Backend Assignment - Store and Solve Algebraic Equation

Problem Statement:

Develop a Spring Boot Web Application for Storing and Evaluating Algebraic Equations Using a Postfix Tree.

The task is to create a RESTful Spring Boot application that uses a postfix tree (expression tree) to manage algebraic equations. The application should:

1. **Store algebraic equations**: Parse and save equations in a postfix (Reverse Polish Notation) tree structure.
2. **Retrieve stored equations**: Fetch a list of stored equations reconstructed from the postfix tree.
3. **Evaluate a specific equation**: Substitute provided variable values and calculate the result using the postfix tree.

The application should support JSON-based requests and responses and use in-memory storage.

Expected API Endpoints and Specifications:

1. Store an Algebraic Equation

- **URL**: `/api/equations/store`
- **HTTP Method**: `POST`

Request Body:
json
Copy code
```
{
  "equation": "3x + 2y - z"
}
```

-

Response:
json
Copy code
```
{
  "message": "Equation stored successfully",
  "equationId": "1"
}
```

- **Functionality**:
  - Parse the equation into postfix notation (e.g., `3x 2y + z -`) and store it in a tree structure where each operator is a parent node, and operands are its children.

---

2. Retrieve Stored Equations

- **URL**: `/api/equations`
- **HTTP Method**: `GET`
- **Request Body**: (none)

Response:
json
Copy code
```
{
  "equations": [
    {
      "equationId": "1",
      "equation": "3x + 2y - z"
    },
    {
      "equationId": "2",
      "equation": "x^2 + y^2 - 4"
    }
  ]
}
```

- **Functionality**:
  - Traverse the postfix tree and reconstruct the infix representation of the stored equations for display.

---

3. Evaluate an Equation

- **URL**: `/api/equations/{equationId}/evaluate`
- **HTTP Method**: `POST`

Request Body:
json
Copy code
```
{
  "variables": {
    "x": 2,
    "y": 3,
    "z": 1
  }
}
```

-

Response:
json
Copy code

```
{
  "equationId": "1",
  "equation": "3x + 2y - z",
  "variables": {
    "x": 2,
    "y": 3,
    "z": 1
  },
  "result": 10
}
```

- **Functionality**:
    - Use the postfix tree to evaluate the equation by substituting provided variable values.

Notes for Applicants:

1. **Postfix Tree Implementation**:
    - Parse the input equation into postfix notation.
    - Construct a postfix (expression) tree where operators are parent nodes, and operands are child nodes.
2. **Validation and Error Handling**:
    - Validate equations for correct syntax and operators.
    - Handle missing or invalid variable values gracefully during evaluation.
3. **Testing**:
    - Provide unit tests for storing, retrieving, and evaluating equations.
    - Include test cases for edge scenarios like complex equations, missing operators, or invalid input.
4. **Documentation**:
    - Include a README file with setup instructions, application overview, and test execution steps.

Evaluation Criteria:

- Assignment code submitted to be in **Java**
- REST Compliant APIs, (No UI)
- Testable by Postman, (No UI)
- Will prefer classes properly refactored and following design patterns,
- Clarity of API documentation and how to run the project.
- Will Prefer TDD, JUnit , **Test Coverage**: Comprehensive test cases covering various equation scenarios.
- **Correctness**: Accurate parsing, storage, and evaluation of equations using the postfix tree.
- **Error Handling**: Robust validation and error messages.