# PROIECT – TESTAREA SISTEMELOR SOFTWARE

Olteanu Ana-Maria

Grupa 343

**CUPRINS**

**Problema**: Licența (https://www.infoarena.ro/problema/licenta)

**1. Testare funcțională**

**a) <u>Partiționarea în clase de echivalență</u>**

*1) Domeniul de intrări*

Date de intrare:

k – numărul de minute necesare pentru prezentare

n – dimensiunea mulțimii S

S – mulțimea intervalelor pentru Mihaela

m – dimensiunea mulțimii T

T – mulțimea intervalelor pentru Decan

- k: distingem 3 clase de echivalență

  K_1 = 1…30

  K_2 = { k | k < 1}

  K_3 = { k | k >30}

- n: distingem 3 clase de echivalență

  N_1 = 1…5

  N_2 = { n | n < 1}

  N_3 = { n | n >5}

- m: distingem 3 clase de echivalență

  M_1 = 1…5

  M_2 = { m | m < 1}

  M_3 = { m | m >5}

- Presupunem că mulțimile S și T conțin valori valide între 0 și 60, dimensiunile acestora sunt egale cu valorile date (|S| = N și |T| = M), iar intervalele din mulțimi sunt disjuncte și respecta proprietatea că XS[i] < YS[i] si XT[i] < YT[i] (capătul din stanga este mai mic decât capătul din dreapta).

2) *Domeniul de iesiri*

Constă în urmatoarele 2 răspunsuri:

- 2 numere naturale separate prin câte un spațiu, reprezentând intervalul de timp în care Mihaela se întâlnește cu Decanul
- Un mesaj care sugerează faptul că intervalul de timp nu a fost găsit

Acestea sunt folosite pentru a impărți domeniul de intrare in 2 clase: una pentru cazul în care se gaseste intervalul de timp în care Mihaela se intâlnește cu Decanul și una pentru cazul în care nu se găsește acest interval:

I_1 = { (S,T) | intersecția dintre cele 2 mulțimi este ≥ k }

I_2 = { (S,T) | nu există o astfel de intersecție }

**Clasele de echivalență globale (în număr de 8)** se obțin ca o combinație a claselor individuale:

C_1111 = { (k, n, S, m ,T) | k \in K_1, n \in N_1, m \in M_1,  (S,T) \in I_1 }

C_1112 = { (k, n, S, m ,T) | k \in K_1, n \in N_1, m \in M_1, (S,T) \in I_2 }

C_2 = { (k, n, S, m ,T)  | k \in K_2 }

C_3 = { (k, n, S, m ,T) | k \in K_3 }

C_12 = { (k, n, S, m ,T) | k \in K_1, n \in N_2 }

C_13 = { (k, n, S, m ,T)  | k \in K_1, n \in N_3 }

C_112 = { (k, n, S, m ,T) | k \in K_1, n \in N_1,  m \in M_2 }

C_113 = { (k, n, S, m ,T) | k \in K_1, n \in N_1, m \in M_3 }

**Set de date:**

c_1111 = (4, 2, {(1, 10), (11, 13)}, 2, {(2, 4), (5, 12)})

c_1112 = (9, 2, {(3, 5), (7, 10)}, 2, {(4, 5), (6, 11)})

c_2 = (0, _, _, _, _)

c_3 = (32, _, _, _, _)

c_12 = (2, -1, _, _, _)

c_13 = (3, 12, _, _, _)

c_112 = (4, 3, {(1, 8), (9, 12), (13, 15)}, -2, _)

c_113 = (3, 2, {(1, 4), (5, 6)}, 12, _)

| | **Intrări** | | | | | **Rezultat afișat(expected)** |
|---|---|---|---|---|---|---|
| **Nr test** | **k** | **n** | **S** | **m** | **T** | |
| **1** | 4 | 2 | {(1, 10), (11, 13)} | 2 | {(2, 4), (5, 12)} | 5 9 |
| **2** | 9 | 2 | {(3, 5), (7, 10)} | 2 | {(4, 5), (6, 11)} | -1 |
| **3** | 0 | | | | | “k trebuie sa fie intre 1 si 30” |
| **4** | 32 | | | | | “k trebuie sa fie intre 1 si 30” |
| **5** | 2 | -1 | | | | “n trebuie sa fie intre 1 si 5” |
| **6** | 3 | 12 | | | | “n trebuie sa fie intre 1 si 5” |
| **7** | 4 | 2 | {(1, 8), (9, 12)} | -2 | | “m trebuie sa fie intre 1 si 5” |
| **8** | 3 | 2 | {(1, 4), (5, 6)} | 12 | | “m trebuie sa fie intre 1 si 5” |

```
//extras cod teste implementate in Java cu libraria JUnit
@Test
public void equivalencePartitioning() throws IOException {

    File actual_file = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\src\\pachet1\\licentaOut.txt");
    ///c_1111
    File f1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_input\\input1.txt");
    tester.gaseste_interval(f1);
    File expected_file1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_expected\\expected1.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file1, actual_file));

    ///c_1112
    File f2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_input\\input2.txt");
    tester.gaseste_interval(f2);
    File expected_file2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_expected\\expected2.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file2, actual_file));

    ///c_2
    File f3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_input\\input3.txt");
    tester.gaseste_interval(f3);
    File expected_file3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_expected\\expected3.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file3, actual_file));

    ///c_3
```

```
    File f4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_input\\input4.txt");
    tester.gaseste_interval(f4);
    File expected_file4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_expected\\expected4.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file4, actual_file));

    ///c_12
    File f5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_input\\input5.txt");
    tester.gaseste_interval(f5);
    File expected_file5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_expected\\expected5.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file5, actual_file));

    ///c_13
    File f6 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_input\\input6.txt");
    tester.gaseste_interval(f6);
    File expected_file6 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_expected\\expected6.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file6, actual_file));

    ///c_112
    File f7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_input\\input7.txt");
    tester.gaseste_interval(f7);
    File expected_file7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_expected\\expected7.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file7, actual_file));
```

```
    ///c_113
    File f8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_input\\input8.txt");
    tester.gaseste_interval(f8);
    File expected_file8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\EP_expected\\expected8.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file8, actual_file));
}
```

