Software development

Final Assignment Report

Members: Alejandra Galán Arróspide / Marcos Caballero Cortés
NIAs: 100451273 / 100451047
Group: 89

Introduction

In this report, we explain the general thinking and decision making for each method: Method 1 and its equivalent classes and boundary values; method 2 and the grammar of the input files, derivation tree and the corresponding tests using the syntax analysis.

An expanded analysis of the tests can be found on the excel files.

Method 1. get_vaccine_date

This method receives as a parameter, in addition to the JSON format file, a date in “YYYYY-MM-DD" format (iso format) with the proposed date for the vaccination. The difference is that we don’t get a number of days, now we get the exact date.
Then, we have to check the correctness of the inputs, for this we have to do some tests.

Tests

As in the next method, you can find all the related information of the tests in the documents section in our github repository.
We used the method studied in class. Equivalence class and boundary values technique was used to implement the tests of this function. The tests check the correctness of the input parameters.
We created 17 test cases from where we can see some differences in the tests:

- Boundary values for the dates: We checked the year number, so the dates with year “0000” are incorrect. We also defined test cases for the boundary values of the month and day, so the months from “01” to “12” included are valid and the days from “01” to “31” included are also valid.
- Equivalence classes and boundary limits related to errors in the strings: We can see some tests that check that the string is in “YYYYY-MM-DD" format. If there are more or less “-”, “Y”, “M” or “D” the test fails.
- Then we created some tests where our input, which is supposed to be correct, is checked so we can prove it is correct.

Method 2. cancel_appointment

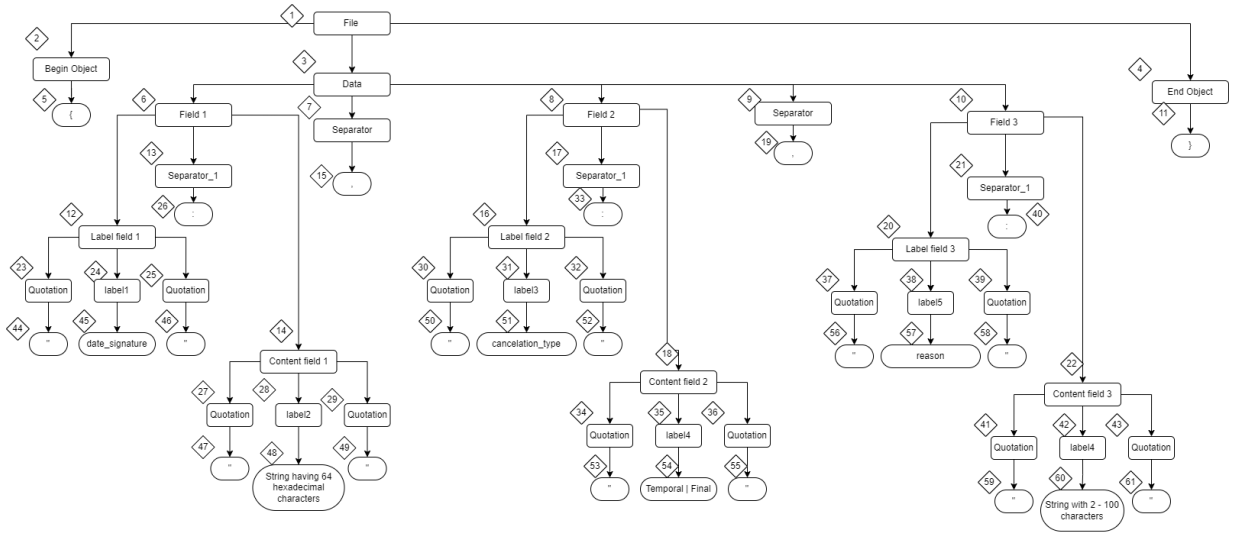
This method is used to cancel a vaccination appointment. It returns a string that represents the date_signature of the canceled appointment, or a VaccineManagementException.
The attribute Input_file represents the path to the file including the input required for the functionality.

