

PL/SQL

EXERCISE 1:

1. IF...THEN

[SQL Worksheet]*

```
1  SET SERVEROUTPUT ON;
2
3  DECLARE
4  ... salary NUMBER := 15000;
5  BEGIN
6  ... IF salary > 10000 THEN
7  ... .. DBMS_OUTPUT.PUT_LINE('High Salary');
8  ... END IF;
9  END;
10 /
11
```

Query result

Script output

DBMS output

Explain Plan

SQL history

SQL> DECLARE
salary NUMBER := 15000;
BEGIN
IF salary > 10000 THEN...
Show more...

High Salary

PL/SQL procedure successfully completed.

2. IF...THEN...ELSE

[SQL Worksheet]*

```
1  SET SERVEROUTPUT ON;
2
3  DECLARE
4  ... salary NUMBER := 12000;
5  BEGIN
6  ... IF salary > 10000 THEN
7  ... .. DBMS_OUTPUT.PUT_LINE('High Salary');
8  ... ELSE
9  ... .. DBMS_OUTPUT.PUT_LINE('Normal Salary');
10 ... END IF;
11 END;
12
```

Query result

Script output

DBMS output

Explain Plan

SQL history

SQL> DECLARE
salary NUMBER := 12000;
BEGIN
IF salary > 10000 THEN...
Show more...

High Salary

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.006

2. 3. IF...ELSIF...ELSE

[SQL Worksheet]*

```
1  SET SERVEROUTPUT ON;
2  DECLARE
3  ... salary NUMBER := 12000;
4  BEGIN
5  ... IF salary > 20000 THEN
6  ... .. DBMS_OUTPUT.PUT_LINE('Very High Salary');
7  ... ELSIF salary > 10000 THEN
8  ... .. DBMS_OUTPUT.PUT_LINE('High Salary');
9  ... ELSE
10 ... .. DBMS_OUTPUT.PUT_LINE('Average Salary');
11 ... END IF;
12 END;
13
```

Query result

Script output

DBMS output

Explain Plan

SQL history

SQL> DECLARE
salary NUMBER := 12000;
BEGIN
IF salary > 20000 THEN...
Show more...

High Salary

PL/SQL procedure successfully completed.

4. Simple CASE Statement

[SQL Worksheet]*

```
1  SET SERVEROUTPUT ON;
2  DECLARE
3  ... grade CHAR(1) := 'B';
4  BEGIN
5  ... CASE grade
6  ... .. WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
7  ... .. WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Good');
8  ... .. WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('Fair');
9  ... .. ELSE DBMS_OUTPUT.PUT_LINE('Needs Improvement');
10 ... END CASE;
11 END;
12 /
13
```

Query result

Script output

DBMS output

Explain Plan

SQL history

SQL> DECLARE
grade CHAR(1) := 'B';
BEGIN
CASE grade...
Show more...

Good

PL/SQL procedure successfully completed.

5. Searched CASE Statement

[SQL Worksheet]*

```
1 SET SERVEROUTPUT ON;
2 DECLARE
3   marks NUMBER := 78;
4 BEGIN
5   CASE
6     WHEN marks >= 90 THEN DBMS_OUTPUT.PUT_LINE('A Grade');
7     WHEN marks >= 75 THEN DBMS_OUTPUT.PUT_LINE('B Grade');
8     WHEN marks >= 60 THEN DBMS_OUTPUT.PUT_LINE('C Grade');
9     ELSE DBMS_OUTPUT.PUT_LINE('Fail');
10  END CASE;
11 END;
12 /
13
```

Query result **Script output** DBMS output Explain Plan SQL history

SQL> DECLARE
marks NUMBER := 78;
BEGIN
CASE ...
Show more...

B Grade

PL/SQL procedure successfully completed.

6. LOOP (Basic loop)

[SQL Worksheet]*

```
1 SET SERVEROUTPUT ON;
2 DECLARE
3   i NUMBER := 1;
4 BEGIN
5   LOOP
6     DBMS_OUTPUT.PUT_LINE('i = ' || i);
7     i := i + 1;
8     EXIT WHEN i > 5;
9   END LOOP;
10 END;
11 /
12
13
```

Query result **Script output** DBMS output Explain Plan SQL history

i = 1
i = 2
i = 3
i = 4
i = 5

7. WHILE loop

```
1 SET SERVEROUTPUT ON;
2 DECLARE
3   i NUMBER := 1;
4 BEGIN
5   WHILE i <= 7 LOOP
6     DBMS_OUTPUT.PUT_LINE('i = ' || i);
7     i := i + 1;
8   END LOOP;
9 END;
10 /
11
12
```

Query result **Script output** DBMS output Explain Plan SQL history

i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7

PL/SQL procedure successfully completed.

8. FOR loop

```
1 SET SERVEROUTPUT ON;
2 DECLARE
3 BEGIN
4   FOR i IN 1..5 LOOP
5     DBMS_OUTPUT.PUT_LINE('i = ' || i);
6   END LOOP;
7 END;
8 /
9
```

Query result **Script output** DBMS output Explain Plan SQL history

i = 1
i = 2
i = 3
i = 4
i = 5

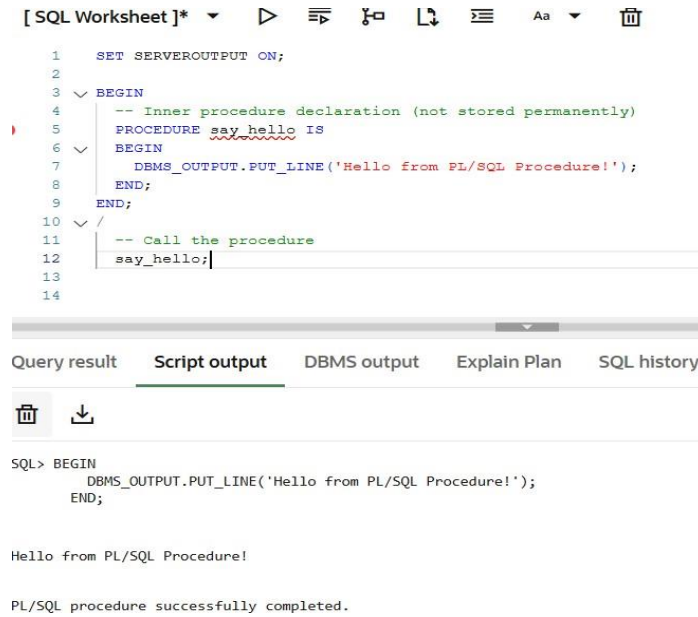
PL/SQL procedure successfully completed.

EXERCISE 2: STORED PROCEDURE What is a

Stored Procedure?

A stored procedure is a named block of code that performs a task and is stored in the Oracle database. You can call it whenever needed.