**b) <u>Analiza valorilor de frontieră</u>**

Odată identificate clasele, valorile de frontieră sunt ușor de identificat. Deci se vor testa următoarele valori:

- K_1 : 1, 30
- K_2 : 0
- K_3 : 31
- N_1 : 1, 5
- N_2 : 0
- N_3 : 6
- M_1 : 1, 5
- M_2 : 0
- M_3 : 6

Pentru restul claselor vom alege o valoare arbitrară.

Avem în final 30 de teste:

- C_1111:
  - (1, 1, {(3, 9)}, 1, {(4, 8)})
  - (1, 1, {(1, 2)}, 5, {(3, 4), (1, 2), (5, 7), (13, 19), (20, 23)})
  - (1, 5, {(2, 4), (7, 9), (5, 6), (12, 29), (10, 11)}, 1, {(13, 15)})
  - (1, 5, {(1, 3), (5, 9), (19, 30), (12, 14), (17, 18)}, 5, {(2, 4), (13, 20), (7, 10), (12, 13), (21, 30)})

- (30, 1, {(2, 47)}, 1, {(1, 40)})
- (30, 1, {(1, 33)}, 5, {(2, 40), (41, 43), (50, 53), (44, 45), (46, 47)})
- (30, 5, {(1,10), (19, 21), (12, 16), (25, 26), (27, 59)}, 1, {(20, 59)})
- (30, 5, {(1, 9), (10, 12), (18, 19), (14, 50), (52, 54)}, 5, {(2, 5), (8, 9), (12, 50), (51, 53), (56, 57)})

- C_1112:
  - (1, 1, {(4,7)}, 1,{(9,10)})
  - (1, 1, {(2,5)}, 5, {(1, 2), (9, 10), (6, 8), (29, 30), (18, 21)})
  - (1, 5, {(1, 2), (7, 10), (5, 8), (20, 21), (30, 33)}, 1, {(2, 5)})
  - (1, 5, {(2, 3), (5, 8), (10, 12), (20, 21), (16, 18)}, 5, {(1, 2), (8, 10), (3, 4), (12, 14), (30, 33)})
  - (30, 1, {(1, 20)}, 1, {(3, 5)})
  - (30, 5, {(1, 3), (4, 5), (6, 9), (20, 31), (34, 36)}, 1, {(21, 26)})
  - (30, 1, {(2, 4)}, 5, {(1, 2), (3, 4), (4, 5), (10, 13), (7, 8)})
  - (30, 5, {(1, 2), (3, 4), (4, 5), (10, 13), (7, 8)}, 5, {(5, 7), (9, 10), (11, 13), (15, 18), (20, 21)})
- C_2:
  - (0, _, _, _, _)
- C_3:
  - (31, _, _, _, _)
- C_12:
  - (1, 0, _, _, _)
  - (30, 0, _, _, _)
- C_13:
  - (1, 6, _, _, _)
  - (30, 6, _, _, _)
- C_112:
  - (1, 1, {(1,2)}, 0, _)
  - (1, 5, {(1, 2), (3, 4), (4, 5), (10, 13), (7, 8)}, 0, _)
  - (30, 1, {(3, 9)}, 0, _)

- (30, 5, {(1,4), (5, 6), (7, 9), (10, 13), (14, 20)}, 0, _)

- C_113:
  - (1, 1, {(1,2)}, 6, _)
  - (1, 5, {(1, 2), (3, 4), (4, 5), (10, 13), (7, 8)}, 6, _)
  - (30, 1, {(3, 9)}, 6, _)
  - (30, 5, {(1,4), (5, 6), (7, 9), (10, 13), (14, 20)}, 6, _)

