

```
1  
2  
3     White-Box Testing {  
4  
5  
6  
7         int courseNumber = 4;  
8  
9  
10  
11  
12     }  
13  
14
```

```
1 Cuprins {
```

```
2  
3     01 White-box Testing
```

```
4         // Ce este testarea White-box  
5         și când este folosită?
```

```
6  
7             02 Tehnicile de testare  
8                 White-box
```

```
9                 // Statement Coverage, Decision  
10                Coverage, Path Coverage
```

```
11                03 Exerciții
```

```
12                    // Întrebări ISTQB de  
13                    White-box
```

```
14 }
```

```
1 01  {
2
3
4
5      [@test White-box]
6
7
8      String definition = "Testare bazată
9          pe o analiză a structurii interne
10         de componentă sau sistem."
11
12
13
14 }
```

```
1      02  {  
2  
3  
4  
5      [@test Statement Coverage]  
6  
7      String definition = "Procentul de  
8      instrucțiuni executabile care au  
9      fost exercitată printr-o suită de  
10     teste."  
11  
12    }  
13  
14
```

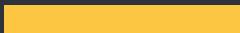
StatementCoverage.java

```
1  class Statement {  
2  
3      | [ 🖥 ] O entitate într-un limbaj de programare, care este  
4      | de obicei cea mai mică unitate de execuție  
5      | indivizibilă.  
6  }  
7  
8  class Main {  
9  
10     | // Un exemplu de statement  
11     | System.out.println("Hello World!");  
12  
13 }  
14 }
```

1 Statement coverage 'Exercitiu' {

2
3 Să luăm în considerare fragmentul de
4 cod de mai jos:

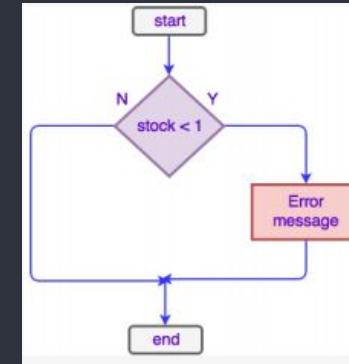
```
5  
6     public void main(String[] args) {  
7         int Stock = 0;  
8  
9         if (Stock < 1) {  
10             System.out.println("Error message");  
11         }  
12     }
```

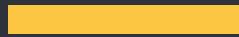
12 Statement coverage  100%

13 }
14 }

Statement coverage 'Exercitiu' {

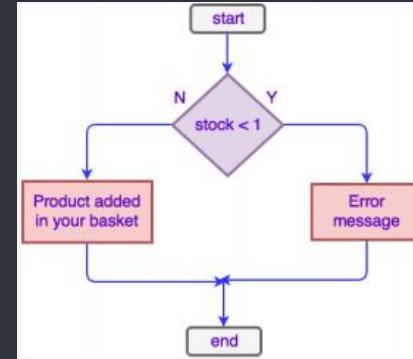
```
1  
2  
3  
4  
5  
6     public void main(String[] args) {  
7         int Stock = 0;  
8  
9         if (Stock < 1) {  
10             System.out.println("Error message");  
11     }  
12 }
```

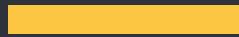


Statement coverage  100%

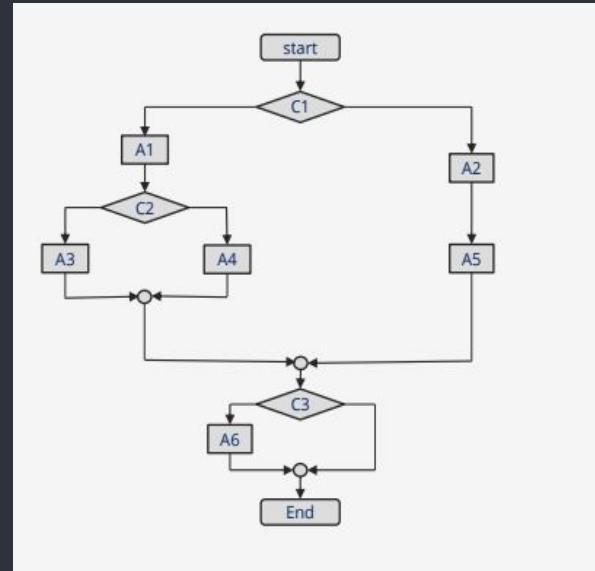
Statement coverage 'Exercitiu' {

```
1  
2  
3  
4  
5     public void main(String[] args) {  
6         int Stock = 0;  
7  
8         if (Stock < 1) {  
9             System.out.println("Error message");  
10        } else {  
11            System.out.println("Product added to basket");  
12        }  
13    }  
14 }
```