1. Simple Stored Procedure: Basic procedure with no parameters.



The screenshot shows the SQL Developer interface with a script titled "[SQL Worksheet]*". The script contains the following PL/SQL code:

```
1 SET SERVEROUTPUT ON;
2
3 BEGIN
4   -- Inner procedure declaration (not stored permanently)
5   PROCEDURE say_hello IS
6   BEGIN
7     DBMS_OUTPUT.PUT_LINE('Hello from PL/SQL Procedure!');
8   END;
9 END;
10
11 -- Call the procedure
12 say_hello;
13
14
```

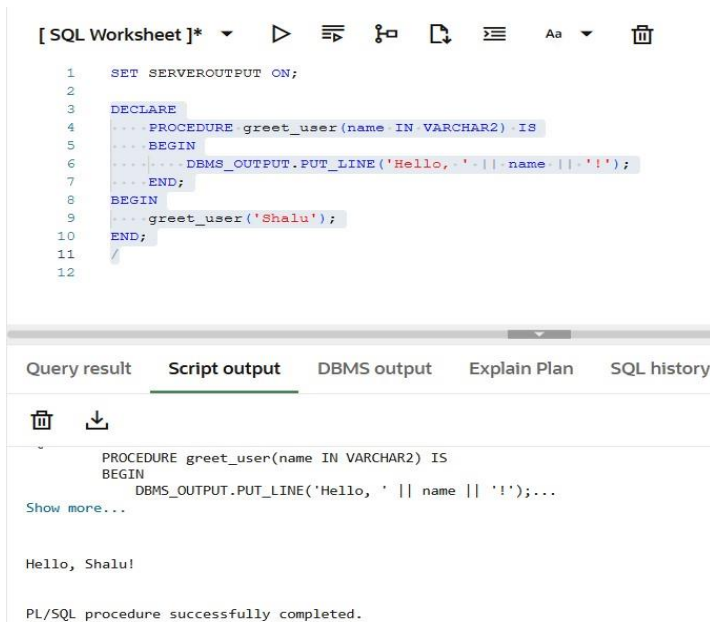
Below the script editor, the "Script output" tab is active, showing the execution results:

```
SQL> BEGIN
      DBMS_OUTPUT.PUT_LINE('Hello from PL/SQL Procedure!');
      END;

Hello from PL/SQL Procedure!

PL/SQL procedure successfully completed.
```

2. Procedure with IN Parameter: Learn how to pass input to a procedure



The screenshot shows the SQL Developer interface with a script titled "[SQL Worksheet]*". The script contains the following PL/SQL code:

```
1 SET SERVEROUTPUT ON;
2
3 DECLARE
4   ...PROCEDURE greet_user(name IN VARCHAR2) IS
5   ...BEGIN
6   ...DBMS_OUTPUT.PUT_LINE('Hello, ' || name || '!');
7   ...END;
8 BEGIN
9   ...greet_user('Shalu');
10 END;
11
12
```

Below the script editor, the "Script output" tab is active, showing the execution results:

```
PROCEDURE greet_user(name IN VARCHAR2) IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Hello, ' || name || '!');...
Show more...

Hello, Shalu!

PL/SQL procedure successfully completed.
```

3. Procedure with IN and OUT Parameters: Learn how to return values from a procedure using OUT.

[SQL Worksheet]*

```

1  SET SERVEROUTPUT ON;
2
3  DECLARE
4      result NUMBER;
5
6      PROCEDURE get_square(num IN NUMBER, square OUT NUMBER) IS
7      BEGIN
8          square := num * num;
9      END;
10 BEGIN
11     get_square(6, result);
12     DBMS_OUTPUT.PUT_LINE('Square is: ' || result);
13 END;
14 /

```

Query result
Script output
DBMS output
Explain Plan
SQL history

```

result NUMBER; -- ☒ Declare variable first
PROCEDURE get_square(num IN NUMBER, square OUT NUMBER) IS...
Show more...

Square is: 36

PL/SQL procedure successfully completed.

```

4. Procedure with DML (Insert into a table)

```

1  SET SERVEROUTPUT ON;
2  -- Create the table if not exists (handled safely)
3  BEGIN
4      EXECUTE IMMEDIATE '
5          CREATE TABLE students (
6              id NUMBER PRIMARY KEY,
7              name VARCHAR2(50)
8          )';
9  EXCEPTION
10     WHEN OTHERS THEN
11         IF SQLCODE = -955 THEN
12             NULL; --Table already exists
13         ELSE
14             RAISE;
15         END IF;
16 END;
17 /
18 DECLARE
19     PROCEDURE insert_student(sid IN NUMBER, sname IN VARCHAR2) IS
20     BEGIN
21         INSERT INTO students(id, name) VALUES (sid, sname);
22         DBMS_OUTPUT.PUT_LINE('Student inserted: ' || sname);
23     EXCEPTION
24         WHEN DUP_VAL_ON_INDEX THEN
25             DBMS_OUTPUT.PUT_LINE('Student ID already exists for ' || sname);
26     END;
27 BEGIN
28     insert_student(101, 'Amit');
29     insert_student(101, 'Ravi');
30 END;
31 /

```

Query result
Script output
DBMS output
Explain Plan
SQL history

```

SQL> DECLARE
    PROCEDURE insert_student(sid IN NUMBER, sname IN VARCHAR2) IS
    BEGIN
        INSERT INTO students(id, name) VALUES (sid, sname);...
Show more...

Student ID already exists for Anu
Student ID already exists for Ravi

PL/SQL procedure successfully completed.

```

5. Procedure with exception handling

```
1 DECLARE
2   PROCEDURE divide_numbers(a IN NUMBER, b IN NUMBER) IS
3     result NUMBER;
4   BEGIN
5     result := a / b;
6     DBMS_OUTPUT.PUT_LINE('Result: ' || result);
7   EXCEPTION
8     WHEN ZERO_DIVIDE THEN
9       DBMS_OUTPUT.PUT_LINE('Division by zero not allowed.');
```

Query result **Script output** DBMS output Explain Plan SQL history

result NUMBER;
BEGIN...

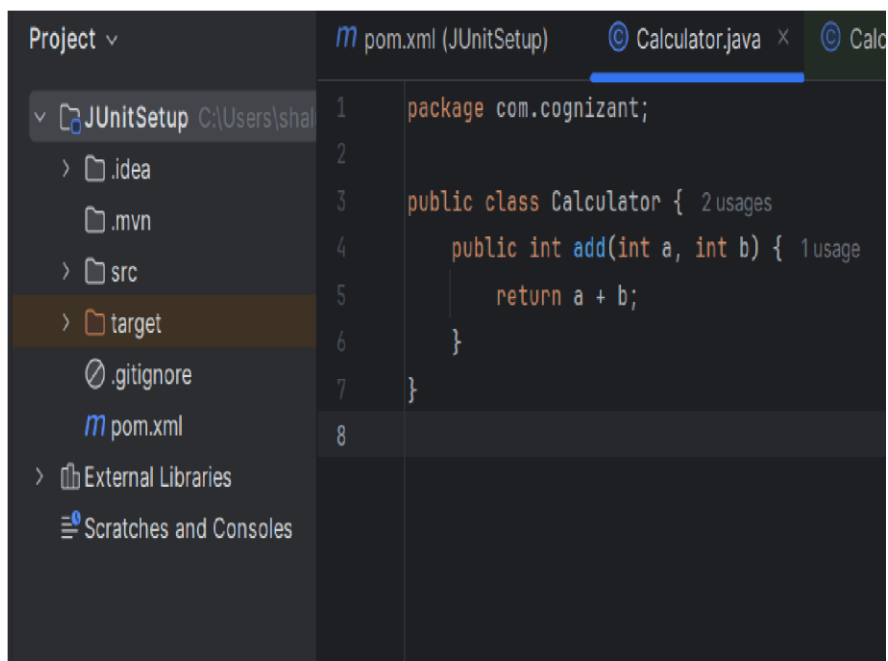
Show more...

