

 This class is thread-safe: All mutative operations (add, set, and remove) are atomic

 they either succeed completely, or they fail completely.

 Java Collections with Atomic Mutative Operations

 CopyOnWriteArrayList:

 All mutative operations are atomic.

 Suitable for scenarios with more reads than writes.

 ConcurrentHashMap:

 Supports atomic operations like put, remove, and replace.

 Optimized for concurrent access with high concurrency.

 ConcurrentLinkedQueue:

 Operations like offer, poll, and remove are atomic.

 Designed for concurrent access.

 ConcurrentSkipListMap:

 Operations such as put, remove, and replace are atomic.

 Sorted map suitable for concurrent access.

 ConcurrentSkipListSet:

 Operations such as add, remove, and replace are atomic.

 Sorted set suitable for concurrent access.

<https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/com/saravanjs/java22/console/collection/CopyOnWriteArrayListExample.java>

*public class* CopyOnWriteArrayListExample {
 *public static void* main(String[] args) {
 *List*<String> list = *new* CopyOnWriteArrayList<>();

 *// Adding elements* list.add("A");
 list.add("B");
 list.add("C");

 System.*out*.println("Initial list: " + list);

 *// Iterate over the list
 for* (String item : list) {
 System.*out*.println("Item: " + item);
 *// Modifying the list during iteration* list.add("D");
 }

 System.*out*.println("Final list: " + list);
 }
}

Initial list: [A, B, C]

Item: A

Item: B

Item: C

Final list: [A, B, C, D, D, D]

*public class* CopyOnWriteArrayListExample {

*https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/console/collection/CopyOnWriteArrayListExample2.java*

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

 *List*<String> copyOnWriteList = *new* CopyOnWriteArrayList<>();

 *Runnable* myTask = *new* Runnable() {
 *public void* run() {
 *for* (String name : copyOnWriteList) {
 System.*out*.println("Read: " + name);
 }
 }
 };

 copyOnWriteList.add("Alice");
 copyOnWriteList.add("Bob");
 copyOnWriteList.add("Charlie");

 *// Create a thread for reading* Thread readerThread = *new* Thread(myTask);

 *// Create a thread for writing* Thread writerThread = *new* Thread(() -> {
 copyOnWriteList.add("David");
 copyOnWriteList.remove("Alice");
 });

 readerThread.start();
 writerThread.start();

 Thread.*sleep*(1000);

 Thread readerThread2 = *new* Thread(myTask);
 readerThread2.start();

 }

}

Read: Alice

Read: Bob

Read: Charlie

...

Read: Bob

Read: Charlie

Read: David