| **Nr test** | **Intrări** | | | | | **Rezultat afișst(expected)** |
|---|---|---|---|---|---|---|
| | **k** | **n** | **S** | **m** | **T** | |
| **1** | 1 | 1 | {(3, 9)} | 1 | {(4, 8)} | 4 5 |
| **2** | 1 | 1 | {(1, 2)} | 5 | {(3, 4), (1, 2), (5, 7), (13, 19), (20, 23)} | 1 2 |
| **3** | 1 | 5 | {(2, 4), (7, 9), (5, 6), (12, 29), (10, 11)} | 1 | {(13, 15)} | 13 14 |
| **4** | 1 | 5 | {(2, 3), (5, 8), (10, 12), (20, 21), (16, 18)} | 5 | {(2, 4), (13, 20), (7, 10), (12, 13), (21, 30)} | 2 3 |
| **5** | 30 | 1 | {(2, 47)} | 1 | {(1, 40)} | 2 32 |
| **6** | 30 | 1 | {(1, 33)} | 5 | {(2, 40), (41, 43), (50, 53), (44, 45), (46, 47)} | 2 32 |
| **7** | 30 | 5 | {(1,10), (19, 21), (12, 16), (25, 26), (27, 59)} | 1 | {(20, 59)} | 27 57 |
| **8** | 30 | 5 | {(1, 9), (10, 12), (18, 19), (14, 50), (52, 54)} | 5 | {(2, 5), (8, 9), (12, 50), (51, 53), (56, 57)} | 14 44 |
| **9** | 1 | 1 | {(4,7)} | 1 | {(9,10)} | -1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **10** | 1 | 1 | {(1, 2)} | 5 | {(3, 4), (30, 40), (5, 7), (13, 19), (20, 23)} | -1 |
| **11** | 1 | 5 | {(1, 2), (7, 10), (5, 8), (20, 21), (30, 33)} | 1 | {(2, 5)} | -1 |
| **12** | 1 | 5 | {(2, 3), (5, 8), (10, 12), (20, 21), (16, 18)} | 5 | {(1, 2), (8, 10), (3, 4), (12, 14), (30, 33)} | -1 |
| **13** | 30 | 1 | {(1, 20)} | 1 | {(3, 5)} | -1 |
| **14** | 30 | 1 | {(2, 4)} | 5 | {(1, 2), (3, 4), (4, 5), (10, 13), (7, 8)} | -1 |
| **15** | 30 | 5 | {(1, 3), (4, 5), (6, 9), (20, 31), (34, 36)} | 1 | {(21, 26)} | -1 |
| **16** | 30 | 5 | {(1, 2), (3, 4), (4, 5), (10, 13), (7, 8)} | 5 | {(5, 7), (9, 10), (11, 13), (15, 18), (20, 21)} | -1 |
| **17** | 0 | | | | | “k trebuie sa fie intre 1 si 30” |
| **18** | 31 | | | | | “k trebuie sa fie intre 1 si 30” |
| **19** | 1 | 0 | | | | “n trebuie sa fie intre 1 si 5” |
| **20** | 30 | 0 | | | | “n trebuie sa fie intre 1 si 5” |
| **21** | 1 | 6 | | | | “n trebuie sa fie intre 1 si 5” |
| **22** | 30 | 6 | | | | “n trebuie sa fie intre 1 si 5” |