Statement coverage  100%

```
1 Statement coverage 'Exercitiu' {  
2  
3     public void main(String[] args) {  
4         int sum = 22;  
5         int number = 2;  
6  
6         double average = 0.0;  
7  
8         boolean condition = false;  
9  
9         if (sum < 10) {  
10             sum = number + sum;  
11             number++;  
12             condition = true;  
13         } else {  
14             average = sum / number;  
15             if (average < 20) {  
16                 System.out.println("Average is smaller than 20");  
17                 condition = true;  
18             } else {  
19                 System.out.println("Average is bigger than 20");  
20                 condition = false;  
21             }  
22         }  
23         if (!condition) {  
24             System.out.println("Condition not fulfilled");  
25         }  
26     }  
27 }  
28 }
```



1 ISTQB Q1 {
2

3 **Question #22 (1 Point)**
4 Which one of the following is the description of statement coverage?

- 5 a) It is a metric, which is the percentage of test cases that have been executed
6 b) It is a metric, which is the percentage of statements in the source code that have been
7 executed
8 c) It is a metric, which is the number of statements in the source code that have been
9 executed by test cases that are passed
10 d) It is a metric, that gives a true/false confirmation if all statements are covered or not

11
12 Select ONE option.
13
14 }

1 **ISTQB Q2 {**

4 **Question #27 (1 Point)**

5 Which of the following descriptions of statement coverage is CORRECT?

- 6 a) Statement coverage is a measure of the number of lines of source code exercised by tests
- 7 b) Statement coverage is a measure of the proportion of executable statements in the source
- 8 code exercised by tests
- 9 c) Statement coverage is a measure of the percentage of lines of source code (without
- 10 comments) exercised by tests
- 11 d) Statement coverage is a measure of the number of executable statements in the source
- 12 code exercised by tests

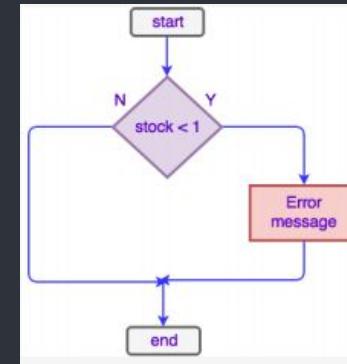
13 Select ONE option.

14 }

```
1      03  {  
2  
3  
4  
5      [@Test Decision Coverage]  
6  
7      String definition = “Acoperirea  
8      deciziilor, legată de testarea ramurilor, este  
9      evaluarea procentului deciziei (de exemplu,  
10     opțiunile adevărate și false ale unei declarații IF)  
11     care a fost exercitat de o suită de teste”  
12  }  
13  
14
```

```
1  Decision Coverage() {  
2  
3  
4      [  ] // Testarea deciziilor este o formă de testare a fluxului  
5          de control, deoarece urmează un flux specific de control  
6          prin punctele de decizie  
7  
8      [  ] // Acoperirea deciziilor este mai puternică decât  
9          acoperirea declarației  
10  
11     [  ] // Acoperirea 100% a deciziilor garantează acoperirea  
12          declarației 100%, dar nu vice versa  
13  
14 }
```