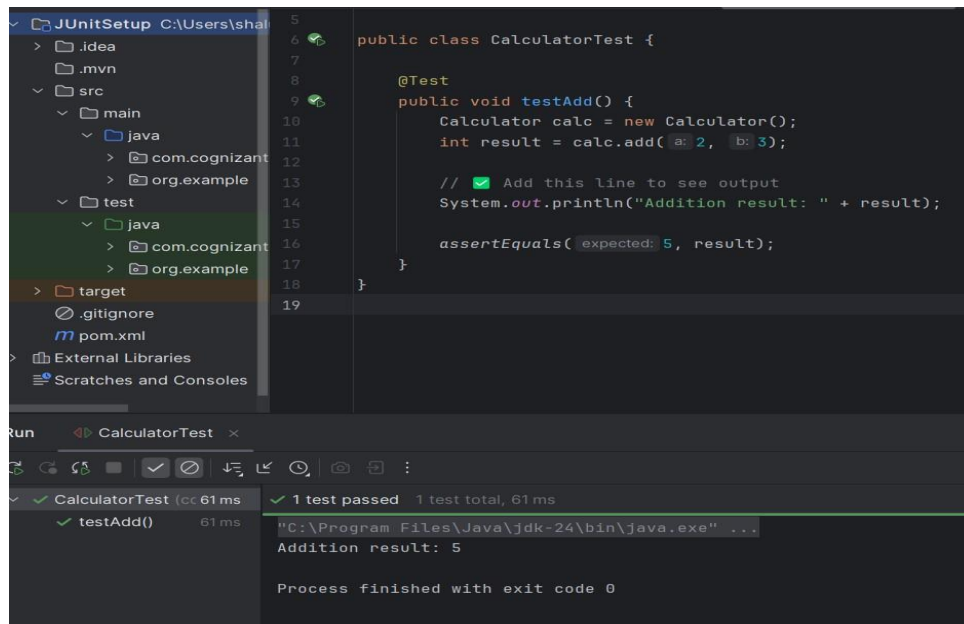
Result: 5
Division by zero not allowed.

PL/SQL procedure successfully completed.