| | | | | | | |
|---|---|---|---|---|---|---|
| **23** | 1 | 1 | {(1,2)} | 0 | | “m trebuie sa fie intre 1 si 5” |
| **24** | 1 | 5 | {(1, 2), (3, 4), (5, 6), (10, 13), (7, 8)} | 0 | | “m trebuie sa fie intre 1 si 5” |
| **25** | 30 | 1 | {(3, 9)} | 0 | | “m trebuie sa fie intre 1 si 5” |
| **26** | 30 | 5 | {(1,4), (5, 6), (7, 9), (10, 13), (14, 20)} | 0 | | “m trebuie sa fie intre 1 si 5” |
| **27** | 1 | 1 | {(1,2)} | 6 | | “m trebuie sa fie intre 1 si 5” |
| **28** | 1 | 5 | {(1, 2), (3, 4), (5, 6), (10, 13), (7, 8)} | 6 | | “m trebuie sa fie intre 1 si 5” |
| **29** | 30 | 1 | {(3, 9)} | 6 | | “m trebuie sa fie intre 1 si 5” |
| **30** | 30 | 5 | {(1,4), (5, 6), (7, 9), (10, 13), (14, 20)} | 6 | | “m trebuie sa fie intre 1 si 5” |

```
@Test
public void boundaryValueAnalysis() throws IOException {
    File actual_file = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\src\\pachet1\\licentaOut.txt");
    //1
    File f1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input1.txt");
    tester.gaseste_interval(f1);
    File expected_file1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected1.txt");
    assertTrue("The files are different!", FileUtils.contentEquals(expected_file1,
actual_file));
```

```
    //2
    File f2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input2.txt");
    tester.gaseste_interval(f2);
    File expected_file2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected2.txt");
    assertTrue("The files are different!", FileUtils.contentEquals(expected_file2,
actual_file));

    //3
    File f3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input3.txt");
    tester.gaseste_interval(f3);
    File expected_file3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected3.txt");
    assertTrue("The files are different!", FileUtils.contentEquals(expected_file3,
actual_file));

    //4
    File f4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input4.txt");
    tester.gaseste_interval(f4);
    File expected_file4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected4.txt");
    assertTrue("The files are different!", FileUtils.contentEquals(expected_file4,
actual_file));

    //5
    File f5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input5.txt");
    tester.gaseste_interval(f5);
    File expected_file5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected5.txt");
    assertTrue("The files are different!", FileUtils.contentEquals(expected_file5,
```

```
actual_file));

    //6
    File f6 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input6.txt");
    tester.gaseste_interval(f6);
    File expected_file6 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected6.txt");
    assertTrue("The files are different!", FileUtils.contentEquals(expected_file6,
actual_file));

    //7
    File f7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input7.txt");
    tester.gaseste_interval(f7);
    File expected_file7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected7.txt");
    assertTrue("The files are different!", FileUtils.contentEquals(expected_file7,
actual_file));

    //8
    File f8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input8.txt");
    tester.gaseste_interval(f8);
    File expected_file8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected8.txt");
    assertTrue("The files are different!", FileUtils.contentEquals(expected_file8,
actual_file));

    //9
    File f9 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input9.txt");
    tester.gaseste_interval(f9);
    File expected_file9 = new File("D:\\An 3\\SEM 2\\Testare
```

```
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected9.txt");
    assertTrue("The files are different!", FileUtils.contentEquals(expected_file9,
actual_file));

    //10
    File f10 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input10.txt");
    tester.gaseste_interval(f10);
    File expected_file10 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected10.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file10, actual_file));

    //11
    File f11 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input11.txt");
    tester.gaseste_interval(f11);
    File expected_file11 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected11.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file11, actual_file));

    //12
    File f12 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input12.txt");
    tester.gaseste_interval(f12);
    File expected_file12 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected12.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file12, actual_file));

    //13
    File f13 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input13.txt");
```

```
    tester.gaseste_interval(f13);
    File expected_file13 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected13.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file13, actual_file));

    //14
    File f14 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input14.txt");
    tester.gaseste_interval(f14);
    File expected_file14 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected14.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file14, actual_file));

    //15
    File f15 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input15.txt");
    tester.gaseste_interval(f15);
    File expected_file15 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected15.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file15, actual_file));

    //16
    File f16 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input16.txt");
    tester.gaseste_interval(f16);
    File expected_file16 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected16.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file16, actual_file));

    //17
```

```
    File f17 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input17.txt");
    tester.gaseste_interval(f17);
    File expected_file17 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected17.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file17, actual_file));

    //18
    File f18 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input18.txt");
    tester.gaseste_interval(f18);
    File expected_file18 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected18.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file18, actual_file));

    //19
    File f19 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input19.txt");
    tester.gaseste_interval(f19);
    File expected_file19 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected19.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file19, actual_file));

    //20
    File f20 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input20.txt");
    tester.gaseste_interval(f20);
    File expected_file20 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected20.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file20, actual_file));
```

```
    //21
    File f21 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input21.txt");
    tester.gaseste_interval(f21);
    File expected_file21 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected21.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file21, actual_file));

    //22
    File f22 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input22.txt");
    tester.gaseste_interval(f22);
    File expected_file22 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected22.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file22, actual_file));

    //23
    File f23 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input23.txt");
    tester.gaseste_interval(f23);
    File expected_file23 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected23.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file23, actual_file));

    //24
    File f24 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input24.txt");
    tester.gaseste_interval(f24);
    File expected_file24 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected24.txt");
```

```
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file24, actual_file));

    //25
    File f25 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input25.txt");
    tester.gaseste_interval(f25);
    File expected_file25 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected25.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file25, actual_file));

    //26
    File f26 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input26.txt");
    tester.gaseste_interval(f26);
    File expected_file26 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected26.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file26, actual_file));

    //27
    File f27 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input27.txt");
    tester.gaseste_interval(f27);
    File expected_file27 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected27.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file27, actual_file));

    //28
    File f28 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input28.txt");
    tester.gaseste_interval(f28);
```

```
    File expected_file28 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected28.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file28, actual_file));

    //29
    File f29 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input29.txt");
    tester.gaseste_interval(f29);
    File expected_file29 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected29.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file29, actual_file));

    //30
    File f30 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_input\\input30.txt");
    tester.gaseste_interval(f30);
    File expected_file30 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\BA_expected\\expected30.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file30, actual_file));

}
```

**c) <u>Partiționarea în categorii</u>**

Cuprinde următorii pași:

1. *Descompune specificația funcțională în unități*: avem o singură unitate.
2. *Identificarea parametrilor*: k, n, m.
3. *Găsește categoriile* fiecărui parametru:
    - k: dacă are valoare validă între 1 și 30
    - n: dacă are valoare validă între 1 și 5
    - m: dacă are valoare validă între 1 și 5
4. *Partiționează fiecare categorie în alternative:*
    - k: <0, 0, 1…30, >30
    - n: <0, 0, 1, 2…5, >5
    - m: <0, 0, 1, 2…5, >5
5. *Scrie specificația de testare:*
    - *k:*
        1) {k | k < 0}
        2) 0
        3) 1…30 [ok, durata_medie]
        4) {k | k > 30}
    - *n:*
        1) {n | n < 0}
        2) 0
        3) 1 [ok, lungime 1]
        4) 2…5 [ok, lungime_medie]
        5) { n | n > 5}
    - *m:*
        1) {m | m < 0}
        2) 0
        3) 1 [ok, lungime 1]

4) 2…5 [ok, lungime_medie]
5) { m | m > 5}

Din specificația de testare ar trebui să rezulte 4 * 5 * 5 = 100 de cazuri de testare. Pe de altă parte, unele combinații nu au sens și pot fi eliminate. Acest lucru se poate realiza adăugând constrângeri acestor alternative. Constrângerile pot fi proprietăți ale alternativelor sau condiții de selecție bazate pe aceste proprietăți. În acest caz, alternativele vor fi combinate doar dacă condițiile de selecție sunt satisfăcute. Folosind acest procedeu voi rămâne cu 16 teste.

