Boundary Value Test

To perform boundary value testing on your application, we will use the generalized boundary value analysis approach. Here's how to proceed:

1. **Identify Variables**: Determine the variables to be tested in your application.
2. **Select Boundary Values**: For each variable, select the minimum, just above the minimum, nominal, just below the maximum, and maximum values.
3. **Test Cases**: Generate test cases based on the 4n+1 rule.

The 4n+1 formula for boundary value analysis helps determine the number of test cases needed when dealing with multiple variables. Here's a breakdown of how it works:

1. **n**: Number of variables.
2. **4n**: Four test cases per variable (minimum, just above minimum, just below maximum, and maximum).
3. **+1**: One nominal test case where all variables are set to nominal values.

Calculation

For each variable, x, we select five values:

- The minimum $x_{\min}$
- Slightly above the minimum $x_{\min+}$
- The nominal x_{nom}
- Slightly below the maximum $x_{\max-}$
- The maximum $x_{\max}$

Therefore, for *n* variables:

- **4n** test cases for boundary values.
- **+1** test case for the nominal value of all variables.

Step-by-Step Process

1. **Identify Variables**:
    - Let's assume we're testing the signup method of UserManager class.
    - Variables: **username**, **password**, **email**, **fullName**.
    - Thus, **n = 4**.

    Applying the Formula

    We have 4 variables, the number of test cases would be calculated as follows:
    - **4 variables**: 4 * 4 + 1 = 16 + 1 = 17 test cases.

2. **Define Boundary Values**:
    - For the sake of example, assume the following:
        - username: Min length 3, Max length 20
            - $x_{\min}$: "abc"
            - $x_{\min+}$: "abcd"
            - $x_{\max-}$: "a"*19

- x_{max}: "a"*20
- x_{nomx}: "user123"

- password: Min length 8, Max length 16, must include upper/lower case, digits, symbols
  - x_{min}: "A1@abcde"
  - x_{min+} : "A1@abcdef"
  - x_{max-} : "A1@bcdefghijklmno"
  - x_{max}: "A1@bcdefghijklmnop"
  - x_{nomx}: "Password1!"
- email: Must follow a valid email format
  - x_{min}: " a@b.c"
  - x_{min+} : " user@example.com"
  - x_{max-} : " test123@example.com"
  - x_{max}: " longemailaddress@example.com"
  - x_{nomx}: " test@example.com"
- fullName: Min length 1, Max length 50
  - x_{min}: "a"
  - x_{min+} : "ab"
  - x_{max-} : "A"*49
  - x_{max}: "A"*49
  - x_{nomx}: "John Doe"

Test Case Table

Case	username	password	email	fullName
1	""	"Password1!"	"test@example.com"	"John Doe"
2	"abc"	"Password1!"	"test@example.com"	"John Doe"
3	"a"*20	"Password1!"	"test@example.com"	"John Doe"
4	"a"*19	"Password1!"	"test@example.com"	"John Doe"
5	"user5"	"A1@abcde"	"test@example.com"	"John Doe"
6	"user1"	"A1@abcdef"	"test@example.com"	"John Doe"
7	"user2"	"A1@bcdefghijklmno"	"test@example.com"	"John Doe"
8	"user3"	"A1@bcdefghijklmnop"	"test@example.com"	"John Doe"
9	"user4"	"Password1!"	"a@b.c"	"John Doe"
10	"user6"	"Password1!"	"user@example.com"	"John Doe"
11	"user7"	"Password1!"	"test123@example.com"	"John Doe"
12	"user8"	"Password1!"	"longemailaddress@example.com"	"John Doe"
13	"user9"	"Password1!"	"test@example.com"	"A"
14	"user10"	"Password1!"	"test@example.com"	"AB"
15	"user11"	"Password1!"	"test@example.com"	"A"*49
16	"user12"	"Password1!"	"test@example.com"	"A"*50
17	"user13"	"Password1!"	"test@example.com"	"John Doe"

