Software Testing - HW0x

Coverage Testing

資訊三丙
D1009331 吳克廷

目錄

目錄 2
前言 3
Baseball - 設計 4
Baseball - Coverage Test 6
心得 13

前言

本文件為解釋程式碼的運作原理，因此會擷取片段的程式碼，並非完整的程式碼。若想參閱完整的程式架構，可以根據以下所提供的網址，前往GitHub進行參閱，老師謝謝。

- **Baseball.java**: https://github.com/alecwu44743/Software-Testing/blob/master/HW0x/src/main/java/org/example/baseball.java

- **BMITest.java**: https://github.com/alecwu44743/Software-Testing/blob/master/HW0x/src/test/java/org/example/baseballTest.java

Baseball - 設計

本題目的目標為計算棒球得分，並當使用者提供錯誤的資料時，必須給予相對應的exception，一開始整個系統的入口為scrore()這個function，會帶入兩對inning和player的資料，並且先做驗證，確保資料正確後，才開始做得分的計算。

```java
// calculate the score of the game
public int score(int inningA[], int inningB[], int playerA[], int playerB[]) throws Exception {
    validateArray(inningA, inningB, playerA, playerB); // validate the input arrays

    // return the score of the game
    // positive number means playerA wins
    // negative number means playerB wins
    return calculateArray(inningA) - calculateArray(inningB);
}
```

而這裡也設計了一個function用於做計算，calculateArray()

```java
public int calculateArray(int input[]){ // calculate total score of an array
    int sum = 0;
    for (int i = 0; i < input.length; i++) {
        if(input[i] == -1){
            continue;
        }
        sum += input[i];
    }
    return sum;
}
```

於下個部分，會開始討論個別exception的部分

一開始先將局數小於九局、局數不一致、總分不一致、不可以和局基本的exception先做處理

```java
if(inningA.length < 9 || inningB.length < 9) { // inning length shouldn't be less than 9
    throw new Exception("局數小於九局");
}

if(inningA.length != inningB.length) { // inning length should be equal
    throw new Exception("局數不一致");
}

// total score of inning should be equal to total score of player
if((calculateArray(inningA) != calculateArray(playerA))
    || (calculateArray(inningB) != calculateArray(playerB))) {
    throw new Exception("總分不一致");
}

if(calculateArray(inningA) == calculateArray(inningB)) { // total score of inning should not be equal
    throw new Exception("不可以和局");
}
```

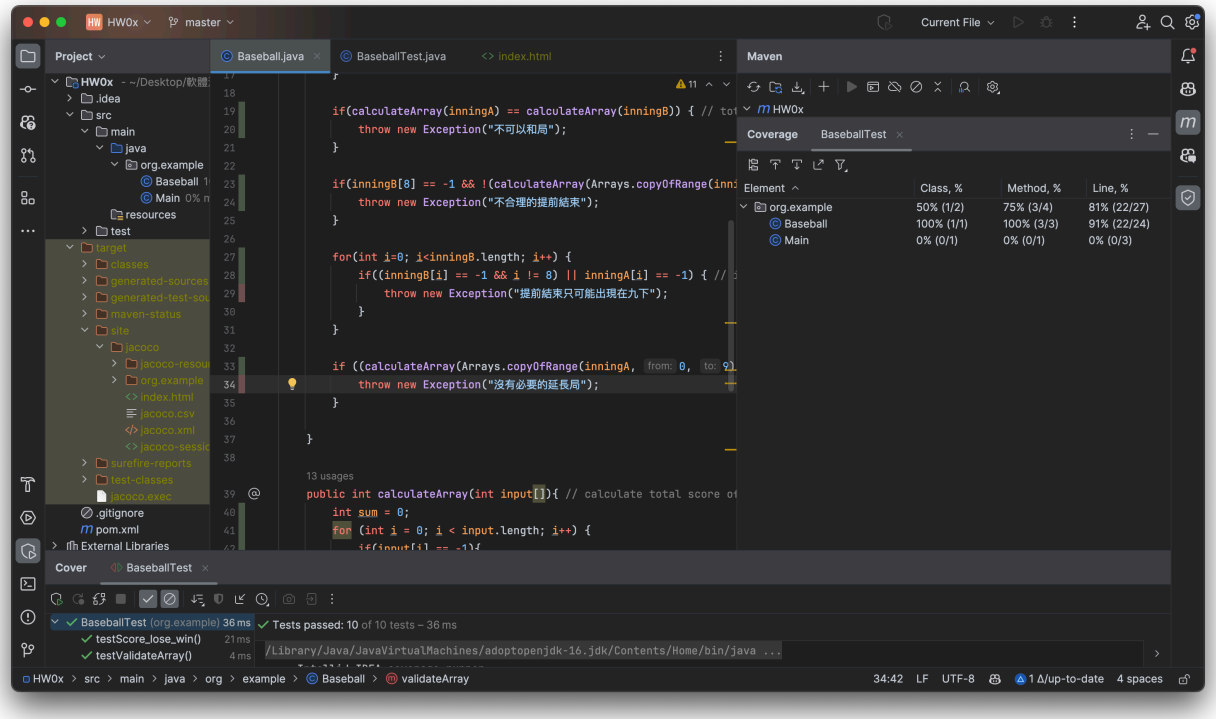
接著再根據和局、延長賽、提前結束時做exception的handling

```java
// if inningB[8] is -1,
// the total score of the first 8 innings of inningB should be
// greater than the total score of the first 9 innings of inningA
if(inningB[8] == -1
    && !(calculateArray(Arrays.copyOfRange(inningA, 0, 9)) < calculateArray(Arrays.copyOfRange(inningB, 0, 8)))) {
    throw new Exception("不合理的提前結束");
}

for(int i=0; i<inningB.length; i++) {
    if((inningB[i] == -1 && i != 8) || inningA[i] == -1) { // inningB[8] should be the only -1 in inningB
        throw new Exception("提前結束只可能出現在九下");
    }
}

if ((calculateArray(Arrays.copyOfRange(inningA, 0, 9)) != calculateArray(Arrays.copyOfRange(inningB, 0, 9)))
        && (inningA.length > 9 || inningB.length > 9)) { // should not have extra innings
    throw new Exception("沒有必要的延長局");
}
```

