Stack Class

The Stack class provides an implementation of a **stack data structure** using an array in Java. A stack follows the **Last In, First Out (LIFO)** principle, where elements are added to and removed from the top of the stack.

Attributes

- maxSize: The maximum size of the stack.
- pos: The current position of the top element in the stack.
- stackArr: Array that stores stack elements.

Constructors

- Stack(int size): Initializes a stack with the specified maximum size.

Methods

- **boolean isEmpty()**:
  - **Purpose**: Checks if the stack is empty.
  - **Returns**: true if the stack is empty, otherwise false.
- **int pop()**:
  - **Purpose**: Removes and returns the top element of the stack.
  - **Returns**: The top element, or -1 if the stack is empty.
- **void push(int element)**:
  - **Purpose**: Adds an element to the top of the stack.
  - **Parameters**: element - the integer value to be added.
  - **Throws**:
    - StackOverflowError if the stack is full.
    - IllegalArgumentException if element is negative.
- **int top()**:
  - **Purpose**: Retrieves the top element without removing it.
  - **Returns**: The top element, or -1 if the stack is empty.
- **int size()**:
  - **Purpose**: Returns the current number of elements in the stack.
  - **Returns**: The stack size as an integer.

StackTest Class

The StackTest class provides unit tests for the Stack class using the **JUnit** framework. These tests ensure that the stack implementation behaves as expected.

Test Methods

- **void checkIfStackIsEmpty()**
  - **Purpose**: Tests if a newly created stack is empty, if it becomes non-empty after pushing an element, and empty again after popping it.
- **void verifyPushFunctionality()**
  - **Purpose**: Tests if elements can be pushed onto the stack and if they appear in the correct order when popped.
- **void validatePopOperation()**
  - **Purpose**: Verifies the behavior of pop() on both empty and non-empty stacks.
- **void checkStackSize()**
  - **Purpose**: Confirms that the size of the stack changes correctly as elements are pushed and popped.
- **void handlePushExceptions()**
  - **Purpose**: Tests if the correct exceptions are thrown when attempting to push an invalid element (negative value) or when pushing into a full stack.
- **void testTop()**
  - **Purpose**: Checks the behavior of top() for both empty and non-empty stacks.