

## WEEK 2

### JUnit, Mockito and SL4J

#### 3. Setting Up JUnit

Created a new Java project in your IDE – Week2

Added JUnit dependency to project

pom.xml:

```
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.13.2</version>
<scope>test</scope>
</dependency>
```

Created a class `Calculator.java`:

```
package week2;
```

```
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
    public int subtract(int a, int b) {
        return a - b;
    }
    public int multiply(int a, int b) {
        return a * b;
    }
    public int divide(int a, int b) {
        if (b == 0) throw new IllegalArgumentException("Cannot divide by zero");
        return a / b;
    }
}
```

Created CalculatorTest.java to Test :

```
package week2;
```

```
import static org.junit.Assert.assertEquals;
```

```
import static org.junit.Assert.fail;
```

```
import org.junit.Test;
```

```
public class CalculatorTest {
```

```
    @Test
```

```
    public void test() {
```

```
        fail("Not yet implemented");
```

```
    }
```

```
    Calculator calc = new Calculator();
```

```
    @Test
```

```
    public void testAdd() {
```

```
        assertEquals(5, calc.add(2, 3));
```

```
    }
```

```
    @Test
```

```
    public void testSubtract() {
```

```
        assertEquals(1, calc.subtract(4, 3));
```

```
    }
```

```
    @Test
```

```
    public void testMultiply() {
```

```

    assertEquals(12, calc.multiply(3, 4));
}

```

@Test

```

public void testDivide() {
    assertEquals(5, calc.divide(10, 2));
}

```

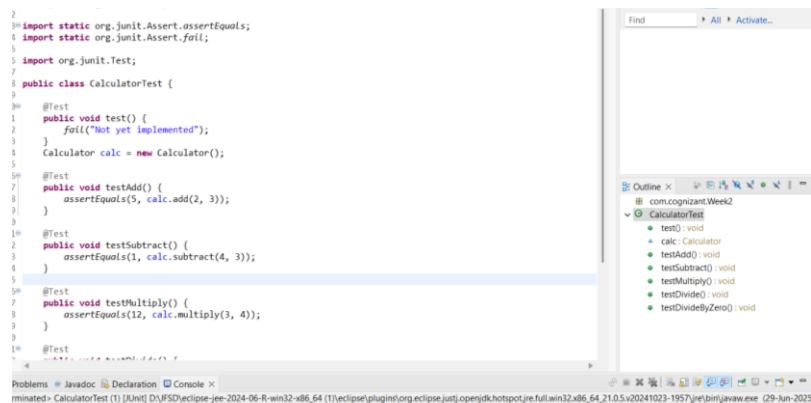
@Test(expected = IllegalArgumentException.class)

```

public void testDivideByZero() {
    calc.divide(10, 0);
}

```

Output:



#### 4. Assertions in Junit:-

Created a test file named with AssertionsTest.java.

```
package week2;
```

```
import org.junit.Test;
```

```
import static org.junit.Assert.*;
```

```
public class AssertionsTest {
```

```
    @Test
```

```
    public void testAssertions() {
```

```
        // Assert equals
```

```
        assertEquals(5, 2 + 3);
```

```
        // Assert true
```

```
        assertTrue(5 > 3);
```

```
        // Assert false
```

```
        assertFalse(5 < 3);
```

```
        // Assert null
```

```
        assertNull(null);
```