這裡我們依據老師在Google Docs上的內容，再將邏輯轉換成程式，並做出正確的exception處理

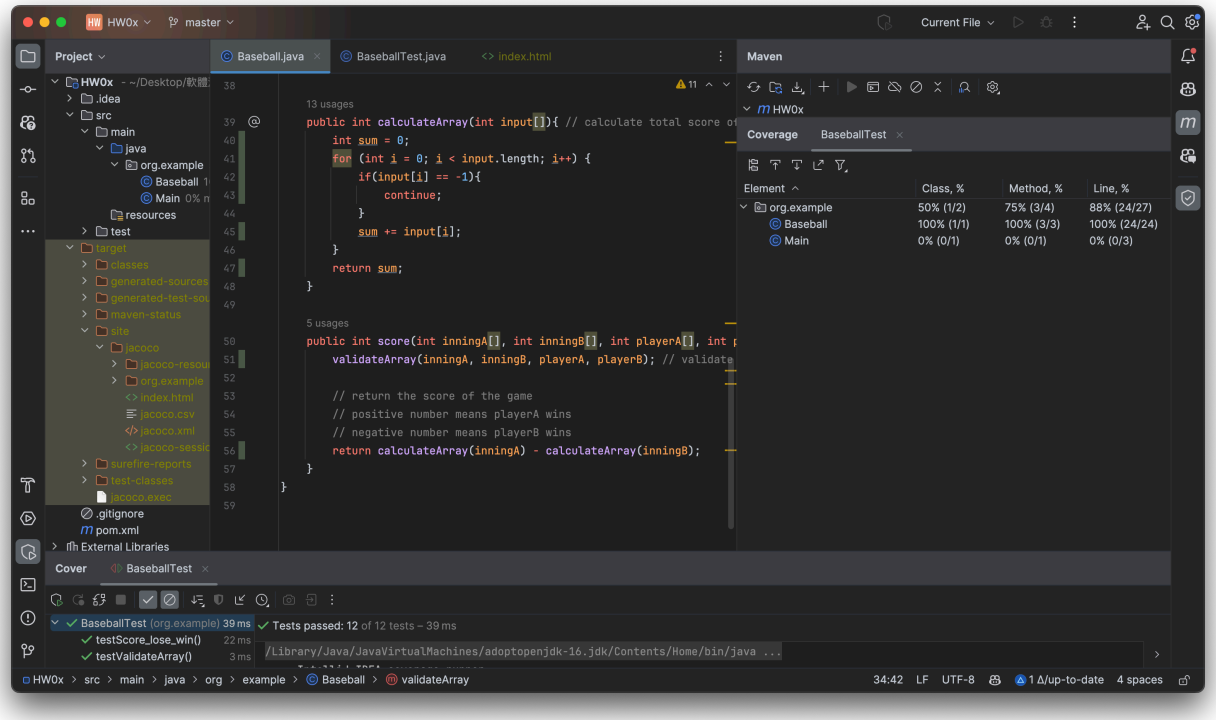
Baseball - Coverage Test

- 測試結果/第一次



這裡可看出來只有91%，還有些程式碼沒有被測試到，包括了提前結束只在九局下、沒必要延長局，因此這裡再修改一下程式碼

- 測試結果/**最後結果**



最後變更了測試資料以及選項後，就覆蓋到了100%，下個階段會介紹這裡的測試方式

- 測試介紹

```java
@Test
public void testValidateArray() throws Exception {
    Baseball game = new Baseball();

    // Add test cases to achieve high coverage
    // ...

    // Example: Test for "局數小於九局" exception
    int[] inningA = {1, 2, 3};
    int[] inningB = {3, 4, 5};
    int[] playerA = {1, 2, 3};
    int[] playerB = {3, 4, 5};

    assertThrows(Exception.class, () -> game.validateArray(inningA, inningB, playerA, playerB));
}

@Test
public void testCalculateArray() {
    Baseball game = new Baseball();

    // Add test cases to achieve high coverage
    // ...

    // Example: Test for calculateArray()
    int[] input = {1, 2, 3};
    int expected = 6;
    int actual = game.calculateArray(input);
    assertEquals(expected, actual);
}
```

這裡一開始先對基本的方向做測試，像是基本的小於九局和測試計算陣列的function是否算出來是對的

```java
// for score()
@Test // Example: Test for "局數小於九局" exception
public void testScore_1() throws Exception {
    Baseball game = new Baseball();

    int[] inningA = {1,1,1,1,1,1,1,1,1};
    int[] inningB = {1,1,1,1,1,1,1,1,2};
    int[] playerA = {2,0,1,1,1,1,0,2};
