Banking System Test Cases

Prepared by: Martin Ushilov, Lucas Pedemonte
Date: 25/03/2025
Course: Software Development

Table of Contents

Introduction

This document defines the test cases for the three banking functions implemented in the system using various software testing techniques. Each function is analyzed using appropriate testing methodologies to ensure correctness, robustness, and reliability.

- **Function 1: Transfer Request** – Uses **Equivalence Class Partitioning** and **Boundary Value Analysis**.
- **Function 2: Deposit Into Account** – Analyzed with **Syntax Analysis** (Grammar & Derivation Tree).
- **Function 3: Calculate Balance** – Evaluated with **Structural Testing** (Control Flow Graph & Path Analysis).

Test Cases for Function 1: Transfer Request

The first function processes a transfer request between two IBAN accounts. The following test cases ensure that valid transactions are processed correctly and invalid transactions are rejected.

Test Cases Summary

1. Valid Transfer Request

Description: Tests a valid transfer request using correct IBANs, a valid amount, and a valid date.
Input:

- from_iban: "ES9121000418450200051332"
- to_iban: "ES7921000813450200056789"
- concept: "House Rent"
- type: "ORDINARY"
- date: "7/7/2025"
- amount: 500.75

Expected Output:

- A 32-character transfer code (MD5 hash).
- **Valid Case**

2. Transfer with Minimum Valid Amount

Description: Ensures that the function correctly processes the lowest valid amount (10.00).
Input:

- amount: 10.00 (minimum valid)
- Other inputs remain valid.

Expected Output:

- A 32-character transfer code.
- **Valid Case**

3. Transfer with Maximum Valid Amount

Description: Ensures that the function correctly processes the highest valid amount (10,000.00).
Input:

- amount: 10,000.00 (maximum valid)
- Other inputs remain valid.
  Expected Output:
- A 32-character transfer code.
- **Valid Case**

4. Invalid IBAN (Wrong Country Code)

Description: Tests if the function rejects IBANs with an incorrect country code.
Input:

- from_iban: "FR9121000418450200051332" (should be ES)
- Other inputs remain valid.
  Expected Output:
- AccountManagementException
- **Invalid Case**

5. Invalid IBAN Format

Description: Tests if the function rejects an IBAN with incorrect characters.
Input:

- from_iban: "ES91A10004184502000513 3B" (contains letters in invalid places)
- Other inputs remain valid.
  Expected Output:
- AccountManagementException
- **Invalid Case**

6. Invalid IBAN Length

Description: Tests if an IBAN with only 23 characters is rejected.
Input:

- from_iban: "ES912100041845020005133" (missing last character)
- Other inputs remain valid.

Expected Output:

- AccountManagementException
- **Invalid Case**

7. Short Transfer Concept

Description: Tests if a transfer concept with fewer than 10 characters is rejected.
Input:

- concept: "Rent" (too short)
- Other inputs remain valid.

Expected Output:

- AccountManagementException
- **Invalid Case**

8. Concept with Exactly 9 Characters

Description: Tests a transfer concept right below the valid limit (should fail).
Input:

- concept: "Rent pay" (9 characters)
- Other inputs remain valid.

Expected Output:

- AccountManagementException
- **Invalid Case**

9. Invalid Transfer Type

Description: Tests if an unsupported transfer type is rejected.
Input:

- type: "FAST" (invalid type)
- Other inputs remain valid.

Expected Output:

- AccountManagementException
- **Invalid Case**

10. Transfer with a Past Date

Description: Ensures that past dates are rejected.
Input:

- date: "31/12/2024" (before current period)
- Other inputs remain valid.
  Expected Output:
- AccountManagementException
- **Invalid Case**

11. Transfer with a Date Beyond 2050

Description: Ensures dates beyond the allowed period are rejected.
Input:

- date: "1/1/2051" (too far in the future)
- Other inputs remain valid.
  Expected Output:
- AccountManagementException
- **Invalid Case**

12. Amount Below Minimum

Description: Ensures that transactions below 10.00 are rejected.
Input:

- amount: 9.99 (too low)
- Other inputs remain valid.
  Expected Output:
- AccountManagementException
- **Invalid Case**

13. Amount Above Maximum

Description: Ensures that transactions above 10,000.00 are rejected.
Input:

- amount: 10,000.01 (too high)
- Other inputs remain valid.

  Expected Output:
- AccountManagementException
- **Invalid Case**

__14. Valid Transfer Between Different Banks__

Description: Tests a valid transfer where both IBANs belong to different banks.
Input:

- from_iban: "ES4420385778983000760236"
- to_iban: "ES3320385778983000760238"
- Other inputs remain valid.