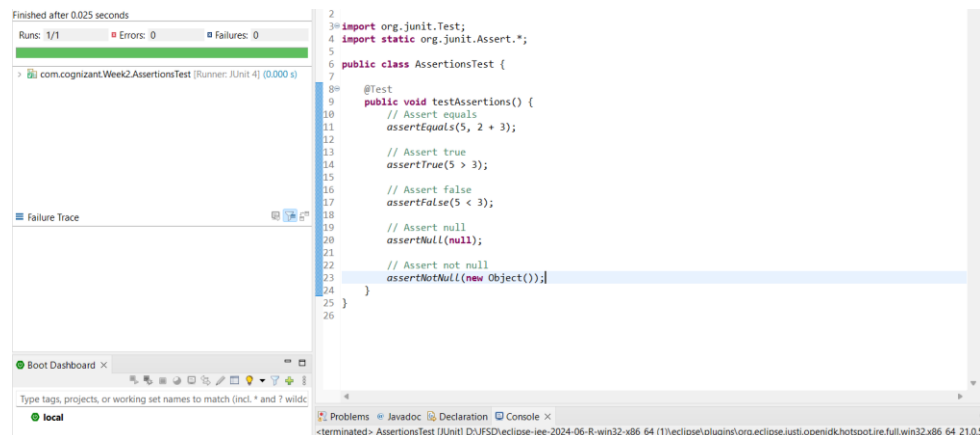
```
        // Assert not null
```

```
        assertNotNull(new Object());
```

```
    }
```

```
}
```

Output:



## 5. Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit

Create a test file named with CalculatorTestAAA.java.

```
package week2;
```

```
import static org.junit.Assert.*;
```

```
import org.junit.After;
```

```
import org.junit.Before;
```

```
import org.junit.Test;
```

```
public class CalculatorTestAAA {
```

```
    private Calculator calculator;
```

```
    // Setup method (runs before every test)
```

```
    @Before
```

```
    public void setUp() {
```

```
        calculator = new Calculator();
```

```
        System.out.println("Setting up Calculator instance...");
```

```
}
```

```
// Teardown method (runs after every test)
```

```
@After
```

```
public void tearDown() {
```

```
    calculator = null;
```

```
    System.out.println("Cleaning up Calculator instance...");
```

```
}
```

```
@Test
```

```
public void testAdd() {
```

```
    // Arrange
```

```
    int a = 2, b = 3;
```

```
    // Act
```

```
    int result = calculator.add(a, b);
```

```
    // Assert
```

```
    assertEquals(5, result);
```

```
}
```

```
@Test
```

```
public void testSubtract() {
```

```
    int result = calculator.subtract(10, 4);
```

```
    assertEquals(6, result);
```

```
}
```

```
@Test
```

```
public void testMultiply() {
```

```
    int result = calculator.multiply(4, 3);  
    assertEquals(12, result);  
}
```

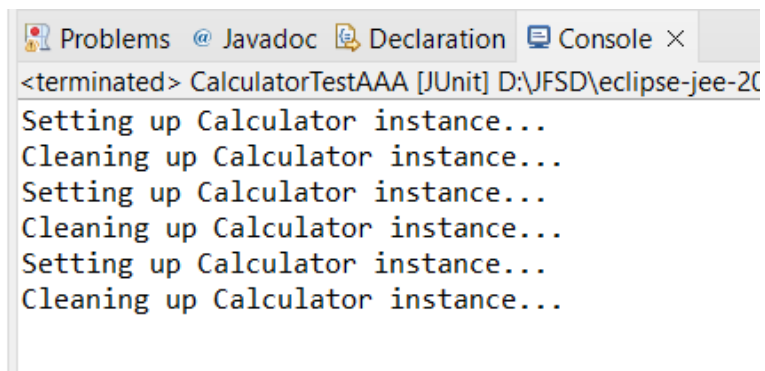
@Test

```
public void testDivide() {  
    int result = calculator.divide(20, 4);  
    assertEquals(5, result);  
}
```

@Test(expected = IllegalArgumentException.class)

```
public void testDivideByZero() {  
    calculator.divide(10, 0);  
}  
}
```

Output:



The screenshot shows the Eclipse IDE's Console window. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output shows the results of a JUnit test run for 'CalculatorTestAAA'. The output is as follows:

```
<terminated> CalculatorTestAAA [JUnit] D:\JFSD\eclipse-jee-20...  
Setting up Calculator instance...  
Cleaning up Calculator instance...  
Setting up Calculator instance...  
Cleaning up Calculator instance...  
Setting up Calculator instance...  
Cleaning up Calculator instance...
```

