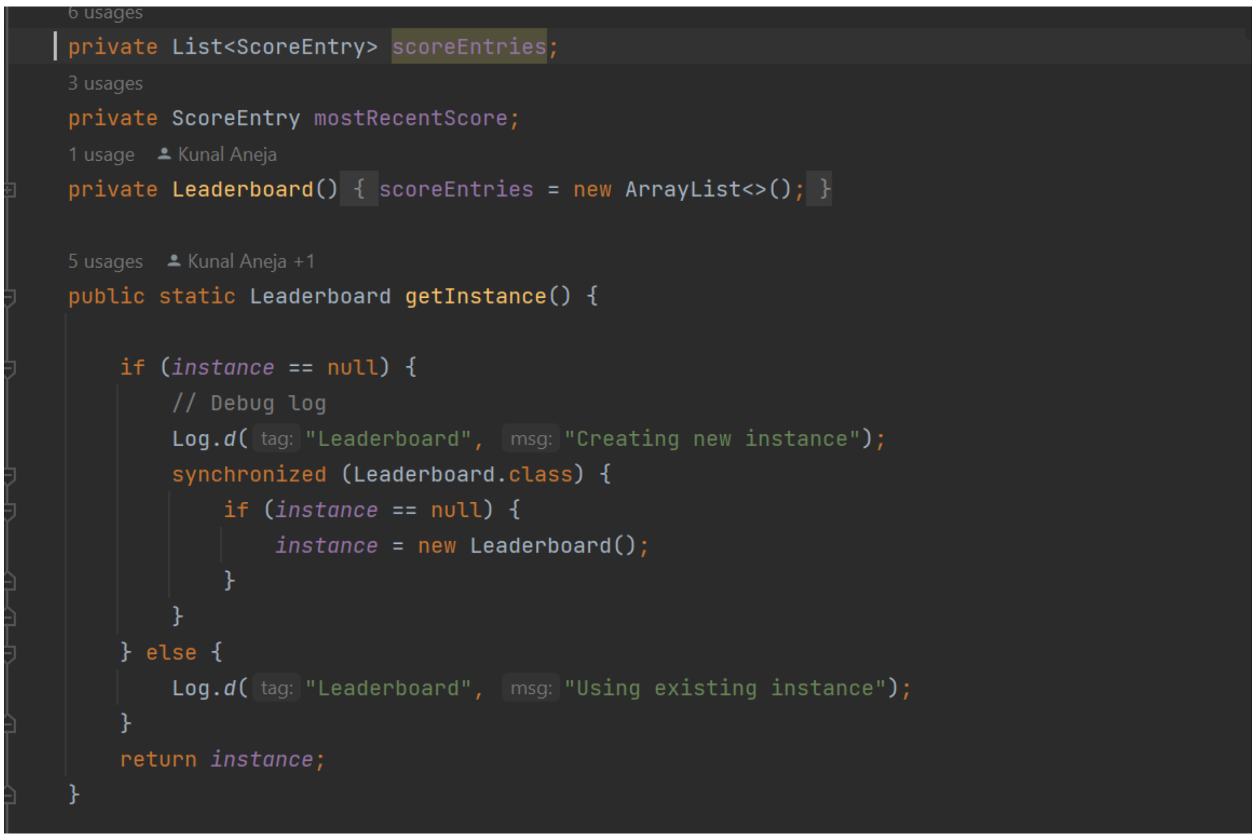
Design Pattern:



```
6 usages
private List<ScoreEntry> scoreEntries;
3 usages
private ScoreEntry mostRecentScore;
1 usage  Kunal Aneja
private Leaderboard() { scoreEntries = new ArrayList<>(); }

5 usages  Kunal Aneja +1
public static Leaderboard getInstance() {

    if (instance == null) {
        // Debug log
        Log.d( tag: "Leaderboard",  msg: "Creating new instance");
        synchronized (Leaderboard.class) {
            if (instance == null) {
                instance = new Leaderboard();
            }
        }
    } else {
        Log.d( tag: "Leaderboard",  msg: "Using existing instance");
    }
    return instance;
}
```

The leaderboard uses the singleton design pattern because it should not be instantiated more than once. In other words, there should not be more than one instance of the leaderboard (as multiple leaderboards don't make sense). To do this, we have a private constructor and a public static method called getInstance(), which will then return the one single object (or create it if it does not exist yet). We also added a logger to help with any potential debugging in the future.