Test Scenarios for Class1 Public API

Overview

The Class1 class provides a repository for storing and managing items with content in JSON or XML format. The following test scenarios will cover the public methods of the class to ensure their correct behavior.

Test Scenarios

1. Initialization

Test Case 1.1: Initialization

- **Objective**: Verify that the repository initializes correctly on instantiation.
- **Test Steps**:
    1. Create an instance of Class1.
- **Expected Result**: The instance should be created without any exceptions, and _initialized should be true.

2. Register Method

Test Case 2.1: Successful Registration of a New Item

- **Objective**: Verify that a new item can be registered successfully.
- **Test Steps**:
    1. Call Register("item1", "{\"key\": \"value\"}", 1).
- **Expected Result**: The item should be added to the repository without exceptions.

Test Case 2.2: Duplicate Registration

- **Objective**: Verify that an exception is thrown when registering an item with an existing name.
- **Test Steps**:
    1. Call Register("item1", "{\"key\": \"value\"}", 1).
    2. Call Register("item1", "{\"key\": \"value\"}", 1) again.
- **Expected Result**: An InvalidOperationException should be thrown with the message "Item with name 'item1' already exists."

3. Retrieve Method

Test Case 3.1: Successful Retrieval

- **Objective**: Verify that an item can be retrieved successfully.
- **Test Steps**:
    1. Register an item with Register("item1", "{\"key\": \"value\"}", 1).
    2. Call Retrieve("item1").
- **Expected Result**: The method should return "{"key": "value"}".

Test Case 3.2: Retrieval of a Non-Existent Item

- **Objective**: Verify that an exception is thrown when retrieving a non-existent item.
- **Test Steps**:
    1. Call Retrieve("nonexistentItem").
- **Expected Result**: A KeyNotFoundException should be thrown with the message "Item with name 'nonexistentItem' not found."

4. GetType Method

Test Case 4.1: Successful Type Detection for JSON

- **Objective**: Verify that the method returns the correct type for JSON content.
- **Test Steps**:
    1. Call GetType("{\"key\": \"value\"}").
- **Expected Result**: The method should return 1.

Test Case 4.2: Successful Type Detection for XML

- **Objective**: Verify that the method returns the correct type for XML content.
- **Test Steps**:
    1. Call GetType("<root><key>value</key></root>").
- **Expected Result**: The method should return 2.

Test Case 4.3: Invalid Content Type

- **Objective**: Verify that an exception is thrown for content that is neither JSON nor XML.
- **Test Steps**:
    1. Call GetType("plain text").
- **Expected Result**: An InvalidDataException should be thrown with the message "Item content is neither JSON nor XML."

5. Deregister Method

Test Case 5.1: Successful Deregistration

- **Objective**: Verify that an item can be deregistered successfully.
- **Test Steps**:
    1. Register an item with Register("item1", "{\"key\": \"value\"}", 1).
    2. Call Deregister("item1").
- **Expected Result**: The item should be removed from the repository without exceptions.

Test Case 5.2: Deregistration of a Non-Existent Item

- **Objective**: Verify that an exception is thrown when attempting to deregister a non-existent item.
- **Test Steps**:
    1. Call Deregister("nonexistentItem").
- **Expected Result**: A KeyNotFoundException should be thrown with the message "Item with name 'nonexistentItem' not found or could not be removed."

6. Validate Method

Test Case 6.1: Valid JSON Content

- **Objective**: Verify that valid JSON content passes validation.
- **Test Steps**:
    1. Call Validate("{\"key\": \"value\"}", 1).
- **Expected Result**: The method should complete without exceptions.

Test Case 6.2: Invalid JSON Content

- **Objective**: Verify that invalid JSON content throws an exception.
- **Test Steps**:
    1. Call Validate("{key: value}", 1).
- **Expected Result**: A FormatException should be thrown with the message "Invalid JSON format."

Test Case 6.3: Valid XML Content

- **Objective**: Verify that valid XML content passes validation.

- **Test Steps**:
    1. Call Validate("<root><key>value</key></root>", 2).
- **Expected Result**: The method should complete without exceptions.

Test Case 6.4: Invalid XML Content

- **Objective**: Verify that invalid XML content throws an exception.
- **Test Steps**:
    1. Call Validate("<root><key>value</key>", 2).
- **Expected Result**: A FormatException should be thrown with the message "Invalid XML format."