
        extends AbstractSet<E>
        implements Serializable

{
    private final CopyOnWriteArrayList<E> al;

    //implementation
}
```

2. CopyOnWriteArraySet Features

The important things to learn about Java CopyOnWriteArraySet class are:

- As normal set data structure, it does not allow duplicates.

- CopyOnWriteArraySet class implement `Serializable` interface and extends `AbstractSet` class.
- Using CopyOnWriteArraySet is costly for update operations, bacause each mutation creates a cloned copy of underlying array and add/update element to it.
- It is thread-safe version of HashSet. Each thread accessing the set sees its own version of snapshot of backing array created while initializing the iterator for this set.
- Because it gets snapshot of underlying array while creating iterator, it does not throw ConcurrentModificationException.
- Mutation operations on iterators are not supported. These methods throw `UnsupportedOperationException`.
- CopyOnWriteArraySet is a concurrent replacement for a synchronized Set and offers better concurrency when iterations outnumber mutations.
- It allows duplicate elements and heterogeneous Objects (use generics to get compile time errors).
- Because it creates a new copy of underlying array everytime iterator is created, performance is slower than HashSet.

3. Java CopyOnWriteArraySet Example

Java program to show how iterators created at different times sees through snapshot version of set in CopyOnWriteArraySet. In given example, we first created list and itr1 when list had elements (1,2,3).

Then we added one more element to list and again created an iterator itr2.

Finally we verified the elements in both iterators.

```
CopyOnWriteArraySet<Integer> set = new CopyOnWriteArraySet<>(Arrays.asList(1,2,3));

System.out.println(set);  //[1, 2, 3]

//Get iterator 1
Iterator<Integer> itr1 = set.iterator();

//Add one element and verify set is updated
set.add(4);
System.out.println(set);  //[1, 2, 3, 4]

//Get iterator 2
Iterator<Integer> itr2 = set.iterator();

System.out.println("====Verify Iterator 1 content====");

itr1.forEachRemaining(System.out :: println);   //1,2,3

System.out.println("====Verify Iterator 2 content====");

itr2.forEachRemaining(System.out :: println);   //1,2,3,4
```

Program Output.

```
[1, 2, 3]
[1, 2, 3, 4]
====Verify Iterator 1 content====
1
2
3
====Verify Iterator 2 content====
1
2
3
4
```

4. CopyOnWriteArraySet Constructors

- CopyOnWriteArraySet() : Creates an empty set.

- CopyOnWriteArraySet(Collection c) : Creates a set containing the elements of the specified collection, in the order they are returned by the collection’s iterator.

Java CopyOnWriteArraySet Performance

Due to added step of creating a new backing array everytime the set is updated, it performs worse than HashSet.

There is no performance overhead on read operations and both classes perform same.