Java Implementation for Boundary Value Testing

```
package PathTesting;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import Proj_375_Classes.UserManager;

class BoundaryValueTest {

   private UserManager userManager = new UserManager();

   @Test
   void testBoundaryValues() {
      // Case 1: Username is empty
      System.out.println("Running test case 1, Username is empty...");
      assertFalse(userManager.signup("", "Password1!", "Password1!",
"test@example.com", "John Doe"));

      // Case 2: Username is at minimum length just above empty
      System.out.println("Running test Case 2: Username is at minimum length just above
empty...");
      assertTrue(userManager.signup("abc", "Password1!", "Password1!",
"test@example.com", "John Doe"));

      // Case 3: Username is at maximum length
      System.out.println("Running test Case 3: Username is at maximum length...");
      assertTrue(userManager.signup("aaaaaaaaaaaaaaaaaaaa", "Password1!",
"Password1!", "test@example.com", "John Doe"));

      // Case 4: Username is just below maximum length
      System.out.println("Running test Case 4: Username is just below maximum length...");
      assertTrue(userManager.signup("aaaaaaaaaaaaaaaaaaa", "Password1!",
"Password1!", "test@example.com", "John Doe"));

      // Case 5: Password is at minimum length with criteria met
      System.out.println("Running test Case 5: Password is at minimum length with criteria
met...");
      assertTrue(userManager.signup("user5", "A1@abcde", "A1@abcde",
"test@example.com", "John Doe"));

      // Case 5a: Password does not match confirmation
      System.out.println("Running test Case 5a: Password does not match confirmation...");
```

```
    assertFalse(userManager.signup("user123", "A1@abcde", "A1@abcd",
"test@example.com", "John Doe"));

    // Case 6: Password is slightly above minimum length
    System.out.println("Running test Case 6: Password is slightly above minimum
length...");
    assertTrue(userManager.signup("user1", "A1@abcdef", "A1@abcdef",
"test@example.com", "John Doe"));

    // Case 7: Password is just below maximum length
    System.out.println("Running test Case 7: Password is just below maximum length...");
    assertTrue(userManager.signup("user2", "A1@bcdefghijklmno", "A1@bcdefghijklmno",
"test@example.com", "John Doe"));

    // Case 8: Password is at maximum length
    System.out.println("Running test Case 8: Password is at maximum length...");
    assertTrue(userManager.signup("user3", "A1@bcdefghijklmnop",
"A1@bcdefghijklmnop", "test@example.com", "John Doe"));

    // Case 9: Invalid email format
    System.out.println("Running test Case 9: Invalid email format...");
    assertFalse(userManager.signup("user4", "Password1!", "Password1!", "a@b.c", "John
Doe"));

    // Case 10: Valid email
    System.out.println("Running test Case 10: Valid email...");
    assertTrue(userManager.signup("user6", "Password1!", "Password1!",
"user@example.com", "John Doe"));

    // Case 11: Valid long email
    System.out.println("Running test Case 11: Valid long email...");
    assertTrue(userManager.signup("user7", "Password1!", "Password1!",
"test123@example.com", "John Doe"));

    // Case 12: Valid extra long email
    System.out.println("Running test Case 12: Valid extra long email...");
    assertTrue(userManager.signup("user8", "Password1!", "Password1!",
"longemailaddress@example.com", "John Doe"));

    // Case 13: Full name is empty
    System.out.println("Running test Case 13: Full name is empty...");
    assertFalse(userManager.signup("user9", "Password1!", "Password1!",
"test@example.com", ""));
```

```
        // Case 14: Full name is at minimum valid length
        System.out.println("Running test Case 14: Full name is at minimum valid length...");
        assertTrue(userManager.signup("user10", "Password1!", "Password1!",
"test@example.com", "A"));

        // Case 15: Full name is just below maximum length
        System.out.println("Running test Case 15: Full name is just below maximum length...");
        assertTrue(userManager.signup("user11", "Password1!", "Password1!",
"test@example.com", "A".repeat(49)));

        // Case 16: Full name is at maximum length
        System.out.println("Running test Case 16: Full name is at maximum length...");
        assertTrue(userManager.signup("user12", "Password1!", "Password1!",
"test@example.com", "A".repeat(50)));

        // Case 17: Nominal case
        System.out.println("Running test Case 17: Nominal case...");
        assertTrue(userManager.signup("user13", "Password1!", "Password1!",
"test@example.com", "John Doe"));
    }
}
```

This method helps in thoroughly testing the boundary conditions of your application, ensuring robustness and reliability.