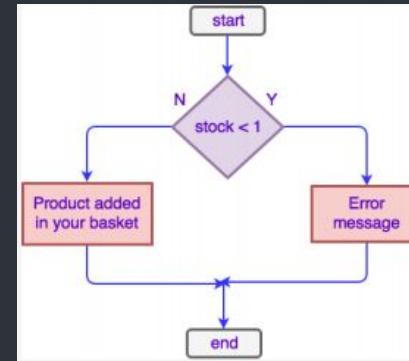
```
1 Decision coverage 'Exercitiu' {  
2  
3  
4  
5  
6     public void main(String[] args) {  
7         int Stock = 0;  
8  
9         if (Stock < 1) {  
10             System.out.println("Error message");  
11         }  
12     }  
13 }  
14 }
```

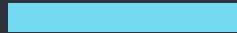


Statement coverage  100%

Decision coverage  50%

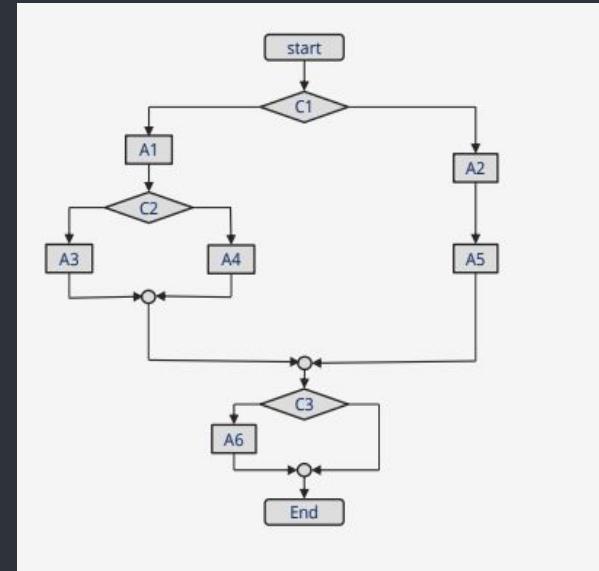
```
1 Decision coverage 'Exercitiu' {  
2  
3  
4  
5     public void main(String[] args) {  
6         int Stock = 0;  
7  
8         if (Stock < 1) {  
9             System.out.println("Error message");  
10        } else {  
11            System.out.println("Product added to basket");  
12        }  
13    }  
14 }
```



Decision coverage  100%

Statement coverage  100%

1 Decision coverage ‘Exercitium’ {
2
3 public void main(String[] args) {
4 int sum = 22;
5 int number = 2;
6
6 double average = 0.0;
7
8 boolean condition = false;
9
9 if (sum < 10) {
10 sum = number + sum;
11 number++;
12 condition = true;
13 } else {
14 average = sum / number;
15 if (average < 20) {
16 System.out.println("Average is smaller than 20");
17 condition = true;
18 } else {
19 System.out.println("Average is bigger than 20");
20 condition = false;
21 }
22 }
23 if (!condition) {
24 System.out.println("Condition not fulfilled");
25 }
26 }
27 }



Decision coverage 'Exercitium' {

```
1 Decision coverage 'Exercitium' {
2
3     if (x > y) {
4
5         if (x > z) {
6
7             System.out.print("Message");
8
9         }
10
11
12
13     }
14 }
```

1 TC

Statement coverage



100%

Decision coverage



33%

Decision coverage 'Exercitium' {

```
1 Decision coverage 'Exercitium' {
2
3     if (x > y) {
4
5         if (x > z) {
6
7             System.out.print("Message");
8
9         }
10
11
12
13     }
14 }
```

3 TC

Statement coverage



100%

Decision coverage



100%

```
1 Decision coverage 'Exercitium' {
2
3
4     if (x > y) {
5
6         if (x > z) {
7
8             System.out.print("Message");
9         } else {
10            System.out.println("Other Message");
11        }
12    }
13
14 }
```

2 TC

Statement coverage



100%

Decision coverage

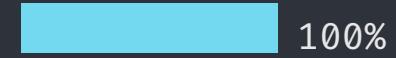


75%

```
1 Decision coverage 'Exercitium' {
2
3
4     if (x > y) {
5
6         if (x > z) {
7
8             System.out.print("Message");
9         } else {
10            System.out.println("Other Message");
11        }
12    }
13
14 }
```