TDD using JUnit5 and Mockito

Exercise 1: Setting Up JUnit

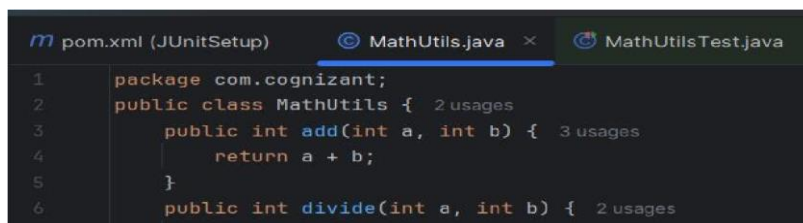




Exercise 2: Assertions in JUnit

Assertion, Use case

- assertEquals: Compare expected and actual values
- assertNotNull: ensure object is not null
- assertEquals: Compare arrays
- AssertThrows: Test if exception is thrown
- AssertTrue/False: Validate boolean conditions



```
m pom.xml (JUnitSetup)  MathUtils.java  MathUtilsTest.java x AppTest.java
1 package com.cognizant;
2 import org.junit.jupiter.api.Test;
3 import static org.junit.jupiter.api.Assertions.*;
4 public class MathUtilsTest {
5     MathUtils utils = new MathUtils(); 7 usages
6     @Test
7     void testAddition() {
8         int result = utils.add(a: 2, b: 3);
9         System.out.println("Addition Result: " + result);
10        assertEquals(expected: 5, result);
11    }
12    @Test
13    void testDivide() {
14        int result = utils.divide(a: 6, b: 3);
15        System.out.println("Division Result: " + result);
16        assertEquals(expected: 2, result);
17    }
18    @Test
19    void testGreeting() {
20        String greeting = utils.getGreeting();
21        System.out.println("Greeting: " + greeting);
22        assertNotNull(greeting);
23        assertEquals(expected: "Hello, JUnit!", greeting);
24    }
25    @Test
```

```
m pom.xml (JUnitSetup)  MathUtils.java  MathUtilsTest.java x AppTest.java
4 public class MathUtilsTest {
5     void testGreeting() {
6         System.out.println("Greeting: " + greeting);
7         assertNotNull(greeting);
8         assertEquals(expected: "Hello, JUnit!", greeting);
9     }
10    @Test
11    void testArrayEquality() {
12        int[] expected = {1, 2, 3};
13        int[] actual = utils.getArray();
14        System.out.println("Array Content: " + java.util.Arrays.toString(actual));
15        assertEquals(expected, actual);
16    }
17    @Test
18    void testException() {
19        System.out.println("Expecting ArithmeticException for divide by zero");
20        assertThrows(ArithmeticException.class, () -> utils.divide(a: 10, b: 0));
21    }
22    @Test
23    void testTrueCondition() {
24        boolean condition = utils.add(a: 2, b: 2) == 4;
25        System.out.println("True Condition Passed: " + condition);
26        assertTrue(condition);
27    }
28    @Test
29    void testFalseCondition() {
30        boolean condition = utils.add(a: 2, b: 2) == 5;
31        System.out.println("False Condition Passed: " + !condition);
32        assertFalse(condition);
33    }
34    @Test
35    void testNull() {
36        String greeting = utils.getGreeting();
37        assertNull(greeting);
38    }
39    @Test
40    void testNullArray() {
41        int[] array = utils.getArray();
42        assertNull(array);
43    }
44    @Test
45    void testNullString() {
46        String greeting = utils.getGreeting();
47        assertNull(greeting);
48    }
49    @Test
50    void testNullInteger() {
51        Integer result = utils.add(a: 2, b: 3);
52        assertNull(result);
53    }
54    @Test
55    void testNullBoolean() {
56        boolean condition = utils.add(a: 2, b: 2) == 4;
57        assertNull(condition);
58    }
59    @Test
60    void testNullDouble() {
61        Double result = utils.add(a: 2.0, b: 3.0);
62        assertNull(result);
63    }
64    @Test
65    void testNullFloat() {
66        Float result = utils.add(a: 2.0f, b: 3.0f);
67        assertNull(result);
68    }
69    @Test
70    void testNullLong() {
71        Long result = utils.add(a: 2L, b: 3L);
72        assertNull(result);
73    }
74    @Test
75    void testNullShort() {
76        Short result = utils.add(a: 2, b: 3);
77        assertNull(result);
78    }
79    @Test
80    void testNullByte() {
81        Byte result = utils.add(a: 2, b: 3);
82        assertNull(result);
83    }
84    @Test
85    void testNullCharacter() {
86        Character result = utils.add(a: 2, b: 3);
87        assertNull(result);
88    }
89    @Test
90    void testNullVoid() {
91        Void result = utils.add(a: 2, b: 3);
92        assertNull(result);
93    }
94    @Test
95    void testNullObject() {
96        Object result = utils.add(a: 2, b: 3);
97        assertNull(result);
98    }
99    @Test
100    void testNullStringArray() {
101        String[] array = utils.getArray();
102        assertNull(array);
103    }
104    @Test
105    void testNullIntegerArray() {
106        Integer[] array = utils.getArray();
107        assertNull(array);
108    }
109    @Test
110    void testNullBooleanArray() {
111        Boolean[] array = utils.getArray();
112        assertNull(array);
113    }
114    @Test
115    void testNullDoubleArray() {
116        Double[] array = utils.getArray();
117        assertNull(array);
118    }
119    @Test
120    void testNullFloatArray() {
121        Float[] array = utils.getArray();
122        assertNull(array);
123    }
124    @Test
125    void testNullLongArray() {
126        Long[] array = utils.getArray();
127        assertNull(array);
128    }
129    @Test
130    void testNullShortArray() {
131        Short[] array = utils.getArray();
132        assertNull(array);
133    }
134    @Test
135    void testNullByteArray() {
136        Byte[] array = utils.getArray();
137        assertNull(array);
138    }
139    @Test
140    void testNullCharacterArray() {
141        Character[] array = utils.getArray();
142        assertNull(array);
143    }
144    @Test
145    void testNullVoidArray() {
146        Void[] array = utils.getArray();
147        assertNull(array);
148    }
149    @Test
150    void testNullObjectArray() {
151        Object[] array = utils.getArray();
152        assertNull(array);
153    }
154    @Test
155    void testNullStringArray2() {
156        String[] array = utils.getArray();
157        assertNull(array);
158    }
159    @Test
160    void testNullIntegerArray2() {
161        Integer[] array = utils.getArray();
162        assertNull(array);
163    }
164    @Test
165    void testNullBooleanArray2() {
166        Boolean[] array = utils.getArray();
167        assertNull(array);
168    }
169    @Test
170    void testNullDoubleArray2() {
171        Double[] array = utils.getArray();
172        assertNull(array);
173    }
174    @Test
175    void testNullFloatArray2() {
176        Float[] array = utils.getArray();
177        assertNull(array);
178    }
179    @Test
180    void testNullLongArray2() {
181        Long[] array = utils.getArray();
182        assertNull(array);
183    }
184    @Test
185    void testNullShortArray2() {
186        Short[] array = utils.getArray();
187        assertNull(array);
188    }
189    @Test
190    void testNullByteArray2() {
191        Byte[] array = utils.getArray();
192        assertNull(array);
193    }
194    @Test
195    void testNullCharacterArray2() {
196        Character[] array = utils.getArray();
197        assertNull(array);
198    }
199    @Test
200    void testNullVoidArray2() {
201        Void[] array = utils.getArray();
202        assertNull(array);
203    }
204    @Test
205    void testNullObjectArray2() {
206        Object[] array = utils.getArray();
207        assertNull(array);
208    }
209    @Test
210    void testNullStringArray3() {
211        String[] array = utils.getArray();
212        assertNull(array);
213    }
214    @Test
215    void testNullIntegerArray3() {
216        Integer[] array = utils.getArray();
217        assertNull(array);
218    }
219    @Test
220    void testNullBooleanArray3() {
221        Boolean[] array = utils.getArray();
222        assertNull(array);
223    }
224    @Test
225    void testNullDoubleArray3() {
226        Double[] array = utils.getArray();
227        assertNull(array);
228    }
229    @Test
230    void testNullFloatArray3() {
231        Float[] array = utils.getArray();
232        assertNull(array);
233    }
234    @Test
235    void testNullLongArray3() {
236        Long[] array = utils.getArray();
237        assertNull(array);
238    }
239    @Test
240    void testNullShortArray3() {
241        Short[] array = utils.getArray();
242        assertNull(array);
243    }
244    @Test
245    void testNullByteArray3() {
246        Byte[] array = utils.getArray();
247        assertNull(array);
248    }
249    @Test
250    void testNullCharacterArray3() {
251        Character[] array = utils.getArray();
252        assertNull(array);
253    }
254    @Test
255    void testNullVoidArray3() {
256        Void[] array = utils.getArray();
257        assertNull(array);
258    }
259    @Test
260    void testNullObjectArray3() {
261        Object[] array = utils.getArray();
262        assertNull(array);
263    }
264    @Test
265    void testNullStringArray4() {
266        String[] array = utils.getArray();
267        assertNull(array);
268    }
269    @Test
270    void testNullIntegerArray4() {
271        Integer[] array = utils.getArray();
272        assertNull(array);
273    }
274    @Test
275    void testNullBooleanArray4() {
276        Boolean[] array = utils.getArray();
277        assertNull(array);
278    }
279    @Test
280    void testNullDoubleArray4() {
281        Double[] array = utils.getArray();
282        assertNull(array);
283    }
284    @Test
285    void testNullFloatArray4() {
286        Float[] array = utils.getArray();
287        assertNull(array);
288    }
289    @Test
290    void testNullLongArray4() {
291        Long[] array = utils.getArray();
292        assertNull(array);
293    }
294    @Test
295    void testNullShortArray4() {
296        Short[] array = utils.getArray();
297        assertNull(array);
298    }
299    @Test
300    void testNullByteArray4() {
301        Byte[] array = utils.getArray();
302        assertNull(array);
303    }
304    @Test
305    void testNullCharacterArray4() {
306        Character[] array = utils.getArray();
307        assertNull(array);
308    }
309    @Test
310    void testNullVoidArray4() {
311        Void[] array = utils.getArray();
312        assertNull(array);
313    }
314    @Test
315    void testNullObjectArray4() {
316        Object[] array = utils.getArray();
317        assertNull(array);
318    }
319    @Test
320    void testNullStringArray5() {
321        String[] array = utils.getArray();
322        assertNull(array);
323    }
324    @Test
325    void testNullIntegerArray5() {
326        Integer[] array = utils.getArray();
327        assertNull(array);
328    }
329    @Test
330    void testNullBooleanArray5() {
331        Boolean[] array = utils.getArray();
332        assertNull(array);
333    }
334    @Test
335    void testNullDoubleArray5() {
336        Double[] array = utils.getArray();
337        assertNull(array);
338    }
339    @Test
340    void testNullFloatArray5() {
341        Float[] array = utils.getArray();
342        assertNull(array);
343    }
344    @Test
345    void testNullLongArray5() {
346        Long[] array = utils.getArray();
347        assertNull(array);
348    }
349    @Test
350    void testNullShortArray5() {
351        Short[] array = utils.getArray();
352        assertNull(array);
353    }
354    @Test
355    void testNullByteArray5() {
356        Byte[] array = utils.getArray();
357        assertNull(array);
358    }
359    @Test
360    void testNullCharacterArray5() {
361        Character[] array = utils.getArray();
362        assertNull(array);
363    }
364    @Test
365    void testNullVoidArray5() {
366        Void[] array = utils.getArray();
367        assertNull(array);
368    }
369    @Test
370    void testNullObjectArray5() {
371        Object[] array = utils.getArray();
372        assertNull(array);
373    }
374    @Test
375    void testNullStringArray6() {
376        String[] array = utils.getArray();
377        assertNull(array);
378    }
379    @Test
380    void testNullIntegerArray6() {
