**Flowchart: Flexible Calculator Implementation**

This flowchart outlines the core functionalities of the calculator implementation.

**Start**

-> **Define Operation Enum** (ADD, SUBTRACT, MULTIPLY, DIVIDE)

-> **Create Calculator Interface**

* calculate(Operation, Number, Number): Performs calculation based on operation.
* chain(Operation, Number): Chains another operation with a new number.
* getResult(): Retrieves the final result.

-> **Implement Calculator Interface** (SimpleCalculator Class)   
| Input | Processing | Output |   
|---|---|---|  
| Initial Value | Stores in result variable | |

| calculate (op, num1, num2) |

- Switch on op:

- ADD: num1 + num2

- SUBTRACT: num1 - num2

- MULTIPLY: num1 \* num2

- DIVIDE:

- Check for division by zero (throw exception)

- Otherwise: num1 / num2

- Default: Throw UnsupportedOperationException

| chain(op, num) |

- Calculates using calculate(op, result, num)

- Updates result with the new value

- Returns this for chaining

| getResult() | Returns the current value of result

-> **Unit Testing (Optional)**

* Test basic operations (ADD, SUBTRACT, MULTIPLY, DIVIDE)
* Test chaining operations
* Test division by zero (exception handling)
* Test unsupported operations (exception handling)

-> **Extensibility**

* Add new operations by modifying the Operation enum and adding logic in calculate.

-> **IoC Compatibility**

* Use the Calculator interface for dependency injection.