

1)

Test Case Number	Inputs batteryPower (watts)	Expected Outputs					Basis Path
		red	yellow	green	bell	siren	
1	0.0	FALSE	FALSE	FALSE	FALSE	TRUE	15-16-32
2	49.9	FALSE	FALSE	FALSE	TRUE	FALSE	15-18-19-32
3	75.0	TRUE	FALSE	FALSE	FALSE	FALSE	15-18-21-22-32
4	124.9	TRUE	TRUE	FALSE	FALSE	FALSE	15-18-21-24-25-32
5	250.0	FALSE	TRUE	FALSE	FALSE	FALSE	15-18-21-24-27-28-32
6	250.1	FALSE	FALSE	TRUE	FALSE	FALSE	15-18-21-24-27-30-32
7	0.1	FALSE	FALSE	FALSE	TRUE	FALSE	-
8	50.0	TRUE	FALSE	FALSE	FALSE	FALSE	-
9	75.1	TRUE	TRUE	FALSE	FALSE	FALSE	-
10	125.0	FALSE	TRUE	FALSE	FALSE	FALSE	-
11	1,000.0	FALSE	FALSE	TRUE	FALSE	FALSE	-

The screenshot shows the Eclipse IDE with a Java project named 'HW4'. The main editor displays the 'P1.java' file, which contains a 'ProblemClass' and a 'Test' class. The 'ProblemClass' has methods for calculating lights (red, yellow, green, bell, siren) based on battery power. The 'Test' class uses JUnit 4 to test these methods with various inputs. The left sidebar shows the 'Package Explorer' with the 'test' package containing 'P1Test.java'. The bottom status bar shows 'Coverage' for 'P1' at 30.4%.

```

package testing;
import static org.junit.Assert.*;

@RunWith(JUnit4.class)
public class P1 {

    ProblemClass problem;

    @Before
    public void setUp() {
        problem = new ProblemClass();
    }

    @SuppressWarnings("unused")
    private static final Object[] parametersForProblemClassTest() {
        return {
            {0.0, false, false, false, false, true},
            {49.9, false, false, false, true, false},
            {75.0, true, false, false, false, false},
            {124.9, true, true, false, false, false},
            {250.0, false, true, false, false, false},
            {250.1, false, false, true, false, false},
            {0.1, false, false, false, true, false},
            {50.0, true, false, false, false, false},
            {75.1, true, true, false, false, false},
            {125.0, false, true, false, false, false},
            {1000.0, false, false, true, false, false}
        };
    }

    @Test
    @Parameters(method="parametersForProblemClassTest")
    public void test(double batteryPower, boolean red, boolean yellow, boolean green, boolean bell, boolean siren) {
        problem.calculateLights(batteryPower);
        assertEquals("red", problem.isRedLight(), red);
        assertEquals("yellow", problem.isYellowLight(), yellow);
        assertEquals("green", problem.isGreenLight(), green);
        assertEquals("bell", problem.isBell(), bell);
        assertEquals("siren", problem.isSiren(), siren);
    }
}

```

2)

MCDC test case	policyHolder	yearsMember	multiPolicies	safetyRating	test case number in table
FFFT	FALSE	5	FALSE	501	8
FFTF	FALSE	5	TRUE	500	9
FFTT	FALSE	5	TRUE	501	7
TFTF	TRUE	5	TRUE	500	1
FTFT	FALSE	6	FALSE	501	6

The screenshot shows the Eclipse IDE interface for a Java project named 'P2'. The main editor window displays the source code for 'P2.java'. The code includes a 'setUp()' method, a '@Parameters' block with a large array of test data, and a 'test()' method. The left sidebar shows the 'Package Explorer' with a tree view of the project structure. The bottom status bar indicates 'Coverage: 50.5%' and 'Total Instructions: 1,882'.

```

33 public void setUp() throws Exception {
34     problem2 = new Problem2Class();
35 }
36
37
38
39 @Parameters
40 public static Collection<Object[]> data() {
41     return Arrays.asList(new Object[][] {
42         {5000.01, true, 5, true, 500, 0.0025, true, 4330.01},
43         {2000.00, true, 6, false, 500, 0.0025, true, 1840.25},
44         {1250.01, true, 6, false, 500, 0.0025, true, 1217.02},
45         {350.00, true, 6, false, 500, 0.0025, true, 350.93},
46         {349.99, true, 6, false, 500, 0.0025, true, 378.86},
47         {349.99, false, 6, false, 501, 0.0025, true, 378.86},
48         {349.99, false, 5, true, 501, 0.0025, true, 378.86},
49         {349.99, false, 5, false, 501, 0.0025, false, 378.86},
50         {349.99, false, 5, true, 500, 0.0025, false, 378.86},
51         {1250.00, false, 6, false, 501, 0.0025, true, 1215.47},
52         {1999.99, false, 6, false, 501, 0.0025, true, 1948.49},
53         {5000.00, false, 6, false, 501, 0.0025, true, 4600.63},
54         {0.00, false, 6, false, 501, 0.0025, true, 0.00},
55         {10000.00, false, 6, false, 501, 0.0025, true, 8600.00},
56         {349.99, false, 5, true, 1, 0.0025, false, 378.86},
57         {349.99, false, 5, true, 999, 0.0025, true, 378.86},
58         {349.99, false, 0, true, 500, 0.0025, false, 378.86},
59         {349.99, false, 50, true, 500, 0.0025, true, 378.86}
60     });
61 }
62
63
64
65 public P2(double premium, boolean policyHolder, int yearsMember, boolean multiPolicies, int safetyRating, double taxRate, boolean primeStatus, double totalPremium) {
66     this.premium = premium;
67     this.policyHolder = policyHolder;
68     this.yearsMember = yearsMember;
69     this.multiPolicies = multiPolicies;
70     this.safetyRating = safetyRating;
71     this.taxRate = taxRate;
72     this.primeStatus = primeStatus;
73     this.totalPremium = totalPremium;
74 }
75
76
77
78 @Test
79 public void test() {
80     problem2.determinePrimeStatus(premium, policyHolder, yearsMember, multiPolicies, safetyRating, taxRate);
81     assertEquals("primeStatus", problem2.isPrimeStatus(), 0.01);
82     assertEquals("totalPremium", problem2.getTotalPremium(), 0.01);
83 }
84
85

