If you create an array A of GameEntry objects and then clone it to create an array B, the behavior depends on how the cloning is performed.

**Understanding Cloning in Java**

* In Java, when you clone an array of objects using A.clone(), it creates a **shallow copy** of the array.
* This means that the new array B will have references to the same GameEntry objects as in A, rather than creating new GameEntry objects.

**Step-by-Step Breakdown**

1. Suppose A is an array of GameEntry objects.
2. B = A.clone(); creates a new array B, but the elements in B are still references to the same objects in A.
3. When you modify A[4].score = 550;, you are modifying the score field of the GameEntry object stored at index 4 in A.
4. Since B[4] refers to the **same** GameEntry object as A[4], the change is reflected in B[4] as well.

**Conclusion**

* The score value of the GameEntry object referenced by B[4] will also be **550**, because A[4] and B[4] refer to the same object in memory.

**Key Takeaway**

If you need a **deep copy** (where B contains new GameEntry objects rather than just references to the old ones), you must explicitly clone each GameEntry object inside a loop or implement a clone() method in GameEntry.