    int[] playerB = {1,1,3,0,0,1,1,2};

    assertThrows(Exception.class, () -> game.score(inningA, inningB, playerA, playerB));
}

@Test // Example: Test for "局數不一致" exception
public void testScore_2() throws Exception {
    Baseball game = new Baseball();

    int[] inningA = {1,1,1,1,1,1,1,1,1,0};
    int[] inningB = {1,1,1,1,1,1,1,2,0,1,1};
    int[] playerA = {2,0,1,1,1,1,0,2,0};
    int[] playerB = {1,1,3,0,0,1,1,2,0};

    assertThrows(Exception.class, () -> game.score(inningA, inningB, playerA, playerB));
}
```

接著，這裡著重在測試其他小於九局的案例以及局數不一致的問題

```java
@Test // Example: Test for "總分不一致" exception
public void testScore_3() throws Exception {
    Baseball game = new Baseball();

    int[] inningA = {1,1,1,1,1,1,1,1,0};
    int[] inningB = {1,1,1,1,1,1,1,2,0};
    int[] playerA = {2,0,1,1,1,1,0,3,0};
    int[] playerB = {1,1,3,0,0,1,1,8,1};

    assertThrows(Exception.class, () -> game.score(inningA, inningB, playerA, playerB));
}

@Test // Example: Test for "不可以和局" exception
public void testScore_4() throws Exception {
    Baseball game = new Baseball();

    int[] inningA = {1,1,1,1,1,1,1,1,0};
    int[] inningB = {1,1,1,1,1,1,1,1,0};
    int[] playerA = {2,0,1,1,1,1,0,2,0};
    int[] playerB = {1,1,1,1,1,1,1,1,0};

    assertThrows(Exception.class, () -> game.score(inningA, inningB, playerA, playerB));
}
```