3 TC

Decision coverage



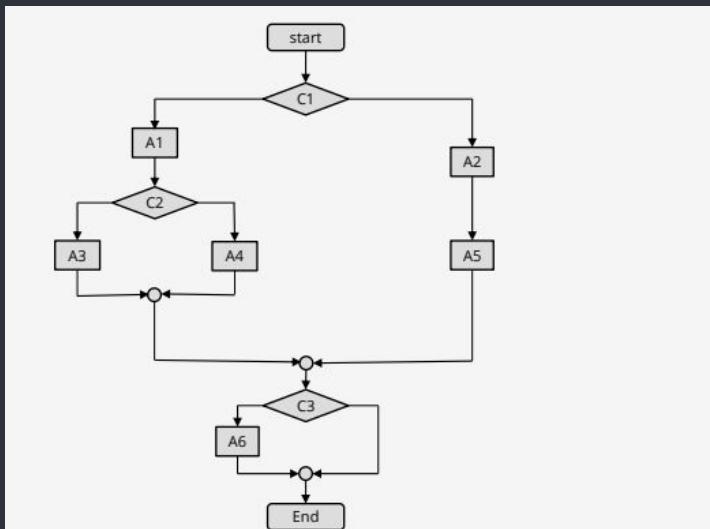
100%

Statement coverage



100%

```
1 Decision coverage 'Exercitiu' {  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13 }  
14 }
```



The diagram shows a control flow graph with the following structure:

- start** leads to decision diamond **C1**.
- From **C1**, two paths emerge:
 - Left path: leads to action **A1**, then decision diamond **C2**. From **C2**, two paths emerge:
 - Left path: leads to action **A3**, then to a merge node (circle).
 - Right path: leads to action **A4**, then to the same merge node.
 - Right path from **C1**: leads to action **A2**, then to action **A5**, then to a merge node.- Both merge nodes converge at a decision diamond **C3**.
- From **C3**, two paths emerge:
 - Left path: leads to action **A6**, then to a final node **End**.
 - Right path: leads back to the merge node before **C3**.

```
1
2
3
4      Decision coverage  100%
5
6      Statement coverage  100%
7
8
9
10     // 100% Decision coverage asigură 100% statement coverage
11
12
13
14
```

1 ISTQB Q3 {

2

3 Question #21 (1 Point)

4 The following statement refers to decision coverage:

5 "When the code contains only a single 'if' statement and no loops or CASE statements, and its
6 execution is not nested within the test, any single test case we run will result in 50% decision
7 coverage."

8 Which of the following statement is correct?

- 9 a) The statement is true. Any single test case provides 100% statement coverage and
10 therefore 50% decision coverage
- 11 b) The statement is true. Any single test case would cause the outcome of the "if" statement to
12 be either true or false
- 13 c) The statement is false. A single test case can only guarantee 25% decision coverage in this
14 case
- 15 d) The statement is false. The statement is too broad. It may be correct or not, depending on
the tested software

16 Select ONE option.

17 }
18 }

1 ISTQB Q4 {
2
3
4

5 **Question #23 (1 Point)**
6 Which statement about the relationship between statement coverage and decision coverage is
7 true?
8
9

- a) 100% decision coverage also guarantees 100% statement coverage
- b) 100% statement coverage also guarantees 100% decision coverage
- c) 50% decision coverage also guarantees 50% statement coverage
- d) Decision coverage can never reach 100%

10 Select ONE option.
11
12
13
14 }

1 ISTQB Q5 {
2
3

4 **Question #28 (1 Point)**
5 Which of the following descriptions of decision coverage is CORRECT?