*6. Creează cazuri de testare(in număr de 16)*

k1 k2 k3n1 k3n2

k3n3m1 k3n3m2

k3n3m3 k3n3m4 k3n3m5

k3n4m1 k3n4m2 k3n4m3 k3n4m4 k3n4m5

k3n5 k4

*7. Creează date de test*

| | **Intrări** | | | | | **Rezultat final(expected)** |
|---|---|---|---|---|---|---|
| **Nr test** | **k** | **n** | **S** | **m** | **T** | |
| 1 | -2 | | | | | “k trebuie sa fie intre 1 si 30” |
| 2 | 0 | | | | | “k trebuie sa fie intre 1 si 30” |
| 3 | 3 | -2 | | | | “n trebuie sa fie intre 1 si 5” |
| 4 | 3 | 0 | | | | “n trebuie sa fie intre 1 si 5” |
| 5 | 2 | 1 | {(1,4)} | -2 | | “m trebuie sa fie intre 1 si 5” |
| 6 | 3 | 1 | {(5,9)} | 0 | | “m trebuie sa fie intre 1 si 5” |
| 7 | 2 | 1 | {(2,6)} | 1 | {(1,9)} | 2 4 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | 2 | 1 | {(3,6)} | 2 | {(1,2), (3, 6)} | 3 5 |
| 9 | 2 | 1 | {(10,20)} | 10 | | “m trebuie sa fie intre 1 si 5” |
| 10 | 3 | 2 | {(2,8), (10, 14)} | -3 | | “m trebuie sa fie intre 1 si 5” |
| 11 | 3 | 2 | {(2,10), (19, 23)} | 0 | | “m trebuie sa fie intre 1 si 5” |
| 12 | 3 | 3 | {(2,10), (19, 23), (1, 2)} | 1 | {(1, 10)} | 2 5 |
| 13 | 3 | 3 | {(2,9), (11, 12), (15, 20)} | 2 | {(1,6), (7, 20)} | 2 5 |
| 14 | 2 | 2 | {(1,4), (7, 29)} | 7 | | “m trebuie sa fie intre 1 si 5” |
| 15 | 3 | 7 | | | | “n trebuie sa fie intre 1 si 5” |
| 16 | 46 | | | | | “k trebuie sa fie intre 1 si 30” |

```
@Test
public void categoryPartitioning() throws IOException {
    File actual_file = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\src\\pachet1\\licentaOut.txt");
    //1
    File f1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input1.txt");
    tester.gaseste_interval(f1);
    File expected_file1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected1.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file1, actual_file));

    //2
    File f2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input2.txt");
    tester.gaseste_interval(f2);
    File expected_file2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected2.txt");
    assertTrue("The files are different!",
```

```
FileUtils.contentEquals(expected_file2, actual_file));

    //3
    File f3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input3.txt");
    tester.gaseste_interval(f3);
    File expected_file3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected3.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file3, actual_file));

    //4
    File f4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input4.txt");
    tester.gaseste_interval(f4);
    File expected_file4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected4.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file4, actual_file));

    //5
    File f5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input5.txt");
    tester.gaseste_interval(f5);
    File expected_file5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected5.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file5, actual_file));

    //6
    File f6 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input6.txt");
    tester.gaseste_interval(f6);
    File expected_file6 = new File("D:\\An 3\\SEM 2\\Testare
```

```
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected6.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file6, actual_file));

    //7
    File f7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input7.txt");
    tester.gaseste_interval(f7);
    File expected_file7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected7.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file7, actual_file));

    //8
    File f8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input8.txt");
    tester.gaseste_interval(f8);
    File expected_file8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected8.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file8, actual_file));

    //9
    File f9 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input9.txt");
    tester.gaseste_interval(f9);
    File expected_file9 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected9.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file9, actual_file));

    //10
    File f10 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input10.txt");
```

```
    tester.gaseste_interval(f10);
    File expected_file10 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected10.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file10, actual_file));

    //11
    File f11 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input11.txt");
    tester.gaseste_interval(f11);
    File expected_file11 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected11.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file11, actual_file));

    //12
    File f12 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input12.txt");
    tester.gaseste_interval(f12);
    File expected_file12 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected12.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file12, actual_file));

    //13
    File f13 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input13.txt");
    tester.gaseste_interval(f13);
    File expected_file13 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected13.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file13, actual_file));

    //14
```

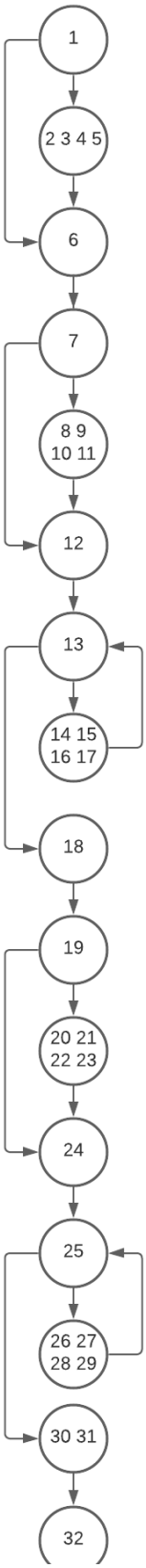
```
    File f14 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input14.txt");
    tester.gaseste_interval(f14);
    File expected_file14 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected14.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file14, actual_file));

    //15
    File f15 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input15.txt");
    tester.gaseste_interval(f15);
    File expected_file15 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected15.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file15, actual_file));

    //16
    File f16 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_input\\input16.txt");
    tester.gaseste_interval(f16);
    File expected_file16 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CP_expected\\expected16.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file16, actual_file));


}
```