## TDD using JUnit5 and Mockito

### 6. Mocking and Stubbing

Add Dependencies:

```
<dependency>
  <groupId>org.mockito</groupId>
  <artifactId>mockito-core</artifactId>
  <version>5.11.0</version> <!-- or latest-->
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter</artifactId>
  <version>5.10.0</version> <!-- or latest-->
  <scope>test</scope>
</dependency>
```

Created the externalAPI interface:

```
package week2;

public interface ExternalApi {
    String getData();
}
```

Created the MyService Class:

```
package week2;

public class MyService {
```



```
private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {
    this.externalApi = externalApi;
}

public String fetchData() {
    return externalApi.getData();
}
}
```

Created the myServiceTest Class:

```
package week2;

import static org.mockito.Mockito.*;
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;
import org.mockito.Mockito;

public class MyServiceTest {

    @Test
    public void testExternalApi() {
        // Step 1: Create a mock of ExternalApi
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);
```

// Step 2: Stub the getData() method

```
when(mockApi.getData()).thenReturn("Mock Data");
```

// Step 3: Inject the mock into MyService

```
MyService service = new MyService(mockApi);
```

// Step 4: Call the method and assert the result

```
String result = service.fetchData();
```

```
assertEquals("Mock Data", result);
```

```
}}
```

Output:

The screenshot displays the Eclipse IDE interface. The top part shows two Java files: `ExternalApi.java` and `MyService.java`. `ExternalApi.java` defines a public interface `ExternalApi` with a `String getData();` method. `MyService.java` defines a public class `MyService` that implements `ExternalApi`. It has a private field `ExternalApi externalApi;`, a constructor `MyService(ExternalApi externalApi)` that assigns the parameter to the field, and a `fetchData()` method that returns `externalApi.getData();`. The bottom part of the screenshot shows the console output. It displays the results of running a JUnit test `CalculatorTestAAA`. The output includes messages like "Setting up Calculator instance...", "Cleaning up Calculator instance...", and "Setting up Calculator instance...". There are also warnings from the OpenJDK 64-Bit Server VM, such as "WARNING: A Java agent has been loaded dynamically" and "WARNING: If a serviceability tool is in use, please run with -XX:-EnableDynamicAgentLoading to hide this warning".

```
3 public interface ExternalApi {
4     String getData();
5 }
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

## 7. Mocking and Stubbing with Multiple Returns:

Created the externalAPIStub interface:

```
package week2;
```

```
public interface ExternalApiStub {  
    String getData();  
}
```

Created the MyServiceStub Class:

```
package week2;
```

```
public class MyServiceStub {  
    private ExternalApiStub externalApiStub;  
  
    public MyServiceStub(ExternalApiStub externalApiStub) {  
        this.externalApiStub = externalApiStub;  
    }  
  
    public String fetchFirstCall() {  
        return externalApiStub.getData();  
    }  
}
```

```
    public String fetchSecondCall() {  
        return externalApiStub.getData();  
    }  
}
```

Created the MyServiceStubTest Class:

```
package week2;
```

```
import static org.mockito.Mockito.*;  
import static org.junit.jupiter.api.Assertions.*;
```

```
import org.junit.jupiter.api.Test;
```

```
public class MyServiceTestStub {
```

```
    @Test
```

```
    public void testExternalApiMultipleReturns() {
```

```
        // Step 1: Create mock object
```

```
        ExternalApiStub mockApi = mock(ExternalApiStub.class);
```

```
        // Step 2: Stub to return different values for consecutive calls
```

```
        when(mockApi.getData())
```

```
            .thenReturn("First Response")
```

```
            .thenReturn("Second Response");
```

// Step 3: Inject into service