這裡主要處理總分不一致和不可以和局的問題

```java
@Test // Example: Test for "不合理的提前結束" exception
public void testScore_5() throws Exception {
    Baseball game = new Baseball();

    // Test case where inningA[7] is -1, and the total score of the first 7 innings is less than or equal to the total score of the first 8 innings of inningB
    int[] inningA = {1, 1, 1, 1, 1, 1, 1, 1, 1};
    int[] inningB = {1, 1, 1, 1, 1, 1, 1, -1, 0};
    int[] playerA = {1, 1, 1, 1, 1, 1, 1, 1};
    int[] playerB = {1, 1, 1, 1, 1, 1, 1, 1};

    assertThrows(Exception.class, () -> game.validateArray(inningA, inningB, playerA, playerB));
}

@Test // Example: Test for "不合理的提前結束" exception
public void testScore_6() throws Exception {
    Baseball game = new Baseball();

    // Test case where inningA[7] is -1, and the total score of the first 7 innings is greater than the total score of the first 8 innings of inningB
    int[] inningA = {1, 1, 1, 1, 1, 1, 1, 1, 1};
    int[] inningB = {1, 1, 1, 1, 1, 1, 1, -1}; // 注意最後一局是提前結束的標記
    int[] playerA = {1, 1, 1, 1, 1, 1, 1, 1};
    int[] playerB = {1, 1, 1, 1, 1, 1, 1, 1};

    assertThrows(Exception.class, () -> game.validateArray(inningA, inningB, playerA, playerB));
}

@Test // Example: Test for "不合理的提前結束" exception
public void testScore_7() throws Exception {
    Baseball game = new Baseball();

    int[] inningA = {1, 1, 1, 1, 1, 1, 1, 1, 1};
    int[] inningB = {1, 1, 1, 1, 1, 1, 1, 1, -1}; // 注意最後一局是提前結束的標記
    int[] playerA = {1, 1, 1, 1, 1, 1, 1, 1, 1};
    int[] playerB = {1, 1, 1, 1, 1, 1, 1, 1, 0};

    assertThrows(Exception.class, () -> game.validateArray(inningA, inningB, playerA, playerB));
}
```

這裡主要在處理多個不合理提前結束的問題

這裡主要處理沒有必要延長的問題以及九下出現的問題

```java
@Test // Example: Test for "沒有必要的延長局" exception
public void testScore_8() throws Exception {
    Baseball game = new Baseball();

    // Test case where inningA[7] is -1, inningB[7] is 'X', and the total score of the first 7 innings of inningA is equal to the total score of the first 7 innings of inningB
    int[] inningA = {1, 1, 1, 1, 1, 1, 1, 2, 3, 1};
    int[] inningB = {1, 1, 1, 1, 1, 1, 1, 1, 2, 1};
    int[] playerA = {1, 1, 1, 1, 1, 1, 1, 2, 3, 1};
    int[] playerB = {1, 1, 1, 1, 1, 1, 1, 1, 2, 1};

    assertThrows(Exception.class, () -> game.validateArray(inningA, inningB, playerA, playerB));
}

@Test // Example: Test for "提前結束只可能出現在九下" exception
public void testScore_9() throws Exception {
    Baseball game = new Baseball();

    // Test case where inningA[7] is -1, and inningB[6] is -1
    int[] inningA = {1, 1, 1, 1, 1, 1, 1, 0, 0};
    int[] inningB = {1, 1, 1, 1, 1, 1, -1, 0, 0};
    int[] playerA = {0, 1, 1, 1, 1, 1, 1, 1};
    int[] playerB = {1, 1, 1, 1, 1, 1, 0, 0};

    assertThrows(Exception.class, () -> game.validateArray(inningA, inningB, playerA, playerB));
}
```

最後再對無問題的資料做輸贏計算

```java
    @Test
    public void testScore_lose_win() throws Exception {
        Baseball game = new Baseball();


        // Example: Test for "局數小於九局" exception
        int[] inningA = {1,1,1,1,1,1,1,1,0};
        int[] inningB = {1,1,1,1,1,1,1,2,0};
        int[] playerA = {2,0,1,1,1,1,0,2,0};
        int[] playerB = {1,1,3,0,0,1,1,2,0};

//        assertThrows(Exception.class, () -> game.score(inningA, inningB, playerA, playerB));
        assertEquals(-1, game.score(inningA, inningB, playerA, playerB));
    }
```

心得

這次的軟體測試作業加分題，從期中考的題目做修改後並對該程式做coverage test，一開始在期中考寫這題時，由於不懂棒球，花了大量的時間搞懂棒球的規則，搞懂後也至剩下十分鐘左右，後來在寫回家作業後，可以快速理解題目意思，並且設計出該程式。

這次的測試方式為coverage test，在測試時有別於過去的方式，要用看程式碼的方式去測試該程式碼，並確保每行程式碼都有測試到，雖然在一開始的覆蓋率就高達九成，但後期的一成也花了不少許多想，最後也很幸運的達成100%。

最後，也感謝老師給我們加分的機會，來彌補期中考的分數，也感謝老師教導我們在coverage test上的技巧，也使我們受益良多。

November 20, 2023