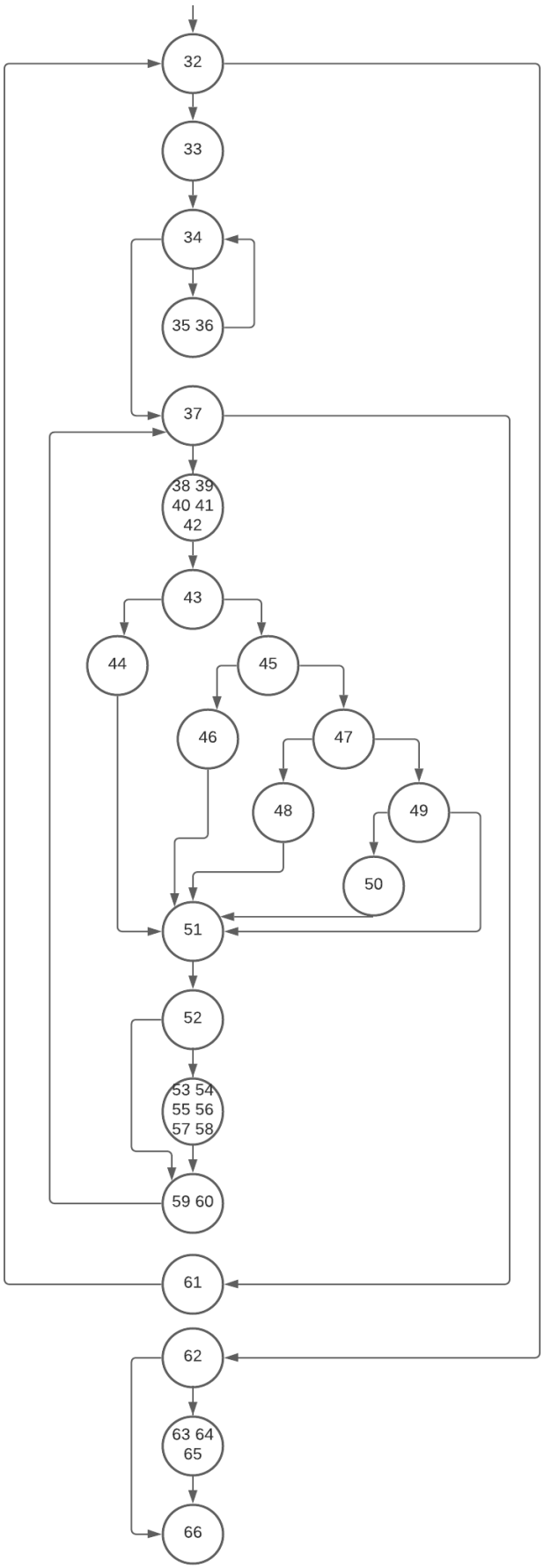
2. **Testare structurală**
   - Transformarea programului într-un graf orientat

```
   public class MyClass{
      public static void gaseste_interval(File f) throws IOException {

       Scanner myScanner = new Scanner(f);
       FileWriter myWriter = new FileWriter("src/pachet1/licentaOut.txt");
       int n, k, m, i, j;
       boolean ok = false;
       k = myScanner.nextInt();1
1      if(k < 1 || k > 30){
2          myWriter.write("k trebuie sa fie intre 1 si 30");
3          myWriter.close();
4          return;
5      }
6      n = myScanner.nextInt();
7      if(n < 1 || n > 5){
8          myWriter.write("n trebuie sa fie intre 1 si 5");
9          myWriter.close();
10         return;
11     }
12     List<Pair<Integer,Integer>> interval_mihaela = new ArrayList<>();
13     for(i = 0; i < n; i ++){
14         int x = myScanner.nextInt();
15         int y = myScanner.nextInt();
16         interval_mihaela.add(new Pair(x,y));
17     }
18     m = myScanner.nextInt();
19     if(m < 1 || m > 5){
20         myWriter.write("m trebuie sa fie intre 1 si 5");
21         myWriter.close();
22         return;
23     }
```

```
24      List<Pair<Integer,Integer>> interval_prof = new ArrayList<>();
25      for(i = 0; i < m; i ++){
26          int x = myScanner.nextInt();
27          int y = myScanner.nextInt();
28          interval_prof.add(new Pair(x,y));
29      }
30      Collections.sort(interval_mihaela, new Compare());
31      Collections.sort(interval_prof, new Compare());

32      for(i = 0; i < n && ok == false; i++){
33          j = 0;
34          while(j < m && interval_prof.get(j).getValue() <
    interval_mihaela.get(i).getKey() ) {
35              j++;
36          }
37          while(j < m && interval_prof.get(j).getKey() <=
    interval_mihaela.get(i).getValue() && !ok){
38              int x_m = interval_mihaela.get(i).getKey();
39              int y_m = interval_mihaela.get(i).getValue();
40              int x_p = interval_prof.get(j).getKey();
41              int y_p = interval_prof.get(j).getValue();
42              Pair<Integer,Integer> interval = new Pair<>(0,0);
43              if(x_m <= x_p && y_m >= y_p)
44                  interval = new Pair<>(x_p, y_p);
45              else if(x_m >= x_p && y_m <= y_p)
46                  interval = new Pair<>(x_m, y_m);
47              else if(x_m > x_p && y_m > y_p)
48                  interval = new Pair<>(x_m, y_p);
49              else if(x_m < x_p && y_m < y_p)
50                  interval = new Pair<>(x_p, y_m);
51              int lungime = interval.getValue() - interval.getKey();

52              if(lungime >= k){
53                  int a = interval.getKey();
```

```
54                  int b = interval.getKey() + k;
55                  myWriter.write(a + " " + b);
56                  myWriter.close();
57                  ok = true;
58              }
59              j++;
60          }
61      }
62      if(!ok) {
63          myWriter.write("-1");
64          myWriter.close();
65      }
66  }

    }
```


