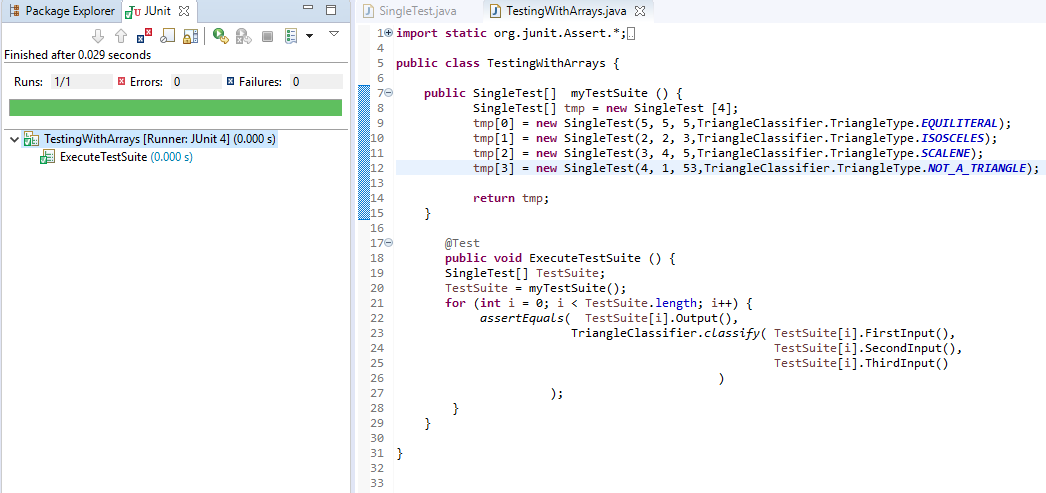
# Task 4.1



The pros of implementing the tests this way is that is is quicker and easier to add new tests as they are all collected in one place.

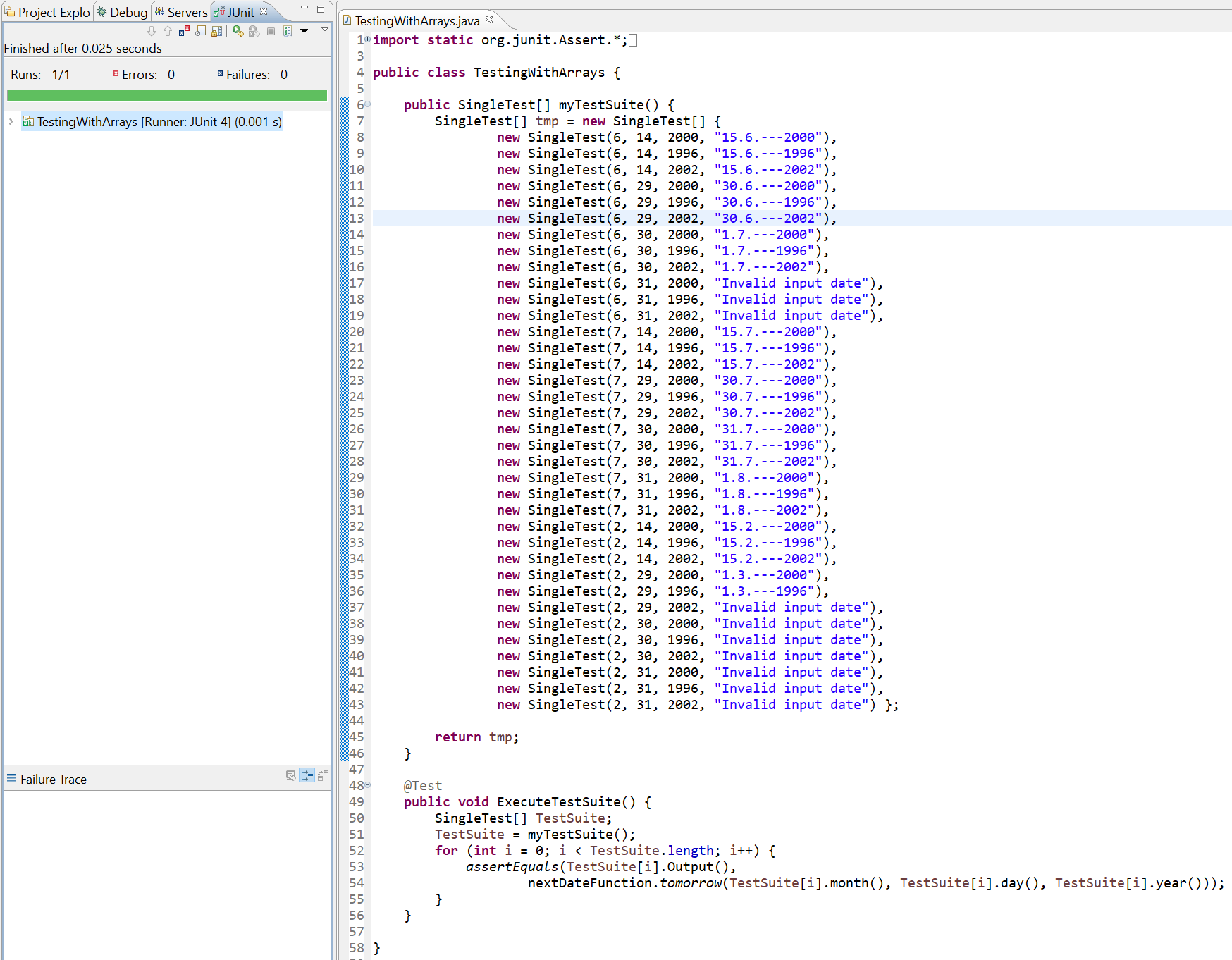
The main con is that the tests will not run individually and show individual results for each test you add to the array, rather one overall result for the entire array of tests. Doing it this way will make it hard to know which test has failed.