  Expected Output:
- A 32-character transfer code.
- **Valid Case**

__15. Valid Transfer Within the Same Bank__

Description: Tests a valid transfer where both IBANs belong to the same bank.
Input:

- from_iban: "ES9121000418450200051332"
- to_iban: "ES9121000418450200051340"
- Other inputs remain valid.

  Expected Output:
- A 32-character transfer code.
- **Valid Case**

__16. Concept with Maximum Valid Length (30 Characters)__

Description: Ensures a concept with 30 characters is accepted.
Input:

- concept: "Mortgage Payment March 25 Home" (30 chars)
- Other inputs remain valid.

  Expected Output:
- A 32-character transfer code.

- **Valid Case**

17. Concept with Minimum Valid Length (10 Characters)

Description: Ensures a concept with exactly 10 characters is accepted.
Input:

- concept: "Loan repay" (10 chars)
- Other inputs remain valid.
  Expected Output:
- A 32-character transfer code.
- **Valid Case**

18. Transfer with Different Valid Types

Description: Ensures the system supports multiple valid transfer types.
Input:

- type: "URGENT" (valid alternative)
- Other inputs remain valid.
  Expected Output:
- A 32-character transfer code.
- **Valid Case**

19. Transfer on Last Valid Date (31/12/2050)

Description: Ensures the latest possible date is accepted.
Input:

- date: "31/12/2050" (last valid date)
- Other inputs remain valid.
  Expected Output:
- A 32-character transfer code.
- **Valid Case**

20. Transfer on First Valid Date (25/03/2025)

Description: Ensures the earliest possible valid date is accepted.
Input:

- date: "25/03/2025" (earliest valid date)
- Other inputs remain valid.
  Expected Output:
- A 32-character transfer code.
- **Valid Case**

Test Cases for Function 2: Deposit Into Account

This function records a deposit operation. It validates the IBAN and amount before generating a deposit signature and saving the operation.

Grammar Definition

The syntax of the input JSON is formalized using BNF notation:

File ::= Start_object Data Fin_object

Start_object ::= "{"

Fin_object ::= "}"

Data ::= Field1 Separator Field2

Field1 ::= Label_data1 Equal Value_data1

Field2 ::= Label_data2 Equal Value_data2

Separator ::= ","

Equal ::= ":"

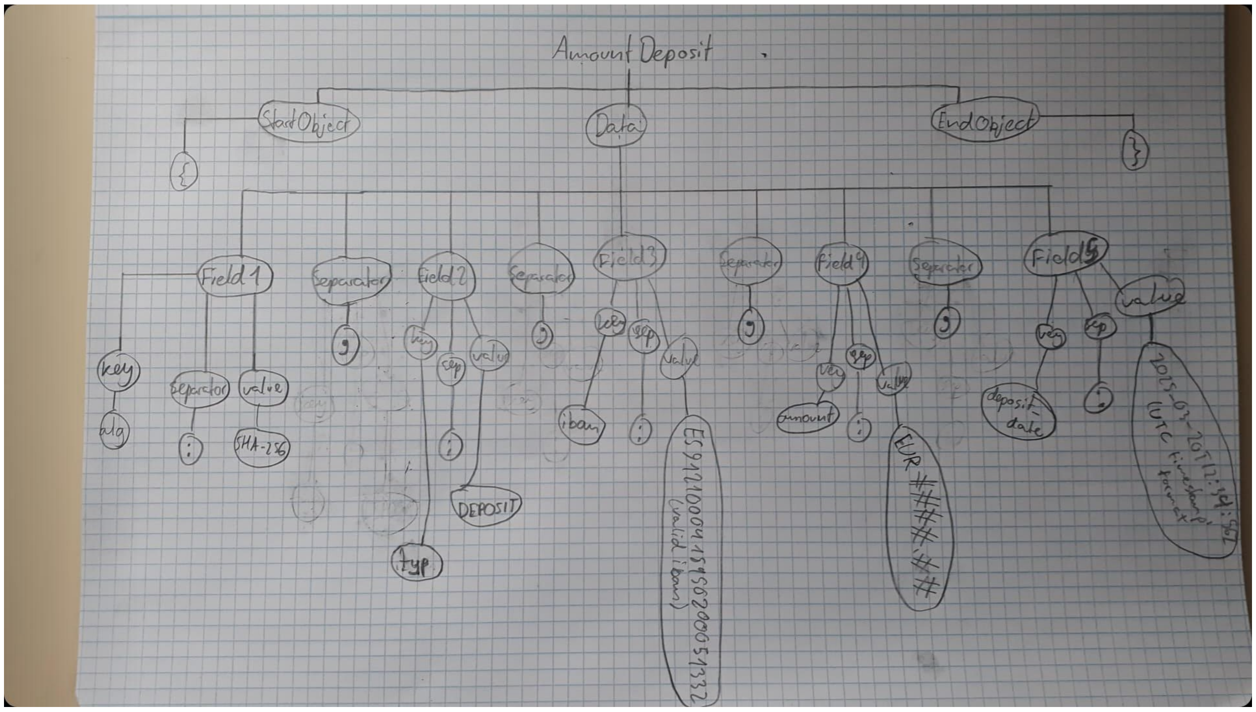
Label_data1 ::= "IBAN"