381        Integer[] array = utils.getArray();
382        assertNull(array);
383    }
384    @Test
385    void testNullBooleanArray6() {
386        Boolean[] array = utils.getArray();
387        assertNull(array);
388    }
389    @Test
390    void testNullDoubleArray6() {
391        Double[] array = utils.getArray();
392        assertNull(array);
393    }
394    @Test
395    void testNullFloatArray6() {
396        Float[] array = utils.getArray();
397        assertNull(array);
398    }
399    @Test
400    void testNullLongArray6() {
401        Long[] array = utils.getArray();
402        assertNull(array);
403    }
404    @Test
405    void testNullShortArray6() {
406        Short[] array = utils.getArray();
407        assertNull(array);
408    }
409    @Test
410    void testNullByteArray6() {
411        Byte[] array = utils.getArray();
412        assertNull(array);
413    }
414    @Test
415    void testNullCharacterArray6() {
416        Character[] array = utils.getArray();
417        assertNull(array);
418    }
419    @Test
420    void testNullVoidArray6() {
421        Void[] array = utils.getArray();
422        assertNull(array);
423    }
424    @Test
425    void testNullObjectArray6() {
426        Object[] array = utils.getArray();
427        assertNull(array);
428    }
429    @Test
430    void testNullStringArray7() {
431        String[] array = utils.getArray();
432        assertNull(array);
433    }
434    @Test
435    void testNullIntegerArray7() {
436        Integer[] array = utils.getArray();
437        assertNull(array);
438    }
439    @Test
440    void testNullBooleanArray7() {
441        Boolean[] array = utils.getArray();
442        assertNull(array);
443    }
444    @Test
445    void testNullDoubleArray7() {
446        Double[] array = utils.getArray();
447        assertNull(array);
448    }
449    @Test
450    void testNullFloatArray7() {
451        Float[] array = utils.getArray();
452        assertNull(array);
453    }
454    @Test
455    void testNullLongArray7() {
456        Long[] array = utils.getArray();
457        assertNull(array);
458    }
459    @Test
460    void testNullShortArray7() {
461        Short[] array = utils.getArray();
462        assertNull(array);
463    }
464    @Test
465    void testNullByteArray7() {
466        Byte[] array = utils.getArray();
467        assertNull(array);
468    }
469    @Test
470    void testNullCharacterArray7() {
471        Character[] array = utils.getArray();
472        assertNull(array);
473    }
474    @Test
475    void testNullVoidArray7() {
476        Void[] array = utils.getArray();
477        assertNull(array);
478    }
479    @Test
480    void testNullObjectArray7() {
481        Object[] array = utils.getArray();
482        assertNull(array);
483    }
484    @Test
485    void testNullStringArray8() {
486        String[] array = utils.getArray();
487        assertNull(array);
488    }
489    @Test
490    void testNullIntegerArray8() {
491        Integer[] array = utils.getArray();
492        assertNull(array);
493    }
494    @Test
495    void testNullBooleanArray8() {
496        Boolean[] array = utils.getArray();
497        assertNull(array);
498    }
499    @Test
500    void testNullDoubleArray8() {
501        Double[] array = utils.getArray();
502        assertNull(array);
503    }
504    @Test
505    void testNullFloatArray8() {
506        Float[] array = utils.getArray();
507        assertNull(array);
508    }
509    @Test
510    void testNullLongArray8() {
511        Long[] array = utils.getArray();
512        assertNull(array);
513    }
514    @Test
515    void testNullShortArray8() {
516        Short[] array = utils.getArray();
517        assertNull(array);
518    }
519    @Test
520    void testNullByteArray8() {
521        Byte[] array = utils.getArray();
522        assertNull(array);
523    }
524    @Test
525    void testNullCharacterArray8() {
526        Character[] array = utils.getArray();
527        assertNull(array);
528    }
529    @Test
530    void testNullVoidArray8() {
531        Void[] array = utils.getArray();
532        assertNull(array);
533    }
534    @Test
535    void testNullObjectArray8() {
536        Object[] array = utils.getArray();
537        assertNull(array);
538    }
539    @Test
540    void testNullStringArray9() {
541        String[] array = utils.getArray();
542        assertNull(array);
543    }
544    @Test
545    void testNullIntegerArray9() {
546        Integer[] array = utils.getArray();
547        assertNull(array);
548    }
549    @Test
550    void testNullBooleanArray9() {
551        Boolean[] array = utils.getArray();
552        assertNull(array);
553    }
554    @Test
555    void testNullDoubleArray9() {
556        Double[] array = utils.getArray();
557        assertNull(array);
558    }
559    @Test
560    void testNullFloatArray9() {
561        Float[] array = utils.getArray();
562        assertNull(array);
563    }
564    @Test
565    void testNullLongArray9() {
566        Long[] array = utils.getArray();
567        assertNull(array);
568    }
569    @Test
570    void testNullShortArray9() {
571        Short[] array = utils.getArray();
572        assertNull(array);
573    }
574    @Test
575    void testNullByteArray9() {
576        Byte[] array = utils.getArray();
577        assertNull(array);
578    }
579    @Test
580    void testNullCharacterArray9() {
581        Character[] array = utils.getArray();
582        assertNull(array);
583    }
584    @Test
585    void testNullVoidArray9() {
586        Void[] array = utils.getArray();
587        assertNull(array);
588    }
589    @Test
590    void testNullObjectArray9() {
591        Object[] array = utils.getArray();
592        assertNull(array);
593    }
594    @Test
595    void testNullStringArray10() {
596        String[] array = utils.getArray();
597        assertNull(array);
598    }
599    @Test
600    void testNullIntegerArray10() {
601        Integer[] array = utils.getArray();
602        assertNull(array);
603    }
604    @Test
605    void testNullBooleanArray10() {
606        Boolean[] array = utils.getArray();
607        assertNull(array);
608    }
609    @Test
610    void testNullDoubleArray10() {
611        Double[] array = utils.getArray();
612        assertNull(array);
613    }
614    @Test
615    void testNullFloatArray10() {
616        Float[] array = utils.getArray();
617        assertNull(array);
618    }
619    @Test
620    void testNullLongArray10() {
621        Long[] array = utils.getArray();
622        assertNull(array);
623    }
624    @Test
625    void testNullShortArray10() {
626        Short[] array = utils.getArray();
627        assertNull(array);
628    }
629    @Test
630    void testNullByteArray10() {
631        Byte[] array = utils.getArray();
632        assertNull(array);
633    }
634    @Test
635    void testNullCharacterArray10() {
636        Character[] array = utils.getArray();
637        assertNull(array);
638    }
639    @Test
640    void testNullVoidArray10() {
641        Void[] array = utils.getArray();
642        assertNull(array);
643    }
644    @Test
645    void testNullObjectArray10() {
646        Object[] array = utils.getArray();
647        assertNull(array);
648    }
649    @Test
650    void testNullStringArray11() {
651        String[] array = utils.getArray();
652        assertNull(array);
653    }
654    @Test
655    void testNullIntegerArray11() {
656        Integer[] array = utils.getArray();
657        assertNull(array);
658    }
659    @Test
660    void testNullBooleanArray11() {
661        Boolean[] array = utils.getArray();
662        assertNull(array);
663    }
664    @Test
665    void testNullDoubleArray11() {
666        Double[] array = utils.getArray();
667        assertNull(array);
668    }
669    @Test
670    void testNullFloatArray11() {
671        Float[] array = utils.getArray();
672        assertNull(array);
673    }
674    @Test
675    void testNullLongArray11() {
676        Long[] array = utils.getArray();
677        assertNull(array);
678    }
679    @Test
680    void testNullShortArray11() {
681        Short[] array = utils.getArray();
682        assertNull(array);
683    }
684    @Test
685    void testNullByteArray11() {
686        Byte[] array = utils.getArray();
687        assertNull(array);
688    }
689    @Test
690    void testNullCharacterArray11() {
691        Character[] array = utils.getArray();
692        assertNull(array);
693    }
694    @Test
695    void testNullVoidArray11() {
696        Void[] array = utils.getArray();
697        assertNull(array);
698    }
699    @Test
700    void testNullObjectArray11() {
701        Object[] array = utils.getArray();
702        assertNull(array);
703    }
704    @Test
705    void testNullStringArray12() {
706        String[] array = utils.getArray();
707        assertNull(array);
708    }
709    @Test
710    void testNullIntegerArray12() {
711        Integer[] array = utils.getArray();
712        assertNull(array);
713    }
714    @Test
715    void testNullBooleanArray12() {
716        Boolean[] array = utils.getArray();
717        assertNull(array);
718    }
719    @Test
720    void testNullDoubleArray12() {
721        Double[] array = utils.getArray();
722        assertNull(array);
723    }
724    @Test
725    void testNullFloatArray12() {
726        Float[] array = utils.getArray();
727        assertNull(array);
728    }
729    @Test
730    void testNullLongArray12() {
731        Long[] array = utils.getArray();
732        assertNull(array);
733    }
734    @Test
735    void testNullShortArray12() {
736        Short[] array = utils.getArray();
737        assertNull(array);
738    }
739    @Test
740    void testNullByteArray12() {
741        Byte[] array = utils.getArray();
742        assertNull(array);
743    }
744    @Test
745    void testNullCharacterArray12() {
746        Character[] array = utils.getArray();
747        assertNull(array);
748    }
749    @Test
750    void testNullVoidArray12() {
751        Void[] array = utils.getArray();
752        assertNull(array);
753    }
754    @Test
755    void testNullObjectArray12() {
756        Object[] array = utils.getArray();
757        assertNull(array);
758    }
759    @Test
760    void testNullStringArray13() {
761        String[] array = utils.getArray();
762        assertNull(array);
763    }
764    @Test
765    void testNullIntegerArray13() {
766        Integer[] array = utils.getArray();
767        assertNull(array);
768    }
769    @Test
770    void testNullBooleanArray13() {
771        Boolean[] array = utils.getArray();
772        assertNull(array);
773    }
774    @Test
775    void testNullDoubleArray13() {
776        Double[] array = utils.getArray();
777        assertNull(array);
778    }
779    @Test
780    void testNullFloatArray13() {
781        Float[] array = utils.getArray();
782        assertNull(array);
783    }
784    @Test
785    void testNullLongArray13() {
786        Long[] array = utils.getArray();
787        assertNull(array);
788    }
789    @Test
790    void testNullShortArray13() {
791        Short[] array = utils.getArray();
792        assertNull(array);
793    }
794    @Test
795    void testNullByteArray13() {
796        Byte[] array = utils.getArray();
797        assertNull(array);
798    }
799    @Test
800    void testNullCharacterArray13() {
801        Character[] array = utils.getArray();
802        assertNull(array);
803    }
804    @Test
805    void testNullVoidArray13() {
806        Void[] array = utils.getArray();
807        assertNull(array);
808    }
809    @Test
810    void testNullObjectArray13() {
811        Object[] array = utils.getArray();
812        assertNull(array);
813    }
814    @Test
815    void testNullStringArray14() {