```

3)

Test Case	Inputs			Expected Outputs						Basis Path	MCDC stmt 22-26
Number	distance (ft.)	cruiseRequested	speed (mph)	redLight	yellowLight	greenLight	caution	warning	cruiseEngaged		
1	200.0	TRUE	40.1	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	9-10-22-23-24-25-26-27	
2	100.1	TRUE	40.1	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	9-12-13-22-23-24-25-26-27	
3	75.0	TRUE	40.1	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	9-12-16-17-22-23-24-25-26-27	
4	74.9	TRUE	40.1	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	9-12-16-19-22-23-24-25-26-27	
5	50.0	FALSE	40.1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	9-12-16-19-22-27	FTTT
6	49.9	TRUE	40.1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	9-12-16-19-22-23-27	TFTT
7	50.0	TRUE	40	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	9-12-16-19-22-23-24-27	TTFT
8	50.0	TRUE	65.1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	9-12-16-19-22-23-24-25-27	TTTF
9	50.0	TRUE	65	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	-	TTTT
10	50.0	TRUE	40.1	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	tests other TT BV	
11	100.0	TRUE	40.1	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	tests missing BV from bPath	
12	199.9	TRUE	40.1	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	tests missing BV from bPath	
13	0.0	TRUE	65	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	Extreme range distance	
14	1,000.0	TRUE	65	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	Extreme range distance	
15	50.0	TRUE	0	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	Extreme range speed	
16	50.0	TRUE	100	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	Extreme range speed	

Test cases 9-16 may appear in any order

- indicates a don't care where ANY value can be used

```

package testing;

import static org.junit.Assert.*;

@RunWith(SuiteRunner.class)
public class P3 {

    ProblemClass problem3;

    @Before
    public void setup() {
        problem3 = new ProblemClass();
    }

    @Parameters({"src/excelTestCases/Problem3TestCasesTable.csv"})
    @Test
    public void test(int testCaseNumber, double distance, boolean cruiseRequested, double speed, boolean redLight, boolean yell
        boolean greenLight, boolean caution, boolean warning, boolean cruiseEngaged, String basisPath, String mdc) {
        problem3.setParameters(cruiseRequested, distance, speed);
        assertEquals("redLight", problem3.isRedLight());
        assertEquals("yellowLight", problem3.isYellowLight());
        assertEquals("greenLight", problem3.isGreenLight());
        assertEquals("caution", problem3.isCaution());
        assertEquals("warning", problem3.isWarning());
        assertEquals("cruiseEngaged", problem3.isCruiseEngaged());
    }
}

```

4)

Test Case Number	Inputs			Exp Out	Basis Path	MCDC
	landin	speed (mp)	altitude (f)	return		
1	TRUE	500.1	4,999.9	engageRetro	10-11-12-13-14-22	stmt 11 TTT
2	FALSE	500.1	4,999.9	disengageRetro	10-21-22	
3	TRUE	150.0	2,499.9	deployPods	10-11-16-17-18-19-22	stmt 16 TTT
4	TRUE	500.1	2,499.9	orbit	10-11-12-22	stmt 11 TFT
5	TRUE	149.9	2,499.9	orbit	10-11-16-22	stmt 16 FTT
6	TRUE	500.1	5,000.0	orbit	10-11-12-13-22	stmt 11 TTF
7	TRUE	150.0	1,000.0	orbit	10-11-16-17-22	stmt 16 TFT
8	TRUE	150.0	2,500.0	orbit	10-11-16-17-18-22	stmt 16 TTF
9	TRUE	500.0	4,999.9	orbit	-	stmt 11 FTT
10	TRUE	500.1	2,500.0	engageRetro	-	
11	TRUE	150.0	1,000.1	deployPods	-	
12	TRUE	200.0	10,000.0	orbit	extreme range altitude	
13	TRUE	200.0	0.0	orbit	extreme range altitude	
14	TRUE	1,000.0	5,000.1	orbit	extreme range speed	
15	TRUE	0.0	5,000.1	orbit	extreme range speed	
- indicates a don't care where ANY value can be used						
Test cases 9-15 can be in any order						