Label_data2 ::= "AMOUNT"

Value_data1 ::= ""

Value_data2 ::= "EUR ####.##"

Derivation Tree



Test Cases Summary

1. Valid Deposit Test

- **Valid JSON** → Ensures a correct JSON structure generates a valid signature.

2. JSON Structure Tests

- **Malformed Braces** → Tests for missing or extra opening/closing braces.
- **Empty Data** → Checks if an empty JSON object is rejected.
- **Duplicate Data** → Ensures duplicate key-value pairs cause an error.

3. Separator Issues

- **Missing or Duplicate Separators** → Ensures correct use of colons (:) and commas (,) in key-value pairs.

4. 'alg' (Algorithm) Key Tests

- **Missing/Modified Key** → Ensures "alg" key must be present and correctly named.
- **Duplicate Key** → Detects multiple "alg" keys in JSON.
- **Invalid Value** → Ensures "alg" has a valid hash algorithm (e.g., rejects "MD5").
- **Missing or Duplicate Value** → Checks for absent or repeated algorithm values.

5. 'typ' (Type) Key Tests

- **Missing/Modified Key** → Ensures "typ" key must be present and correctly named.
- **Duplicate Key** → Detects multiple "typ" keys in JSON.
- **Invalid Value** → Ensures "typ" is a valid transaction type (e.g., rejects "TRANSFER").
- **Missing or Duplicate Value** → Checks for absent or repeated type values.

6. 'iban' Key Tests

- **Missing/Modified Key** → Ensures "iban" key must be present and correctly named.
- **Duplicate Key** → Detects multiple "iban" keys.
- **Invalid Value** → Ensures "iban" follows valid IBAN format.

7. 'amount' Key Tests

- **Missing/Modified Key** → Ensures "amount" key must be present and correctly named.
- **Duplicate Key** → Detects multiple "amount" keys.

- **Invalid Value** → Ensures "amount" is correctly formatted (e.g., "EUR 1000.50").

