Assignment 3 - G03P02

Group information

- Ana Inês Oliveira de Barros - `up201806593@fe.up.pt`;
- João de Jesus Costa - `up201806560@fe.up.pt`

Function Selection Process

From the previous assinment's report:

> The aim of this assignment is to perform black-box testing. This is problematic because none of methods in the code are documented (e.g. javadoc). In order to find the purpose of each method, we needed to follow our intuition about the names of the methods, arguments, and classes. Having more extensive documentation would allow for better black-box testing.
>
> Since we aren't using mocks, we tried to test methods that didn't depend on other objects of the project. We discarded functions belonging to the `gui` package due to its dependence on *swing*. The `misc` package was also discarded since it only contains one function (not enough for the completion of the assignment). Functions related to elapsed time were also ignored.
>
> The selected package for testing was the `de.dominik_geyer.jtimesched.project` package.

In this assignment, we selected 3 methods used in the previous one.

Method 1

Method: `void setSecondsOverall(int secondsOverall)` in `Project.java` line 178.

Method's purpose: This function sets the *seconds overall* of a project as the value it receives as an argument (if valid).

Reason for selection: It is important that this function works as expected since other methods depend on it.

Identify the parameters

`secondsOverall` - integer (`int`) representing the number of seconds overall.

Characteristics of the parameters

The integer should represent a positive number between 0 and infinity.

Constraints

Negative time is not allowed - `secondsOverall >= 0`

Partitions

- E1 - negative number
  - `secondsOverall < 0`
- E2 - positive number (including 0)
  - `secondsOverall >= 0`

Boundaries

Partition	On-point(s)	Off-point(s)
E1	`0`	`-1`
E2	`0`	`-1`

Generate tests

Partition	Boundary	Input
E1	On-point	0
E1	Off-point	-1
E2	On-point	0
E2	Off-point	-1

4 tests.

Filter redundant tests

Partition	Boundary	Input	Expected output
E1	On-point	0	0
E1	Off-point	-1	0

Filtered down to 2 tests.

Unit Tests

We created one test with the inputs of each line on the table. The test function is the `void setSecondsOverallTest(int secondsOverall)` and the input generator is `Stream<Arguments> setSecondsOverallInputs()`. Both of these are present in the `ProjectTest.java` file of the `test` package.

Results: all the tests pass successfully.

Method 2

Method: `public static int parseSeconds(String strTime)` in `ProjectTime.java` line 36.

Purpose: This function receives a string representing time, in `hh:mm:ss` format, and returns the total number of seconds it represents.

Reason for selection: This method deals with parsing of user input, which needs to be robust.

Identify the parameters

`strTime` is string representing time in 3 components:

- hours (hh);
- minutes (mm);
- seconds (ss).

Characteristics of the parameters

- The string should represent a valid time in the format: `hh:mm:ss`
- *hh* -- represents the hour
- *mm* -- represents the minutes
- *ss* -- represents the seconds
- It should be possible to pass single digits for each component

Constraints

- Input can't be `null`
    - `input != null`
- String can only contain digits and the `:` char
- Seconds lie within the interval [0, 59]
    - `0 <= ss <= 59`
- Minutes lie within the interval [0, 59]
    - `0 <= mm <= 59`

Partitions

- E1 - null input - `null`
- E2 - empty input - `""`
- E3 - non-empty string input - `"00:00:00"`

E3 can be sub-divided into other categories:

- E4 - input containing non-digit - `"0a:00:00"`
- E5 - input containing only digit and the `:` char - `"00:00:00"`
- E6 - seconds < 0 - `"00:00:-1"`
- E7 - seconds > 59 - `"00:00:60"`
- E8 - 0 <= seconds <= 59 - `00:00:30`
- E9 - minutes < 0 - `"00:-1:00"`
- E10 - minutes > 59 - `"00:60:00"`
- E11 - 0 <= minutes <= 59 - `00:30:00`

Boundaries

Partition	On-point(s)	Off-point(s)
E1	`null`	`""`, `"00:00:00"`
E2	`""`	`null`, `"00:00:00"`

Partition	On-point(s)	Off-point(s)
E3	`"00:00:00"`	`""`, `null`
E4	`"0a:00:00"`	`"00:00:00"`
E5	`"00:00:00"`	`"0a:00:00"`
E6	`"00:00:00"`	`"00:00:-1"`
E7	`"00:00:59"`	`"00:00:60"`
E8	`"00:00:00"` and `"00:00:59"`	`"00:00:-1"`, `"00:00:60"`
E9	`"00:00:00"`	`"00:-1:00"`
E10	`"00:59:00"`	`"00:60:00"`
E11	`"00:00:00"` and `"00:59:00"`	`"00:-1:00"`, `"00:60:00"`