816        String[] array = utils.getArray();
817        assertNull(array);
818    }
819    @Test
820    void testNullIntegerArray14() {
821        Integer[] array = utils.getArray();
822        assertNull(array);
823    }
824    @Test
825    void testNullBooleanArray14() {
826        Boolean[] array = utils.getArray();
827        assertNull(array);
828    }
829    @Test
830    void testNullDoubleArray14() {
831        Double[] array = utils.getArray();
832        assertNull(array);
833    }
834    @Test
835    void testNullFloatArray14() {
836        Float[] array = utils.getArray();
837        assertNull(array);
838    }
839    @Test
840    void testNullLongArray14() {
841        Long[] array = utils.getArray();
842        assertNull(array);
843    }
844    @Test
845    void testNullShortArray14() {
846        Short[] array = utils.getArray();
847        assertNull(array);
848    }
849    @Test
850    void testNullByteArray14() {
851        Byte[] array = utils.getArray();
852        assertNull(array);
853    }
854    @Test
855    void testNullCharacterArray14() {
856        Character[] array = utils.getArray();
857        assertNull(array);
858    }
859    @Test
860    void testNullVoidArray14() {
861        Void[] array = utils.getArray();
862        assertNull(array);
863    }
864    @Test
865    void testNullObjectArray14() {
866        Object[] array = utils.getArray();
867        assertNull(array);
868    }
869    @Test
870    void testNullStringArray15() {
871        String[] array = utils.getArray();
872        assertNull(array);
873    }
874    @Test
875    void testNullIntegerArray15() {
876        Integer[] array = utils.getArray();
877        assertNull(array);
878    }
879    @Test
880    void testNullBooleanArray15() {
881        Boolean[] array = utils.getArray();
882        assertNull(array);
883    }
884    @Test
885    void testNullDoubleArray15() {
886        Double[] array = utils.getArray();
887        assertNull(array);
888    }
889    @Test
890    void testNullFloatArray15() {
891        Float[] array = utils.getArray();
892        assertNull(array);
893    }
894    @Test
895    void testNullLongArray15() {
896        Long[] array = utils.getArray();
897        assertNull(array);
898    }
899    @Test
900    void testNullShortArray15() {
901        Short[] array = utils.getArray();
902        assertNull(array);
903    }
904    @Test
905    void testNullByteArray15() {
906        Byte[] array = utils.getArray();
907        assertNull(array);
908    }
909    @Test
910    void testNullCharacterArray15() {
911        Character[] array = utils.getArray();
912        assertNull(array);
913    }
914    @Test
915    void testNullVoidArray15() {
916        Void[] array = utils.getArray();
917        assertNull(array);
918    }
919    @Test
920    void testNullObjectArray15() {
921        Object[] array = utils.getArray();
922        assertNull(array);
923    }
924    @Test
925    void testNullStringArray16() {
926        String[] array = utils.getArray();
927        assertNull(array);
928    }
929    @Test
930    void testNullIntegerArray16() {
931        Integer[] array = utils.getArray();
932        assertNull(array);
933    }
934    @Test
935    void testNullBooleanArray16() {
936        Boolean[] array = utils.getArray();
937        assertNull(array);
938    }
939    @Test
940    void testNullDoubleArray16() {
941        Double[] array = utils.getArray();
942        assertNull(array);
943    }
944    @Test
945    void testNullFloatArray16() {
946        Float[] array = utils.getArray();
947        assertNull(array);
948    }
949    @Test
950    void testNullLongArray16() {
951        Long[] array = utils.getArray();
952        assertNull(array);
953    }
954    @Test
955    void testNullShortArray16() {
956        Short[] array = utils.getArray();
957        assertNull(array);
958    }
959    @Test
960    void testNullByteArray16() {
961        Byte[] array = utils.getArray();
962        assertNull(array);
963    }
964    @Test
965    void testNullCharacterArray16() {
966        Character[] array = utils.getArray();
967        assertNull(array);
968    }
969    @Test
970    void testNullVoidArray16() {
971        Void[] array = utils.getArray();
972        assertNull(array);
973    }
974    @Test
975    void testNullObjectArray16() {
976        Object[] array = utils.getArray();
977        assertNull(array);
978    }
979    @Test
980    void testNullStringArray17() {
981        String[] array = utils.getArray();
982        assertNull(array);
983    }
984    @Test
985    void testNullIntegerArray17() {
986        Integer[] array = utils.getArray();
987        assertNull(array);
988    }
989    @Test
990    void testNullBooleanArray17() {
991        Boolean[] array = utils.getArray();
992        assertNull(array);
993    }
994    @Test
995    void testNullDoubleArray17() {
996        Double[] array = utils.getArray();
997        assertNull(array);
998    }
999    @Test
1000    void testNullFloatArray17() {
1001        Float[] array = utils.getArray();
1002        assertNull(array);
1003    }
1004    @Test
1005    void testNullLongArray17() {
1006        Long[] array = utils.getArray();
1007        assertNull(array);
1008    }
1009    @Test
1010    void testNullShortArray17() {
1011        Short[] array = utils.getArray();
1012        assertNull(array);
1013    }
1014    @Test
1015    void testNullByteArray17() {
1016        Byte[] array = utils.getArray();
1017        assertNull(array);
1018    }
1019    @Test
1020    void testNullCharacterArray17() {
1021        Character[] array = utils.getArray();
1022        assertNull(array);
1023    }
1024    @Test
1025    void testNullVoidArray17() {
1026        Void[] array = utils.getArray();
1027        assertNull(array);
1028    }
1029    @Test
1030    void testNullObjectArray17() {
1031        Object[] array = utils.getArray();
1032        assertNull(array);
1033    }
1034    @Test
1035    void testNullStringArray18() {
1036        String[] array = utils.getArray();
1037        assertNull(array);
1038    }
1039    @Test
1040    void testNullIntegerArray18() {
1041        Integer[] array = utils.getArray();
1042        assertNull(array);
1043    }
1044    @Test
1045    void testNullBooleanArray18() {
1046        Boolean[] array = utils.getArray();
1047        assertNull(array);
1048    }
1049    @Test
1050    void testNullDoubleArray18() {
1051        Double[] array = utils.getArray();
1052        assertNull(array);
1053    }
1054    @Test
1055    void testNullFloatArray18() {
1056        Float[] array = utils.getArray();
1057        assertNull(array);
1058    }
1059    @Test
1060    void testNullLongArray18() {
1061        Long[] array = utils.getArray();
1062        assertNull(array);
1063    }
1064    @Test
1065    void testNullShortArray18() {
1066        Short[] array = utils.getArray();
1067        assertNull(array);
1068    }
1069    @Test
1070    void testNullByteArray18() {
1071        Byte[] array = utils.getArray();
1072        assertNull(array);
1073    }
1074    @Test
1075    void testNullCharacterArray18() {
1076        Character[] array = utils.getArray();
1077        assertNull(array);
1078    }
1079    @Test
1080    void testNullVoidArray18() {
1081        Void[] array = utils.getArray();
1082        assertNull(array);
1083    }
1084    @Test
1085    void testNullObjectArray18() {
1086        Object[] array = utils.getArray();
1087        assertNull(array);
1088    }
1089    @Test
1090    void testNullStringArray19() {
1091        String[] array = utils.getArray();
1092        assertNull(array);
1093    }
1094    @Test
1095    void testNullIntegerArray19() {
1096        Integer[] array = utils.getArray();
1097        assertNull(array);
1098    }
1099    @Test
1100    void testNullBooleanArray19() {
1101        Boolean[] array = utils.getArray();
1102        assertNull(array);
1103    }
1104    @Test
1105    void testNullDoubleArray19() {
1106        Double[] array = utils.getArray();
1107        assertNull(array);
1108    }
1109    @Test
1110    void testNullFloatArray19() {
1111        Float[] array = utils.getArray();
1112        assertNull(array);
1113    }
1114    @Test
1115    void testNullLongArray19() {
1116        Long[] array = utils.getArray();
1117        assertNull(array);
1118    }
1119    @Test
1120    void testNullShortArray19() {
1121        Short[] array = utils.getArray();
1122        assertNull(array);
1123    }
1124    @Test
1125    void testNullByteArray19() {
1126        Byte[] array = utils.getArray();
1127        assertNull(array);
1128    }
1129    @Test
1130    void testNullCharacterArray19() {
1131        Character[] array = utils.getArray();
1132        assertNull(array);
1133    }
1134    @Test
1135    void testNullVoidArray19() {
1136        Void[] array = utils.getArray();
1137        assertNull(array);
1138    }
1139    @Test
1140    void testNullObjectArray19() {
1141        Object[] array = utils.getArray();
1142        assertNull(array);
1143    }
1144    @Test
1145    void testNullStringArray20() {
1146        String[] array = utils.getArray();
1147        assertNull(array);
1148    }
1149    @Test
1150    void testNullIntegerArray20() {
1151        Integer[] array = utils.getArray();
1152        assertNull(array);
1153    }
1154    @Test
1155    void testNullBooleanArray20() {
1156        Boolean[] array = utils.getArray();
1157        assertNull(array);
1158    }
1159    @Test
1160    void testNullDoubleArray20() {
1161        Double[] array = utils.getArray();
1162        assertNull(array);
1163    }
1164    @Test
1165    void testNullFloatArray20() {
1166        Float[] array = utils.getArray();
1167        assertNull(array);
1168    }
1169    @Test
1170    void testNullLongArray20() {
1171        Long[] array = utils.getArray();
1172        assertNull(array);
1173    }
1174    @Test
1175    void testNullShortArray20() {
1176        Short[] array = utils.getArray();
1177        assertNull(array);
1178    }
1179    @Test
1180    void testNullByteArray20() {
1181        Byte[] array = utils.getArray();
1182        assertNull(array);
1183    }
1184    @Test
1185    void testNullCharacterArray20() {
1186        Character[] array = utils.getArray();
1187        assertNull(array);
1188    }
1189    @Test
1190    void testNullVoidArray20() {
1191        Void[] array = utils.getArray();
1192        assertNull(array);
1193    }
1194    @Test
1195    void testNullObjectArray20() {
1196        Object[] array = utils.getArray();
1197        assertNull(array);
1198    }
1199    @Test
1200    void testNullStringArray21() {
1201        String[] array = utils.getArray();
1202        assertNull(array);
1203    }
1204    @Test
1205    void testNullIntegerArray21() {
1206        Integer[] array = utils.getArray();
1207        assertNull(array);
1208    }
1209    @Test
1210    void testNullBooleanArray21() {
1211        Boolean[] array = utils.getArray();
1212        assertNull(array);
1213    }
1214    @Test
1215    void testNullDoubleArray21() {
1216        Double[] array = utils.getArray();
1217        assertNull(array);
1218    }
1219    @Test
1220    void testNullFloatArray21() {
1221        Float[] array = utils.getArray();
1222        assertNull(array);
1223    }
1224    @Test
1225    void testNullLongArray21() {
1226        Long[] array = utils.getArray();
1227        assertNull(array);
1228    }
1229    @Test
1230    void testNullShortArray21() {
1231        Short[] array = utils.getArray();
1232        assertNull(array);
1233    }
1234    @Test
1235    void testNullByteArray21() {
1236        Byte[] array = utils.getArray();
1237        assertNull(array);
1238    }
1239    @Test
1240    void testNullCharacterArray21() {
124
```


