題目：

用一個 score[][] 來儲存兩隊伍的成績 score[0] 表示隊伍A的分數; score[0][1] 表示第一局的分數，score[0][9] 表示第九局的分數，score[0][0] 表示總分數。score[1] 表示隊伍B的分數。

1. 如果第七局後（含七局），相差十分，則提前結束。若沒有相差十分，以九局結算，總得分高者獲勝，若平分則最多打到12局。若到12局未分勝負，則為平手。
2. 當上半場結束後已經確定後攻者（B）獲勝，則下半場不需要進行，score 儲存為 -1，需顯示時顯示為 X。
3. 資料有異常，應拋出例外，例外的訊息能說明適當的錯誤資訊：異常分數超出範圍、異常的提前結束、異常末局分數、超過局數、異常的未結束
4. 每局分數介於 1-99 分
5. 透過 log4j 來協助日後的 debug; log 會寫入 baseball.log 的檔案中
6. 宣告一個 exception，各種不同的例外用 message 來呈現即可

題目要求

1. case7, case8 錯誤，不該有多餘的局數，但程式並沒有反應該異常
2. 缺陷：displayScore() 固定都是顯示12局，沒有進行的局數以 0 表示，造成誤解（以為有進行，該局0分）
3. 缺陷：case5 有檢測出異常，卻仍然公告勝負
4. 請加上 exception 及 log 來提升程式的強健性。目前程式用 System.out.println() 來印出錯誤或提示訊息，請改為 exception 或(與) log
5. 加上 PMD 來檢驗此程式的靜態品質，請產出檢視報告
6. 改以 JUnit 進行測試

程式、執行畫面及其說明：

1. 修正case7, case8 錯誤，不該有多餘的局數，但程式並沒有反應該異常

```
===== 測試案例 7. 九局已分勝負仍有多餘局數（預期 GameException） =====
[16:47:51] INFO  BaseballScoreChecker - 開始執行案例 #7：7. 九局已分勝負仍有多餘局數（預期 GameException）
[16:47:51] INFO  BaseballScoreChecker - --- 開始處理新的比分表 ---
[16:47:51] INFO  BaseballScoreChecker - 九局結束分出勝負。A:4 B:5
[16:47:51] WARN  BaseballScoreChecker - 超過局數：比賽於第 9 局結束，但輸入仍含後續局資料。
[16:47:51] ERROR BaseballScoreChecker - 【例外】7. 九局已分勝負仍有多餘局數（預期 GameException）: 超過局數：比賽於第 9 局結束，但輸入仍含後續局資料。
【例外】超過局數：比賽於第 9 局結束，但輸入仍含後續局資料。

===== 測試案例 8. 11 局已分勝負仍有多餘局數（預期 GameException） =====
[16:47:51] INFO  BaseballScoreChecker - 開始執行案例 #8：8. 11 局已分勝負仍有多餘局數（預期 GameException）
[16:47:51] INFO  BaseballScoreChecker - --- 開始處理新的比分表 ---
[16:47:51] INFO  BaseballScoreChecker - 九局結束分出勝負。A:4 B:5
[16:47:51] WARN  BaseballScoreChecker - 超過局數：比賽已於第 9 局結束，但第 11 局仍有分數。
[16:47:51] ERROR BaseballScoreChecker - 【例外】8. 11 局已分勝負仍有多餘局數（預期 GameException）: 超過局數：比賽已於第 9 局結束，但第 11 局仍有分數。
【例外】超過局數：比賽已於第 9 局結束，但第 11 局仍有分數。
```

```java
// 結束後仍有多餘局數
int logicalLen = endedInning + 1;
if (score[0].length > logicalLen || score[1].length > logicalLen) {
    String msg = "超過局數：比賽於第 " + endedInning + " 局結束，但輸入仍含後續局資料。";
    log.warn(msg);
    throw new GameException(msg);
}
```

2. 修正沒有進行的局數以 0 表示，造成誤解
   （改成用 X 表示）

```
===== 測試案例 2. 7 局提前結束 B 獲勝（12-2, B下半X） =====
[16:47:51] INFO  BaseballScoreChecker - 開始執行案例 #2：2. 7 局提前結束 B 獲勝（12-2, B下半X）
[16:47:51] INFO  BaseballScoreChecker - --- 開始處理新的比分表 ---
[16:47:51] INFO  BaseballScoreChecker - 比賽於第 7 局上半後結束，B 無需下半（X）。A:2 B:12
[16:47:51] INFO  BaseballScoreChecker - 比分處理完成。第 7 局結束。結果：隊伍B獲勝（12-2）

-------------------------------------------------
|   隊伍   | 總分 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 狀態 |
-------------------------------------------------
| A（先攻） |   2 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |  |
| B（後攻） |  12 | 2 | 5 | 0 | 1 | 0 | 4 | X | 獲勝 |
-------------------------------------------------
```

```java
for (int t = 0; t < 2; t++) {
    System.out.printf("| %s |%4d |", teamNames[t], score[t][0]);
    for (int i = 1; i <= end; i++) {
        int s = score[t][i];
        if (s == -1) System.out.print(" X |");
        else System.out.printf("%2d |", s);
    }
```