- 6
- 7
- 8
- 9
- 10
- a) Decision coverage is a measure of the percentage of possible paths through the source code exercised by tests
- b) Decision coverage is a measure of the percentage of business flows through the component exercised by tests
- c) Decision coverage is a measure of the 'if' statements in the code that are exercised with both the true and false outcomes
- d) Decision coverage is a measure of the proportion of decision outcomes in the source code exercised by tests

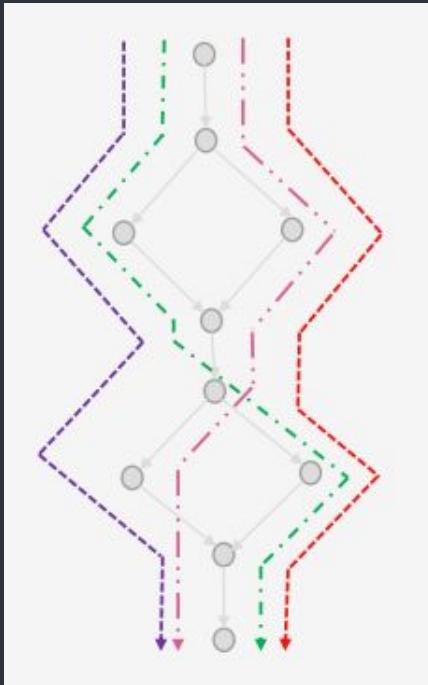
11 Select ONE option.

12
13
14 }

```
1      04  {  
2  
3  
4  
5      [@Test Path Coverage]  
6  
7  
8      String definition = "Se  
9          concentrează pe executarea tuturor  
10         căilor posibile prin program."  
11  
12     }  
13  
14
```

PathCoverage.java

1
2
3
4
5
6
7
8
9
10
11
12
13
14



2 TCs pentru statement coverage
- ROSU
- MOV

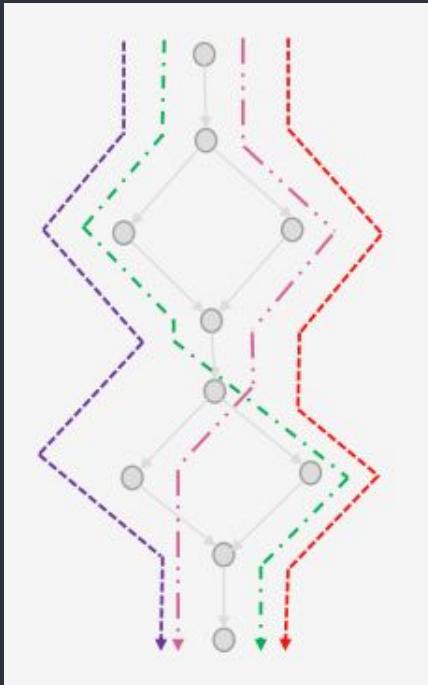
Statement coverage



100%

PathCoverage.java

1
2
3
4
5
6
7
8
9
10
11
12
13
14



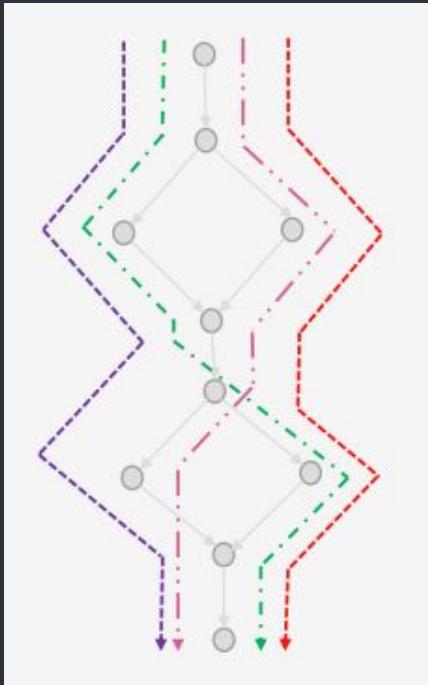
2 TCs pentru statement coverage
- VERDE
- ROZ

Decision coverage



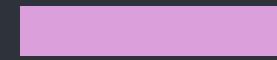
PathCoverage.java

1
2
3
4
5
6
7
8
9
10
11
12
13
14



- 4 TCs pentru path coverage
- ROSU, MOV
 - VERDE, ROZ

Path coverage



100%

Path coverage 'Exercitiu' {

1
2
3
4
5
6
7
8
9
10
11
12
13
14 }