1
2 3 4 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


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

Pe baza grafului se pot defini diverse acoperiri:

- *Acoperirea la nivel de instrucțiune*: fiecare instrucțiune (nod al grafului) este parcursă măcar o data
- *Acoperirea la nivel de ramură*: fiecare ramură a grafului este parcursă măcar o dată
- *Acoperirea la nivel de condiție*: fiecare condiție individuală dintr-o decizie să ia atât valoarea de adevărat cât și valoarea fals
- *Acoperirea la nivel de cale*: fiecare cale din graf este parcursă măcar o dată

### a) Acoperirea la nivel de instrucțiune (statement coverage)

Pentru a obține o acoperire la nivel de instrucțiune, ne concentrăm asupra instrucțiunilor care sunt controlare de condiții (corespunzătoare ramificațiilor din graf)

| **Intrări** | | | | | **Rezultat afișat** | **Instrucțiuni parcurse** |
|---|---|---|---|---|---|---|
| **k** | **n** | **S** | **m** | **T** | | |
| 4 | 2 | {(1, 10), (11, 13)} | 2 | {(2,4), (5, 12)} | 5 9 | 1, 6, 7, 12, 13, 14…17, 18, 19, 24, 25, 26…29, 30, 31, 32, 33, 34, 36, 37, 38…42, 43, 44, 45, 47, 49, 50, 51, 52, 53…58, 59, 60, 61, 62, 66 |
| 30 | 1 | {(1, 20)} | 1 | {(3, 5)} | -1 | 1, 6, 7, 12, 13, 14…17, 18, 19, 24, 25, 26…29, 30, 31, 32, 33, 34, 37, 38…42, 43, 44, 51, 52, 59, 60, 61, 62, 63…65, 66 |

```
@Test
public void statementCoverage() throws IOException {
    File actual_file = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\src\\pachet1\\licentaOut.txt");

    //1
    File f1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\SC_input\\input1.txt");
    tester.gaseste_interval(f1);
    File expected_file1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\SC_expected\\expected1.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file1, actual_file));

    //2
    File f2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\SC_input\\input2.txt");
    tester.gaseste_interval(f2);
    File expected_file2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\SC_expected\\expected2.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file2, actual_file));


}
```

**b) Acoperire la nivel de decizie (decision coverage)**

- Generează date de test care testează cazruile când fiecare decizie este adevărată sau falsă.
- Avantaj: testează toate ramurile (inclusiv ramurile nule ale instrucțiunilor if/else)
- Dezavantaj: nu testează condițiile individuale ale fiecărei decizii

| | **Decizii** |
|---|---|
| 1 | if(k < 1 \|\| k > 30) |
| 2 | if(n < 1 \|\| n > 5) |
| 3 | for(i = 0; i < n; i++) |
| 4 | if(m < 1 \|\| m > 5) |
| 5 | for(i = 0; i < m; i++) |
| 6 | for (i = 0; i < n && ok == false; i++) |
| 7 | while(j < m && interval_prof.get(j).getValue() < interval_mihaela.get(i).getKey() ) |
| 8 | while(j < m && interval_prof.get(j).getKey() <= interval_mihaela.get(i).getValue()) |
| 9 | if(x_m <= x_p && y_m >= y_p) |
| 10 | else if(x_m >= x_p && y_m <= y_p) |
| 11 | else if(x_m > x_p && y_m > y_p) |
| 12 | else if(x_m < x_p && y_m < y_p) |
| 13 | if(lungime >= k) |
| 14 | if(!ok) |