```
MyServiceStub service = new MyServiceStub(mockApi);
```

```
String first = service.fetchFirstCall();
```

```
String second = service.fetchSecondCall();
```

```
System.out.println("First: " + first); // <- Print here
```

```
System.out.println("Second: " + second);
```

// Step 4: Test both calls

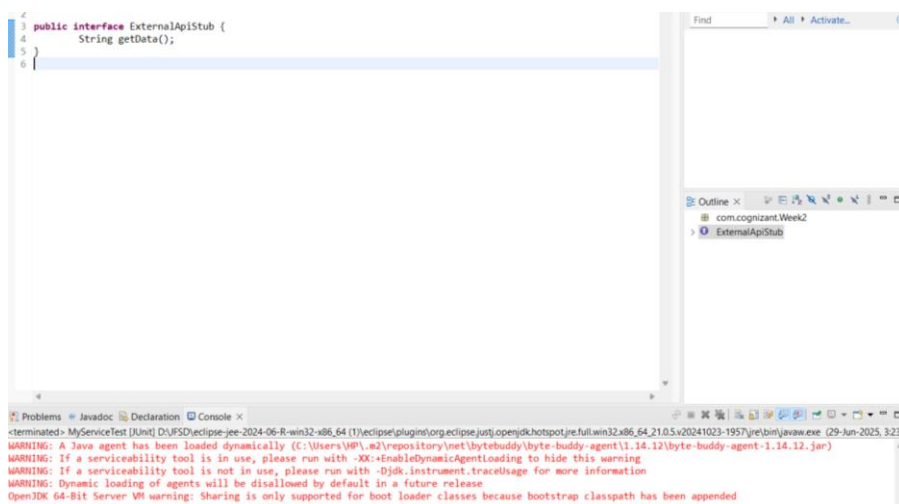
```
assertEquals("First Response", service.fetchFirstCall());
```

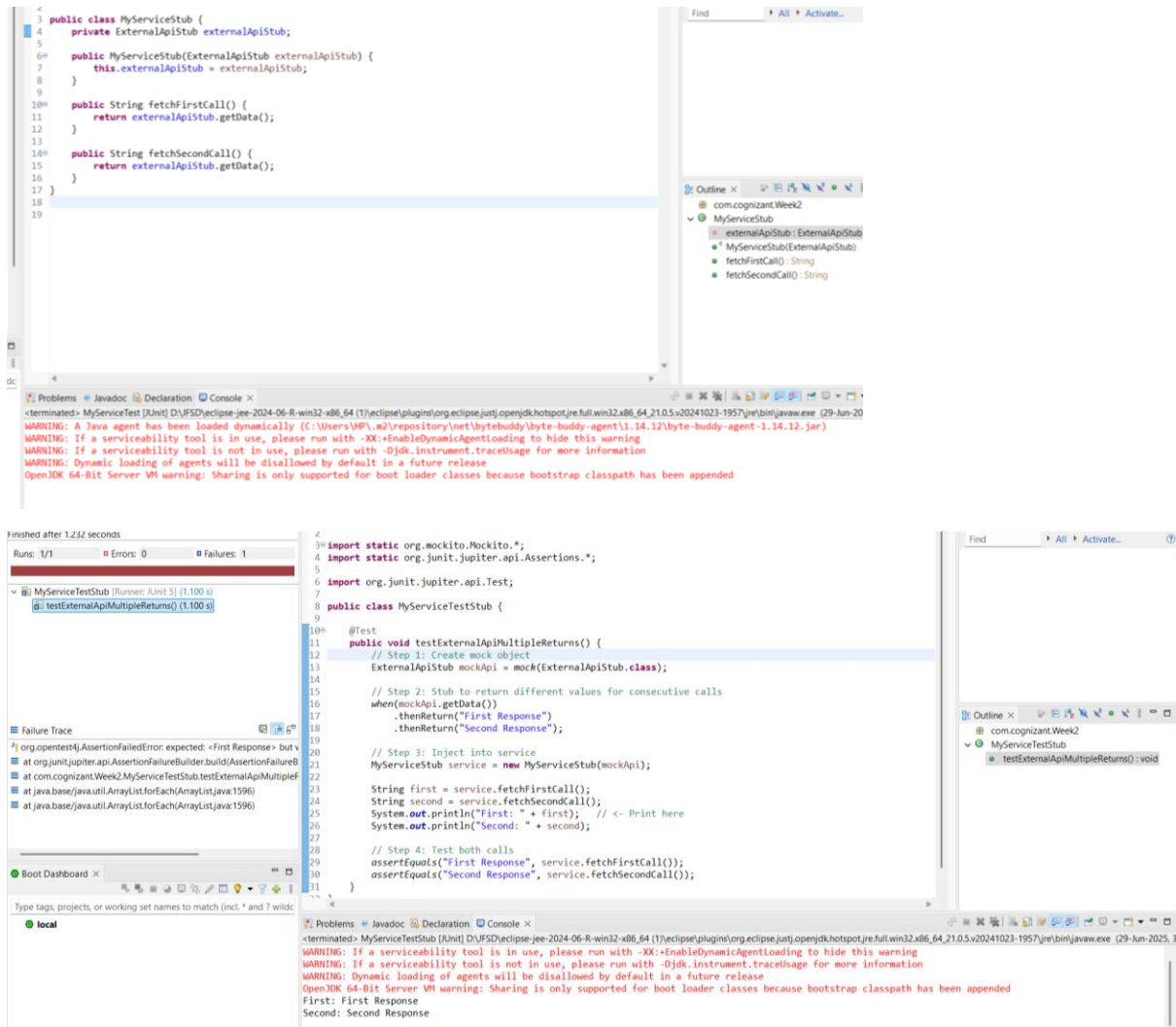
```
assertEquals("Second Response", service.fetchSecondCall());
```