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure with 'P4.java' selected. The main editor shows the source code of 'P4.java', which includes a package declaration, imports, and a public class 'P4' with a 'test' method. The 'test' method is annotated with '@RunWith(JUnit4.class)' and '@Test'. It uses 'assertEquals' to verify the output of the 'landCraft' method. The bottom status bar shows the current file is 'P4 (13 Nov, 2019 3:12:33 PM)' and the coverage is 6.0%.

```

package testing;
import static org.junit.Assert.*;

@RunWith(JUnit4.class)
public class P4 {
    ProblemClass problem;

    @Before
    public void setUp() {
        problem = new ProblemClass();
    }

    @Parameters("src/excelTestCases/ProblemTestCasesTable.csv")
    @Test
    public void test(int testCase, boolean landing, double speed, double altitude, ProblemClass.landing output, String basic) {
        assertEquals(output, problem.landCraft(landing, altitude, speed));
    }
}

```

The Package Explorer shows the following test results:

- test (0.001 s)
 - [1] 1.TRUE,500.1,4999.9,engageRetro,10-11-12-13-14-22,stmt 11 TTT (test) (0.001 s)
 - [2] 2.FALSE,500.1,4999.9,disengageRetro,10-21-22,stmt 16 TTT (test) (0.001 s)
 - [3] 3.TRUE,150.0,2499.9,deployPods,10-11-16-17-18-19-22,stmt 16 TTT (test) (0.000 s)
 - [4] 4.TRUE,500.1,2499.9,orbit,10-11-12-22,stmt 11 TFT (test) (0.000 s)
 - [5] 5.TRUE,149.9,2499.9,orbit,10-11-16-22,stmt 16 FTT (test) (0.000 s)
 - [6] 6.TRUE,500.1,5000.0,orbit,10-11-12-13-22,stmt 11 TTF (test) (0.000 s)
 - [7] 7.TRUE,150.0,1000.0,orbit,10-11-16-17-22,stmt 16 TFT (test) (0.000 s)
 - [8] 8.TRUE,150.0,2500.0,orbit,10-11-16-17-18-22,stmt 16 TTF (test) (0.002 s)
 - [9] 9.TRUE,500.0,4999.9,orbit,11 FTT (test) (0.001 s)
 - [10] 10.TRUE,500.1,2500.0,engageRetro,stmt 11 FTT (test) (0.002 s)
 - [11] 11.TRUE,150.0,1000.1,deployPods,stmt 16 TTT (test) (0.002 s)
 - [12] 12.TRUE,200.0,10000.0,orbit,extreme range altitude,stmt 11 FTT (test) (0.001 s)
 - [13] 13.TRUE,200.0,0.0,orbit,extreme range altitude,stmt 16 TTT (test) (0.002 s)
 - [14] 14.TRUE,1000.0,5000.1,orbit,extreme range speed,stmt 11 FTT (test) (0.002 s)
 - [15] 15.TRUE,0.0,5000.1,orbit,extreme range speed,stmt 16 TTT (test) (0.001 s)

5)

Test Case Number	Inputs x	Exp Out y	Basis Path Tested
1	-4.01	0.00	7-8-20
2	-2.00	2.00	7-10-11-20
3	1.99	1.96	7-10-13-14-20
4	3.99	0.01	7-10-13-16-17-20
5	4.00	0.00	7-10-13-16-19-20
6	-6.00	0.00	-
7	-4.00	0.00	-
8	-1.99	1.96	-
9	2.00	2.00	-
10	8.00	0.00	-
11	-3.00	1.00	-
12	3.00	1.00	-
13	0.00	-2.00	-
14	1.00	-1.00	-
alternative x=-1.00 and y=-1.00			
Test Cases 6-14 can be in any order			

The screenshot displays the Eclipse IDE environment. On the left, the Package Explorer shows a test suite 'testing.P5' with 14 test cases listed, each with its parameters and expected output. The main editor shows the source code of 'P5.java', which includes a 'test' method that takes test case parameters and asserts the result. The Outline on the right shows the class structure. The bottom status bar shows coverage information: 3.4% coverage, 64 covered instructions, 1,818 missed instructions, and 1,882 total instructions.