Grammar

File ::= Begin Data End
Begin Object ::= {
End Object ::= }
Data ::= Field1 Separator Field2 Separator Field3
Separator ::= ,

Field 1 ::= Label field 1 Separator_1 Content Field 1
Label field 1::= Quotation label1 Quotation
Quotation ::= “
Content Field 1 ::= Quotation label2 Quotation
label1 ::= date_signature
label2 ::= a|b|c|d|e|f|0|1|2|3|4|5|6|7|8|9 (64)
Separator_1 ::= :
Field 2 ::= Label field 2 separator_1 Content field 2
Label field 2 ::= Quotation label 3 Quotation
label3 ::= cancelation_type
Content field 2 ::= Quotation labe l4 Quotation
label4::= Temporal / Final
Field3 ::= Label field 3 Separator_1 Content field 3
Label field 3 ::= Quotation label 5 Quotation
label5 ::= reason
Content field 3 ::= Quotation label 6 Quotation
label6::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|1|2|3|4|5|6|7|8|9|0 (2-100)

Derivation tree



Test cases

The test has been structured by separating terminals and not terminals. Note that for each terminal we made 2 Json files(duplicate and delete) and 1 Json file for each terminal (modify).
In total, we can find 56 Json files(considering that “ nodes and : nodes have not been repeated since they have the same structure).

We created different tests for checking everything. We first created a test for the valid case. Then we check if reason is duplicated outside of the parametrized test case, sice it is valid. Then we create another test case, where we check all the invalid cases. We have several test cases, where we check if the appointment does not exist, if the appointment date has already passed or if the appointment has already been canceled.

Flow graph

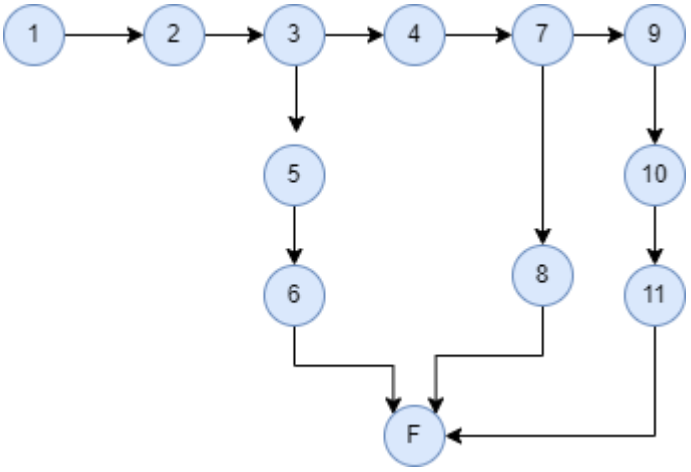
```
def create_cancellation_from_json_file(json_file):
    """Creates the Cancellation and returns its date_signature"""
    # Add the cancellation to the Cancellation Store
1   cancellation_parser = CancellationJsonParser(json_file)
2   json_date_signature = cancellation_parser.json_content[cancellation_parser.DATE_SIGNATURE_KEY]

3   try:
4       appointment_to_be_cancelled = VaccinationAppointment.get_appointment_from_date_signature(
            json_date_signature
        )
5   except VaccineManagementException as exception:
6       raise VaccineManagementException(
            "The appointment with the given date_signature does not exist") from exception

    # check if the date signature of the appointment is outdated
7   if appointment_to_be_cancelled.issued_at < datetime.timestamp(datetime.now()):
8       raise VaccineManagementException("The appointment with the given date_signature is outdated")

9   cancellations_store = CancellationJsonStore()
10  cancellations_store.add_item(cancellation_parser.json_content)

    # return the data signature of the cancellation
11  return cancellation_parser.json_content[cancellation_parser.DATE_SIGNATURE_KEY]
```



Conclusion

In this course we have learnt the basics of the software development process, learning the techniques shown in this course for specifying, designing, programming, and testing a simple software component.

In this last project we were able to apply all our knowledge of this subject: applying testing using boundary values, equivalence classes, grammar and derivation trees; following requirements; creating control flow graphs; applying refactorings; and applying singleton patterns.