3. 修正 case5 有檢測出異常，卻仍然公告勝負

```
===== 測試案例 5. 異常分數 100 (預期 GameException) =====
[16:47:51] INFO  BaseballScoreChecker - 開始執行案例 #5：5. 異常分數 100 (預期 GameException)
[16:47:51] INFO  BaseballScoreChecker - --- 開始處理新的比分表 ---
[16:47:51] WARN  BaseballScoreChecker - 異常分數超出範圍：A 第 1 局得分必須介於 0..99。
[16:47:51] ERROR BaseballScoreChecker - 【例外】5. 異常分數 100 (預期 GameException): 異常分數超出範圍：A 第 1 局得分必須介於 0..99。
【例外】異常分數超出範圍：A 第 1 局得分必須介於 0..99。
```

```java
// 檢查分數合理性
if (a < 0 || a > 99) {
    String msg = "異常分數超出範圍：A 第 " + i + " 局得分必須介於 0..99。";
    log.warn(msg);
    throw new GameException(msg);
}
if ((b < 0 || b > 99) && b != -1) {
    String msg = "異常分數超出範圍：B 第 " + i + " 局得分必須介於 0..99 或為 -1 (未打下半)。";
    log.warn(msg);
    throw new GameException(msg);
}
```

4. 加上 exception 及 log 來提升程式的強健性。

（挑一段作為範例）

```
===== 測試案例 4. 9 局平手 (預期拋出 GameException) =====
[16:57:09] INFO  BaseballScoreChecker - 開始執行案例 #4：4. 9 局平手 (預期拋出 GameException)
[16:57:09] INFO  BaseballScoreChecker - --- 開始處理新的比分表 ---
[16:57:09] WARN  BaseballScoreChecker - 異常的未結束：第 9 局結束平手，應繼續進行至第 12 局或分出勝負。
[16:57:09] ERROR BaseballScoreChecker - 【例外】4. 9 局平手 (預期拋出 GameException): 異常的未結束：第 9 局結束平手，應繼續進行至第 12 局或分出勝負。
【例外】異常的未結束：第 9 局結束平手，應繼續進行至第 12 局或分出勝負。
```

```java
// 檢查未結束狀況
if (!gameEnded) {
    if (maxProvided < 9) {
        String msg = "異常的未結束：資料只到第 " + maxProvided + " 局，不足九局。";
        log.warn(msg);
        throw new GameException(msg);
    }
    if (teamATotal == teamBTotal) {
        if (maxProvided < MAX_INNINGS) {
            String msg = "異常的未結束：第 " + maxProvided + " 局結束平手，應繼續進行至第 12 局或分出勝負。";
            log.warn(msg);
            throw new GameException(msg);
        } else {
            endedInning = MAX_INNINGS;
            gameEnded = true;
            log.info( message: "十二局結束為平手。A:{} B:{}", teamATotal, teamBTotal);
        }
    } else {
        endedInning = maxProvided;
        gameEnded = true;
        log.info( message: "於第 {} 局結束並分出勝負。A:{} B:{}", endedInning, teamATotal, teamBTotal);
    }
}
```

5. 加上 PMD 來檢驗此程式的靜態品質，請產出檢視報告

PMD Results

The following document contains the results of PMD 6.55.0.

Violations By Priority

Priority 1

org/example/BaseballScoreChecker.java

Rule	Violation	Line
FieldNamingConventions	The constant name 'log' doesn't match '[A-Z][A-Z_0-9]*'	7
VariableNamingConventions	Variables that are final and static should be all capitals, 'log' is not all capitals.	7

6. 改以 JUnit 進行測試

```java
@Test
void normalNineInnings_Awins() {
    int[][] case1 = {
            {0, 2, 1, 0, 3, 2, 1, 3, 3, 0},
            {0, 1, 0, 2, 1, 0, 3, 1, 2, 0}
    };
    GameResult r = BaseballScoreChecker.checkScore(case1);
    assertEquals( expected: 9, r.getEndedInning());
    assertEquals( expected: "隊伍A獲勝 (15-10)", r.getOutcome());
}

@Test
void mercyAtSeven_Bwins_noBottom() {
    int[][] case2 = {
            {0, 0, 1, 0, 0, 1, 0, 0},
            {0, 2, 5, 0, 1, 0, 4, -1}
    };
    GameResult r = BaseballScoreChecker.checkScore(case2);
    assertTrue(r.isEndedEarly());
    assertEquals( expected: 7, r.getEndedInning());
    assertTrue( condition: r.getScores()[1][7] == -1); // X
}

@Test
void tieAfterNine_mustExtend_throw() {
    int[][] case4 = {
            {0, 1, 0, 1, 0, 0, 0, 1, 0, 0},
            {0, 0, 1, 0, 1, 0, 1, 0, 0, 0}
    };
    assertThrows(GameException.class, () -> BaseballScoreChecker.checkScore(case4));
}
```

```java
@Test
void abnormalScore_100_throws_and_noOutcomePrinted() {
    int[][] case5 = {
            {0, 100, 1, 0, 0, 0, 0, 0, 0, 0},
            {0, 0, 1, 0, 1, 0, 1, 0, 0, 0}
    };
    assertThrows(GameException.class, () -> BaseballScoreChecker.checkScore(case5));
}

@Test
void extraInningsProvidedAfterDecided_throws_case7() {
    int[][] case7 = {
            {0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0},
            {0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0}
    };
    assertThrows(GameException.class, () -> BaseballScoreChecker.checkScore(case7));
}

@Test
void extraInningsProvidedAfterDecided_throws_case8() {
    int[][] case8 = {
            {0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0},
            {0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0}
    };
    assertThrows(GameException.class, () -> BaseballScoreChecker.checkScore(case8));
}
```

參考資料與使用工具及比例（包含 AI）：

上課講義、AI(大約 70%)