Generate tests

Partition	Boundary	Input	Expected outcome
E1	On-point	`null`	Thrown exception
E1	Off-point 1	`""`	Thrown exception
E1	Off-point 2	`"00:00:00"`	Thrown exception
E2	On-point	`""`	Thrown exception
E2	Off-point 1	`null`	Thrown exception
E2	Off-point 2	`"00:00:00"`	Thrown exception
E3	On-point	`"00:00:00"`	Thrown exception
E3	Off-point 1	`""`	Thrown exception
E3	Off-point 2	`null`	Thrown exception
E4	On-point	`"0a:00:00"`	Thrown exception
E4	Off-point	`"00:00:00"`	Thrown exception
E5	On-point	`"00:00:00"`	Thrown exception
E5	Off-point	`"0a:00:00"`	Thrown exception
E6	On-point	`"00:00:00"`	Thrown exception
E6	Off-point	`"00:00:-1"`	Thrown exception
E7	On-point	`"00:00:59"`	Thrown exception
E7	Off-point	`"00:00:60"`	Thrown exception
E8	On-point 1	`"00:00:00"`	Thrown exception

Partition	Boundary	Input	Expected outcome
E8	On-point 2	`"00:00:59"`	Thrown exception
E8	Off-point 1	`"00:00:-1"`	Thrown exception
E8	Off-point 2	`"00:00:60"`	Thrown exception
E9	On-point	`"00:00:00"`	Thrown exception
E9	Off-point	`"00:-1:00"`	Thrown exception
E10	On-point	`"00:59:00"`	Thrown exception
E10	Off-point	`"00:60:00"`	Thrown exception
E11	On-point 1	`"00:00:00"`	Thrown exception
E11	On-point 2	`"00:59:00"`	Thrown exception
E11	Off-point 1	`"00:-1:00"`	Thrown exception
E11	Off-point 2	`"00:60:00"`	Thrown exception

29 tests.

Filter redundant tests

Partition	Boundary	Input	Expected outcome
E1	On-point	`null`	Thrown exception
E1	Off-point 1	`""`	Thrown exception
E1	Off-point 2	`"00:00:00"`	Thrown exception
E4	On-point	`"0a:00:00"`	Thrown exception
E6	Off-point	`"00:00:-1"`	Thrown exception
E7	On-point	`"00:00:59"`	Thrown exception
E7	Off-point	`"00:00:60"`	Thrown exception
E9	Off-point	`"00:-1:00"`	Thrown exception
E10	On-point	`"00:59:00"`	Thrown exception
E10	Off-point	`"00:60:00"`	Thrown exception

Filtered down to 10 tests.

Unit Tests

We created one test with the inputs of each line on the table.

The test function:

- `parseSecondsValidTest(String hours, String minutes, String seconds)`
- `parseSecondsInvalidTest(String timeStr)`

- `parseSecondsNullTest()`

The input generators (respectively):

- `parseSecondsValidInputs()`
- `parseSecondsInvalidInputs()`
- *No input generator*

All of these are present in the `ProjectTimeTest.java` file of the `test` package.

Results: All the test-cases pass successfully.

Method 3

Method: `void adjustSecondsToday(int secondsToday)` in `Project.java` line 192.

Method's purpose: This function sets the *seconds overall* of a project as the value it receives as an argument (if valid).

Reason for selection: It is important that this function works as expected since other methods depend on it.

Identify the parameters

`secondsToday` - integer (`int`) representing the number of seconds today.

Characteristics of the parameters

The integer should represent a positive number between 0 and infinity.

Constraints

Negative time is not allowed - `secondsToday >= 0`

Partitions

- E1 - negative number
  - `secondsToday < 0`
- E2 - positive number (including 0)
  - `secondsToday >= 0`

Boundaries

Partition	On-point(s)	Off-point(s)
E1	`0`	`-1`
E2	`0`	`-1`

Generate tests

Partition	Boundary	Input
E1	On-point	0

Partition	Boundary	Input
E1	Off-point	-1
E2	On-point	0
E2	Off-point	-1

4 tests.

Filter redundant tests

Partition	Boundary	Input	Expected output
E1	On-point	0	0
E1	Off-point	-1	0

Filtered down to 2 tests.

Unit Tests

We created one test with the inputs of each line on the table.

The test function:

- `void adjustSecondsValidTest(int secondsToday)`
- `void adjustSecondsInvalidTest(int secondsToday)`

The input generators (respectively):

- `Stream<Arguments> adjustSecondsValidInputs()`
- `Stream<Arguments> adjustSecondsInvalidInputs()`

All of these are present in the `ProjectTest.java` file of the `test` package.

Results: all the tests pass successfully.