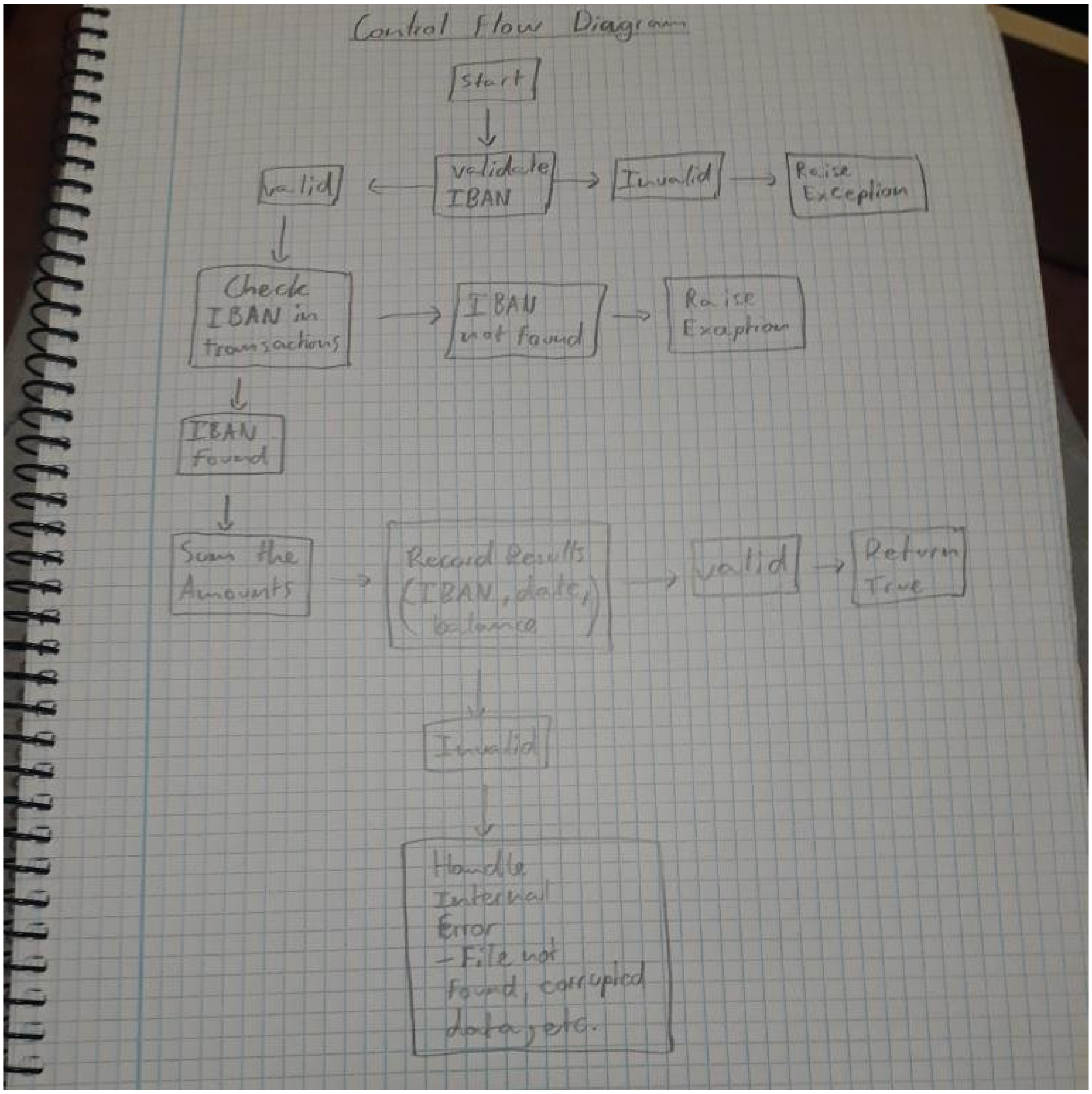
8. 'deposit_date' Key Tests

- **Missing/Modified Key** → Ensures "deposit_date" key must be present and correctly named.
- **Duplicate Key** → Detects multiple "deposit_date" keys.
- **Invalid Value** → Ensures date format is valid (e.g., rejects "INVALID_DATE").
- **Missing Value** → Ensures deposit date is not empty.

Test Cases for Function 3: Calculate Balance

This function calculates the balance of an IBAN from a transactions file.

Control Flow Graph (CFG)



Basic Paths Definition

The following paths represent the distinct execution flows:

1. **Path 1:** Start → Validate IBAN → IBAN Exists → Sum Amounts → Store Balance → End
2. **Path 2:** Start → Validate IBAN → IBAN Does Not Exist → Exception → End
3. **Path 3:** Start → Validate IBAN → File Error → Exception → End
4. **Path 4:** Start → Validate IBAN → Sum Amounts → File Write Error → Exception → End

Extra Test Cases for Loops

Since the function iterates over transactions, we must test:

- Zero iterations (No transactions for IBAN)
- One iteration (Single transaction)
- Multiple iterations (Multiple transactions for IBAN)

Test Cases Summary

1. Valid IBAN & Transaction Processing

- **Valid IBAN - Sum Transactions** → Ensures correct balance calculation for a valid IBAN with deposits.
- **Valid IBAN - Only Deposits** → Verifies deposits are summed correctly.
- **Valid IBAN - Only Withdrawals** → Ensures withdrawals are deducted correctly.
- **Valid IBAN - Mixed Transactions** → Confirms the system correctly processes both deposits and withdrawals.
- **Valid IBAN - Edge Case (Zero Transaction Amount)** → Tests handling of a zero-amount transaction.

2. Invalid or Missing IBAN Handling

- **Invalid IBAN Exception** → Ensures an error is raised for an invalid IBAN.
- **IBAN Not in Transactions** → Tests if the system detects when an IBAN is missing from records.

3. Transaction File Handling

- **Empty Transactions File Exception** → Ensures an error is raised when no transactions are present.
- **File Not Found Exception** → Tests error handling when the transactions file is missing.

- **Corrupted JSON File Exception** → Verifies an exception is raised for an invalid JSON structure.

Conclusion

This document outlines the test cases, grammar rules, and structural testing techniques used to validate the three banking functions. By applying Equivalence Class Partitioning, Boundary Value Analysis, Syntax Analysis, and Control Flow Graphs, we ensure concrete testing coverage.