# Task 4.2

After the testing phase has been completed it is required to prove to the client that the testing has been completed and the tests have passed. To do this we can provide three things. Firstly, evidence that the tests have been carried out and that they passed. Next, we need to give the client the actual results of the tests. These results will be recorded in the log-files that can be easily shown to the client. Lastly, some sort of documentation and analysis of the tests should be given. From the tests that are carried out on the software an easy to read HTML document can be generated to show the test coverage of the program. This will indicate that each class and method has been tested as extensively as required.

All of this should be enough evidence of the testing that was carried out, however the client could still have doubts as these results could have been forged. This is a hard thing to prove as the client is likely to has less technological understanding. One way to solve this would be to show the client the software being tested live so they can see each test being carried out and passing.

Below is a screenshot showing that the 36 tests and that they all passed.



## Corrected Code – nextDateFunction.java

**import** java.util.Scanner;

**public** **class** nextDateFunction {

**public** **static** **boolean** monthIsInRange(**int** month) {

**return** ((1 <= month) && month <= 12);

}

**public** **static** **boolean** dayIsInRange(**int** day) {

**return** ((1 <= day) && day <= 31);

}

**public** **static** **boolean** yearIsInRange(**int** year) {

**return** ((1812 <= year) && year <= 2012);

}

**public** **static** **boolean** hasThirtyOneDaysAndIsNotDecember(**int** month) {

**return** (month == 1) || (month == 3) || (month == 5) || (month == 7) || (month == 10);

}

**public** **static** **boolean** hasThirtyDays(**int** month) {

**return** (month == 4) || (month == 6) || (month == 9) || (month == 11);

}

**public** **static** **boolean** isFebruary(**int** month) {

**return** (month == 2);

}

**public** **static** **boolean** isDecember(**int** month) {

**return** (month == 12);

}

**public** **static** **boolean** isCenturyYear(**int** year) {

**return** (year % 100 == 0);

}

**public** **static** **boolean** isLeapYear(**int** year) {

**if** (*isCenturyYear*(year)) {

**return** (year % 400) == 0;

} **else** {

**return** (year % 4) == 0;

}

}

// works for correct inputs only

**public** **static** String computeTomorrow(**int** day, **int** month, **int** year) {

String s = "";

**int** tomorrowDay;

**int** tomorrowMonth;

**int** tomorrowYear;

**if** (*hasThirtyOneDaysAndIsNotDecember*(month)) {

**if** (day < 31) {

tomorrowDay = day + 1;

tomorrowMonth = month;

tomorrowYear = year;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

} **else** {

tomorrowDay = 1;

tomorrowMonth = month + 1;

tomorrowYear = year;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

}

} **else** **if** (*hasThirtyDays*(month)) {

**if** (day < 30) {

tomorrowDay = day + 1;

tomorrowMonth = month;

tomorrowYear = year;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

} **else** **if** (day > 30) {

s = s + "Invalid input date";

} **else** {

tomorrowDay = 1;

tomorrowMonth = month + 1;

tomorrowYear = year;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

}

} **else** **if** (*isDecember*(month)) {

**if** (day < 31) {

tomorrowDay = day + 1;

tomorrowMonth = month;

tomorrowYear = year;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

} **else** {

**if** (year == 2012) {

s = "Next day out of range";

} **else** {

tomorrowDay = 1;

tomorrowMonth = 1;

tomorrowYear = year + 1;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

}

}

} **else** **if** (*isFebruary*(month)) {

**if** (day <= 27) {

tomorrowDay = day + 1;

tomorrowMonth = month;

tomorrowYear = year;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

} **else** **if** (day == 28) {

**if** (*isLeapYear*(year)) {

tomorrowDay = day + 1;

tomorrowMonth = month;

tomorrowYear = year;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

} **else** { // not a leap year

tomorrowDay = 1;

tomorrowMonth = 3;

tomorrowYear = year;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

}

} **else** **if** (day == 29) {

**if** (*isLeapYear*(year)) {

tomorrowDay = 1;

tomorrowMonth = 3;

tomorrowYear = year;

s = s + tomorrowDay + "." + tomorrowMonth + ".---" + tomorrowYear;

} **else** { // not a leap year

s = s + "Invalid input date";

}

} **else** **if** (day >= 30) {

s = s + "Invalid input date";

}

}

**return** s;

}

// takes input error into account

**public** **static** String tomorrow(**int** day, **int** month, **int** year) {

String s = "";

**if** (*monthIsInRange*(month) && *dayIsInRange*(day) && *yearIsInRange*(year)) {

s = *computeTomorrow*(day, month, year);

} **else** {

**if** (!*monthIsInRange*(month)) {

s = s + "month out of range";

}

**if** (!*dayIsInRange*(month)) {

s = s + "day out of range";

}

**if** (!*yearIsInRange*(month)) {

s = s + "year out of range";

}

}

**return** s;

}

**public** **static** **void** main(String args[]) {

Scanner input = **new** Scanner(System.***in***);

System.***out***.print("Day: ");

**int** d = input.nextInt();

System.***out***.print("Month: ");

**int** m = input.nextInt();

System.***out***.print("Year: ");

**int** y = input.nextInt();

System.***out***.println(*tomorrow*(d, m, y));

}

}