```
}
```

```
}
```

Output:





## 8. Verifying Interactions:

NotificationService:

package week2;

```

public interface NotificationService {

    void sendEmail();

    void sendSMS();

```

```
    void sendPush();  
}
```

**UserNotifier:**

```
package week2;
```

```
public class UserNotifier {  
    private NotificationService service;  
  
    public UserNotifier(NotificationService service) {  
        this.service = service;  
    }  
  
    public void notifyUser() {  
        service.sendEmail();  
        service.sendSMS();  
        service.sendPush();  
    }  
}
```

**UserNotifierTest:**

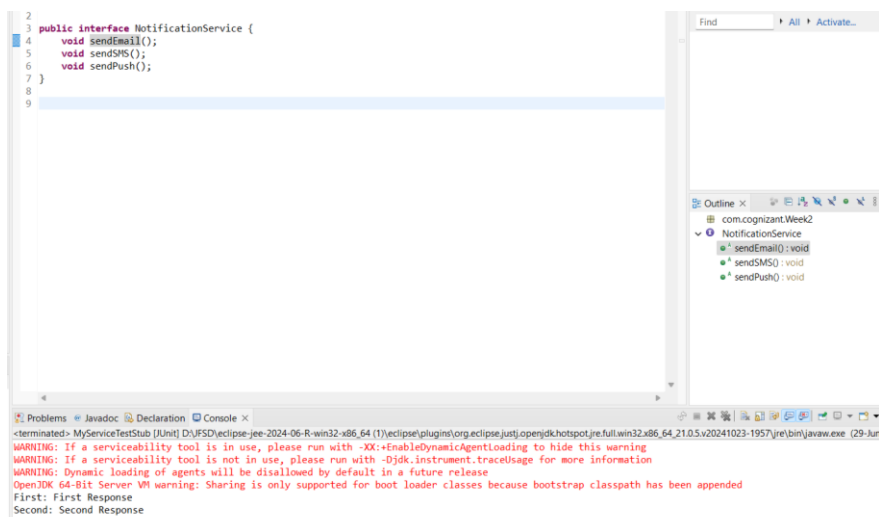
```
package week2;
```

```
import static org.mockito.Mockito.*;  
import org.junit.jupiter.api.Test;  
import org.mockito.InOrder;  
  
public class UserNotifierTest {
```

@Test

```
public void testNotificationOrder() {  
  
    // Step 1: Create a mock  
    NotificationService mockService = mock(NotificationService.class);  
  