Exercise 3: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods

The **Arrange-Act-Assert (AAA)** Pattern helps organize each test case into three clear steps: first, you arrange or set up the necessary data and environment; second, you act by calling the method under test; and finally, you assert the expected outcome. A test fixture refers to reusable setup code, such as creating a Bank Account object before each test, which ensures consistency and avoids duplication. The **@BeforeEach** annotation is used to execute setup code before each test method runs, making sure every test starts with a clean state. Similarly, **@AfterEach** is used to define any cleanup or logging that should happen after each test completes. These practices together make your tests more readable, maintainable, and reliable.

```
m pom.xml (JUnitSetup)  BankAccount.java  BankAccountTest.java x  AppTest.java
1  package com.cognizant;
2  import org.junit.jupiter.api.*;
3  import static org.junit.jupiter.api.Assertions.*;
4  public class BankAccountTest {
5      BankAccount account; 7 usages
6      @BeforeEach
7      void setUp() {
8          account = new BankAccount(); // Arrange (Fixture)
9          System.out.println("Setup complete.");
10     }
11     @AfterEach
12     void tearDown() {
13         System.out.println("Test complete.\n");
14     }
15     @Test
16     void testDeposit() {
17         account.deposit(100); // Act
18         assertEquals(100, account.getBalance()); // Assert
19     }
20     @Test
21     void testWithdraw() {
22         account.deposit(200);
23         account.withdraw(50); // Act
24         assertEquals(150, account.getBalance()); //Assert
25     }
26     @Test
27     void testWithdrawThrows() {
28         assertThrows(IllegalArgumentException.class, () -> account.withdraw(500)); //Assert
29     }
30 }
```

```
m pom.xml (JUnitSetup)  BankAccount.java x  BankAccountTest.java  AppTest.java
1  package com.cognizant;
2
3  public class BankAccount { no usages
4      private double balance = 0; 4 usages
5
6      public void deposit(double amount) { balance += amount; } no usages
7
8      public void withdraw(double amount) { no usages
9          if (amount > balance) throw new IllegalArgumentException("Insufficient funds");
10         balance -= amount;
11     }
12
13     public double getBalance() { return balance; } no usages
14 }
```



```
✓ 3 tests passed 3 tests total, 70 ms
"C:\Program Files\Java\jdk-24\bin\j
Setup complete.
Test complete.

Setup complete.
Test complete.

Setup complete.
Test complete.

Process finished with exit code 0
```

