Mission 7.1 GradeDisplayer

D0745378 薛竣祐

GradeDisplayer 的開發重點在正確的呼叫 GradeCollecter 跟 GradeSorter，並提供相關的參數，並確定接收到的回傳結果正確的。測試時可以先以 stub 設計 GradeCollecter 跟 GradeSorter 回傳假結果，讓 GradeDisplayer 先以該假結果正常執行，並確認該做的事情無誤，例如印出成績等等。

GradeCollecter 的開發重點在需要區呼叫 GradeDB 並將結果透過 AverageComputer 算出各科平均。可以 mock 來建立一個假的 GradeDB，並將回傳結果寫死，傳進 AverageComputer，可以先預設一個答案，並使用 Assertion 來驗證 AverageComputer 的結果是否正常。若 AverageComputer 還不可用，也可以透過一樣方式設一個假結果，並將對應的內容回傳至上層。

當 GradeSorter 和 GradeCollecter 完成時，GradeDisplayer 可以將之前的 stub 設定部分註解掉，直接使用真實函式，並檢驗結果是否正常。

App

```
package GradeDisplayer;
import java.util.Arrays;

public class App {
    public static void main(String[] args) {
        App app = new App();
        app.gradeDisplayer(1);
    }
    //main app
    void gradeDisplayer(int classID){
        int[] grade = gradeCollecter(classID);
        grade = gradeSorter(grade);
        System.out.println(Arrays.toString(grade));
    }
    //return grade[]
    int[] gradeCollecter(int classID){
        int[][] subjectGrade = gradeDB(classID);
        return averageComputer(subjectGrade);
    }
    //return grade[]
    int[] averageComputer(int[][] subjectGrade){
        int peopleCount = subjectGrade.length;
        int[] grade = new int[peopleCount];
        for(int i=0;i<peopleCount;i++){
            for(int j=0;j<subjectGrade[i].length;j++){
                grade[i] += subjectGrade[i][j];
            }
            grade[i]/=subjectGrade[i].length;
        }
        return grade;
```

```
    }

    //return subjectGrade[][]
    int[][] gradeDB(int classID){
        if (classID==1){
            return new int[][]{
                    {10,20,30,40},
                    {60,60,60,60},
                    {90,100,80,70}
            };
        }
        if (classID==2){
            return new int[][]{
                    {20,60,80,90},
                    {30,40,50,30},
                    {12,23,34,45}
            };
        }
        return null;
    }
    //return a sorted grade[]
    int[] gradeSorter(int[] grade){
        for(int i=0;i<grade.length;i++){
            for(int j=i+1;j<grade.length;j++){
                if(grade[i]>grade[j]){
                    grade = dataSwaper(grade,i,j);
                }
            }
        }
        return grade;
    }
    //swap i,j int data[]
    int[] dataSwaper(int[] data, int i, int j){
        int tmp = data[i];
        data[i] = data[j];
        data[j] = tmp;
        return data;
    }
}
```

AppTest

```
/*
 * This Java source file was generated by the Gradle 'init' task.
 */
package GradeDisplayer;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.util.Arrays;

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

class AppTest {
    App app;
    @BeforeEach
    public void setUp(){
        app = spy(App.class);
    }

    @Test
```

```
    public void averageComputerTest() {
        //stub
        when(app.gradeDB(10))
                .thenReturn(new int[][]{{10,10,10},{20,20,20}});
        int[][] subject = app.gradeDB(10);
        int[] exp_grade = app.averageComputer(subject);
        int[] real_grade = {10,20};
        assertArrayEquals(exp_grade, real_grade);
    }
    @Test
    void gradeSorterTest(){
        int[][] useless = new int[][]{{1,2,3}};
        when(app.averageComputer(useless))
                .thenReturn(new int[]{30,20,10});
        int[] grade = app.averageComputer(useless);
        int[] exp_grade = app.gradeSorter(grade);
        int[] real_grade = {10,20,30};
        assertArrayEquals(exp_grade,real_grade);
    }
    @Test
    void gradeCollecterTest(){
        when(app.gradeDB(3))
                .thenReturn(new
int[][]{{10,20,30},{60,70,80},{100,90,80}});
        int[] exp_grade = app.gradeCollecter(3);
        System.out.println(Arrays.toString(exp_grade));
        int[] real_grade = {20,70,90};
        assertArrayEquals(exp_grade,real_grade);
    }

}
```

過程：

先將 func 的各參數與回傳設定好，由最上層開始撰寫，先以 spy 方式預設下層的結果，並嘗試呼叫，確認印出結果無誤時，往下層移動，重複步驟，也是以 spy 預設下層結果，逐漸嘗試。直到最下層時，可以先寫一份假程式，例如 DB 可以直接回傳假資料，並回推上層是否能正確呼叫，或是實際完成該程式，像是 Swap 可以實作真實程式，並往上推，去掉 stub 後測試是否正常，並逐漸回到最上層，完成整體程式。