    // Step 2: Call methods in a specific order  
    UserNotifier notifier = new UserNotifier(mockService);  
    notifier.notifyUser();  
  
    // Step 3: Verify interaction order  
    InOrder inOrder = inOrder(mockService);  
    inOrder.verify(mockService).sendEmail();  
    inOrder.verify(mockService).sendSMS();  
    inOrder.verify(mockService).sendPush();  
}  
}
```

Output:





```
2 public class UserNotifier {
3     private NotificationService service;
4
5     public UserNotifier(NotificationService service) {
6         this.service = service;
7     }
8
9     public void notifyUser() {
10        service.sendEmail();
11        service.sendSMS();
12        service.sendPush();
13    }
14 }
15
16
```

Find | All | Activate...

Outline X

- com.cognizant.Week2
  - UserNotifier

Problems | Javadoc | Declaration | Console X

<terminated> MyServiceTestStub [JUnit] D:\FSD\ eclipse-jee-2024-06-R-win32-x86\_64 (1)\eclipse\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86\_64.21.0.5.v20241023-1957\re\bin\javaw.exe (29-Jun-2025, 3:35)

WARNING: If a serviceability tool is in use, please run with -XX:-enableDynamicAgentLoading to hide this warning

WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.tracelUsage for more information

WARNING: Dynamic loading of agents will be disabled by default in a future release

OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended

First: First Response

Second: Second Response

```
3 import static org.mockito.Mockito.*;
4 import org.junit.jupiter.api.Test;
5 import org.mockito.InOrder;
6
7 public class UserNotifierTest {
8
9     @Test
10    public void testNotificationOrder() {
11        // Step 1: Create a mock
12        NotificationService mockService = mock(NotificationService.class);
13
14        // Step 2: Call methods in a specific order
15        UserNotifier notifier = new UserNotifier(mockService);
16        notifier.notifyUser();
17
18        // Step 3: Verify interaction order
19        InOrder inOrder = inOrder(mockService);
20        inOrder.verify(mockService).sendEmail();
21        inOrder.verify(mockService).sendSMS();
22        inOrder.verify(mockService).sendPush();
23    }
24 }
25
26
```

Find | All | Activate...

Outline X

- com.cognizant.Week2
  - UserNotifierTest

Problems | Javadoc | Declaration | Console X

<terminated> UserNotifierTest [JUnit] D:\FSD\ eclipse-jee-2024-06-R-win32-x86\_64 (1)\eclipse\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86\_64.21.0.5.v20241023-1957\re\bin\javaw.exe (29-Jun-2025, 3:35)

WARNING: A Java agent has been loaded dynamically (C:\Users\HP\1.m2\repository\net\bytebuddy\byte-buddy-agent\1.14.12\byte-buddy-agent-1.14.12.jar)

WARNING: If a serviceability tool is in use, please run with -XX:-enableDynamicAgentLoading to hide this warning

WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.tracelUsage for more information

WARNING: Dynamic loading of agents will be disabled by default in a future release

OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended

## SLF4J logging framework

### 9. Logging Error Messages and Warning Levels:

Added Dependencies:

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

Create LoggingExample.java:

```
package week2;
```

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
```

```
public class LoggingExample {
```

```
    private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);
```

```
    public static void main(String[] args) {
```

```
        logger.error("This is an error message");
```

```
        logger.warn("This is a warning message");
```

```
    }
```

```
}
```

Output:



```
<terminated> LoggingExample [Java Application] D:\FSD\workspace\jee-2024-06-R-win32-x86_64 (1)\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.5.v20241023-1957\re\bin\javaw.exe (29-3)
15:36:12.534 [main] ERROR com.cognizant.Week2.LoggingExample - This is an error message
15:36:12.537 [main] WARN com.cognizant.Week2.LoggingExample - This is a warning message
```