Mockito exercises

Exercise 1: Mocking and Stubbing

```
m pom.xml (MockitoExample) x UserRepository.java UserSevice.java UserSeviceTest.java AppTest.java
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4
5   <groupId>org.example</groupId>
6   <artifactId>MockitoExample</artifactId>
7   <version>1.0-SNAPSHOT</version>
8   <packaging>jar</packaging>
9
10  <name>MockitoExample</name>
11  <url>http://maven.apache.org</url>
12
13  <properties>
14    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15  </properties>
16
17  <dependencies>
18    <!-- JUnit 5 -->
19    <dependency>
20      <groupId>org.junit.jupiter</groupId>
21      <artifactId>junit-jupiter</artifactId>
22      <version>5.10.0</version>
23      <scope>test</scope>
24    </dependency>
25    <!-- Mockito -->
26    <dependency>
27      <groupId>org.mockito</groupId>
28      <artifactId>mockito-core</artifactId>
29      <version>5.11.0</version>
30      <scope>test</scope>
31    </dependency>
32  </dependencies>
33
34 </project>
```

```
m pom.xml (MockitoExample) x UserRepository.java x
1 package com.cognizant;
2
3 public interface UserRepository { no usages
4     String getUsernameById(int id); no usages
5 }
6
```

```

m pom.xml (MockitoExample)  © UserRepository.java  © UserService.java
1  package com.cognizant;
2  public class UserService {  2 usages
3      private UserRepository repository;  2 usages
4      public UserService(UserRepository repository) {  1 usage
5          this.repository = repository; }
6      public String getFormattedUsername(int id) {  1 usage
7          String name = repository.getUsernameById(id);
8          return "User: " + name; }
9  }

```

```

m pom.xml (MockitoExample)  © UserRepository.java  © UserService.java  © UserSer
1  package com.cognizant;
2
3  import org.junit.jupiter.api.Test;
4  import org.mockito.Mockito;
5  import static org.junit.jupiter.api.Assertions.assertEquals;
6  public class UserServiceTest {
7      @Test
8      void testGetFormattedUsername() {
9          UserRepository mockRepo = Mockito.mock(UserRepository.class);
10         Mockito.when(mockRepo.getUsernameById(1)).thenReturn("Shalu");
11         // Inject mock into service
12         UserService service = new UserService(mockRepo);
13         // Act
14         String result = service.getFormattedUsername(1);
15         // Assert
16         assertEquals("User: Shalu", result);
17         System.out.println("Test passed with result: " + result);
18     }

```

```
Test passed with result: User: Shalu
```

```
Process finished with exit code 0
```

Exercise 2: Verifying Interactions

In Mockito, verifying interactions helps ensure that specific methods were called (or not called) on mocked dependencies. This is crucial when testing behavior — especially in scenarios like sending notifications, updating databases, or logging. For instance, after a login() method is called, we may want to confirm that a notification service's send() method was triggered exactly once. Mockito provides verify() for this purpose, and it also allows checking that no more or unwanted interactions occurred using methods like verifyNoMoreInteractions() or verifyNever().

Example: Verifying Login Notification with Mockito

```
m pom.xml (MockitoExample) × © NotificationService.java × © U
1 package com.cognizant;
2
3 public class NotificationService { no usages
4     public void send(String message) { no usages
5         System.out.println("Sending: " + message);
6     }
7 }
```

```
m pom.xml (MockitoExample) © NotificationService.java © User.java
1 package com.cognizant;
2 public class User { 4 usages
3     private NotificationService notificationService; 2 usages
4
5     public User(NotificationService notificationService) {
6         this.notificationService = notificationService;
7     }
8     public void login() { 2 usages
9         notificationService.send("Login Successful");
10    }
11 }
```

```
m pom.xml (MockitoExample) © NotificationService.java UserTest.java × AppTest
1 package com.cognizant;
2
3 import org.junit.jupiter.api.Test;
4 import static org.mockito.Mockito.*;
5 public class UserTest {
6     @Test
7     public void testLoginSendsNotification() {
8         NotificationService mockService = mock(NotificationService.class);
9         User user = new User(mockService);
10        user.login();
11        // Test file
12        verify(mockService).send("Login Successful"); }
13
14    @Test
15    public void testLoginDoesNotSendOtherMessage() {
16        NotificationService mockService = mock(NotificationService.class);
17        User user = new User(mockService);
18        user.login();
19        verify(mockService, never()).send("Wrong Message");
20    }
21 }
```

SL4J Logging exercises

Exercise 3: Logging Error Messages and Warning Levels

```
pom.xml (SLF4JExampleProject)  LoggerExample.java  AppTest.java
1  package com.cognizant;
2
3  import org.slf4j.Logger;
4  import org.slf4j.LoggerFactory;
5  public class LoggerExample {
6      private static final Logger logger = LoggerFactory.getLogger(LoggerExample.class);
7      public void performTask(int input) { 3 usages
8          logger.info("Task started");
9          if (input < 0) {
10             logger.warn("Negative input provided: {}", input);
11          }
12          try {
13             int result = 10 / input;
14             logger.info("Computation result: {}", result);
15          } catch (ArithmeticException e) {
16             logger.error("Error occurred while computing: Division by zero", e);
17          }
18          logger.info("Task completed");
19      }
20
21      public static void main(String[] args) {
22          LoggerExample example = new LoggerExample();
23          example.performTask(input: 5); // Normal
24          example.performTask(input: 0); // Will trigger error
25          example.performTask(input: -2); // Will trigger warning
26      }
27  }
```

```
[main] INFO com.cognizant.LoggerExample - Task started
[main] INFO com.cognizant.LoggerExample - Computation result: 2
[main] INFO com.cognizant.LoggerExample - Task completed
[main] INFO com.cognizant.LoggerExample - Task started
[main] ERROR com.cognizant.LoggerExample - Error occurred while computing:
Division by zero
java.lang.ArithmeticException Create breakpoint: / by zero
    at com.cognizant.LoggerExample.performTask(LoggerExample.java:18)
    at com.cognizant.LoggerExample.main(LoggerExample.java:30)
[main] INFO com.cognizant.LoggerExample - Task completed
[main] INFO com.cognizant.LoggerExample - Task started
[main] WARN com.cognizant.LoggerExample - Negative input provided: -2
[main] INFO com.cognizant.LoggerExample - Computation result: -5
[main] INFO com.cognizant.LoggerExample - Task completed

Process finished with exit code 0
```