| Intrări | | | | | Rezultat afișat | Decizii acoperite |
|---|---|---|---|---|---|---|
| **k** | **n** | **S** | **m** | **T** | | |
| -2 | | | | | “k trebuie sa fie intre 1 si 30” | 1 |
| 4 | 0 | | | | “n trebuie sa fie intre 1 si 5” | 2 |
| 4 | 1 | {(1, 2)} | 10 | | “m trebuie sa fie intre 1 si 5” | 3 4 |
| 2 | 1 | {(1, 3)} | 1 | {(1, 7)} | 1 3 | 3 5 6 8 10 13 |
| 2 | 1 | {(4, 9)} | 2 | {(1, 3), (5, 9)} | 5 7 | 3 5 6 7 8 12 13 |
| 2 | 1 | {(4, 8)} | 1 | {(5, 7)} | 5 7 | 3 5 6 8 9 13 |
| 2 | 1 | {(4, 8)} | 1 | {(3, 7)} | 4 6 | 3 5 6 8 11 13 |
| 2 | 1 | {(1, 9)} | 1 | {(10, 13)} | -1 | 3 5 6 14 |

```
@Test
public void decisionCoverage() throws IOException {
    File actual_file = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\src\\pachet1\\licentaOut.txt");

    //1
    File f1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_input\\input1.txt");
    tester.gaseste_interval(f1);
    File expected_file1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_expected\\expected1.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file1, actual_file));

    //2
    File f2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_input\\input2.txt");
    tester.gaseste_interval(f2);
    File expected_file2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_expected\\expected2.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file2, actual_file));

    //3
    File f3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_input\\input3.txt");
    tester.gaseste_interval(f3);
    File expected_file3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_expected\\expected3.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file3, actual_file));
```

```
    //4
    File f4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_input\\input4.txt");
    tester.gaseste_interval(f4);
    File expected_file4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_expected\\expected4.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file4, actual_file));

    //5
    File f5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_input\\input5.txt");
    tester.gaseste_interval(f5);
    File expected_file5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_expected\\expected5.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file5, actual_file));

    //6
    File f6 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_input\\input6.txt");
    tester.gaseste_interval(f6);
    File expected_file6 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_expected\\expected6.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file6, actual_file));

    //7
    File f7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_input\\input7.txt");
    tester.gaseste_interval(f7);
    File expected_file7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_expected\\expected7.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file7, actual_file));

    //8
    File f8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_input\\input8.txt");
    tester.gaseste_interval(f8);
    File expected_file8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\DC_expected\\expected8.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file8, actual_file));

}
```

**c) Acoperire la nivel de conditie (condition coverage)**

- Generează date de test astfel încât fiecare condiție individuală dintr-o decizie să ia atât valoarea adevărat cât și valoarea fals

| | **Decizii** | **Conditii individuale** |
|---|---|---|
| 1 | if(k < 1 \|\| k > 30) | k < 1, k > 30 |
| 2 | if(n < 1 \|\| n > 5) | n < 1, n > 5 |
| 3 | for(i = 0; i < n; i++) | i < n |
| 4 | if(m < 1 \|\| m > 5) | m < 1, m > 5 |
| 5 | for(i = 0; i < m; i++) | i < m |
| 6 | for (i = 0; i < n && ok == false; i++) | i < n, ok |
| 7 | while(j < m && interval_prof.get(j).getValue() < interval_mihaela.get(i).getKey() ) | j < m, interval_prof.get(j).getvalue() < interval_mihaela.get(i).getkey() |
| 8 | while(j < m && interval_prof.get(j).getKey() <= interval_mihaela.get(i).getValue() && !ok) | j < m, interval_prof.get(j).getkey() <= interval_mihaela.get(i).getvalue(), ok |
| 9 | if(x_m <= x_p && y_m >= y_p) | x_m <= x_p, y_m >= y_p |
| 10 | else if(x_m >= x_p && y_m <= y_p) | x_m >= x_p && y_m <= y_p |
| 11 | else if(x_m > x_p && y_m > y_p) | x_m > x_p && y_m > y_p |
| 12 | else if(x_m < x_p && y_m < y_p) | x_m < x_p && y_m < y_p |
| 13 | if(lungime >= k) | lungime >= k |
| 14 | if(!ok) | ok |

| **Intrări** | | | | | **Rezultat afișat** | **Conditii individuale acoperite** |
|---|---|---|---|---|---|---|
| **k** | **n** | **S** | **m** | **T** | | |
| -2 | | | | | “k trebuie sa fie intre 1 si 30” | 1.1 |
| 40 | | | | | “k trebuie sa fie intre 1 si 30” | 1.2 |
| 4 | 0 | | | | “n trebuie sa fie intre 1 si 5” | 2.1 |
| 2 | 10 | | | | “n trebuie sa fie intre 1 si 5” | 2.2 |
| 4 | 1 | {(1, 2)} | -2 | | “m trebuie sa fie intre 1 si 5” | 4.1 |
| 4 | 1 | {(1, 2)} | 10 | | “m trebuie sa fie intre 1 si 5” | 4.2 |
| 2 | 1 | {(1, 3)} | 1 | {(1, 7)} | 1 3 | 3 5 6 8 9 |
| 2 | 1 | {(4, 9)} | 2 | {(1, 3), (5, 9)} | 5 7 | 3 5 6 7 8 12 13 |
| 2 | 1 | {(4, 8)} | 1 | {(5, 7)} | 5 7 | 3 5 6 8 9 13 |
| 2 | 1 | {(4, 8)} | 1 | {(3, 7)} | 4 6 | 3 5 6 8 11 13 |
| 2 | 1 | {(1, 9)} | 1 | {(10, 13)} | -1 | 3 5 6 14 |

```
@Test
public void conditionCoverage() throws IOException {

    File actual_file = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\src\\pachet1\\licentaOut.txt");

    //1
    File f1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input1.txt");
    tester.gaseste_interval(f1);
    File expected_file1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected1.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file1, actual_file));

    //2
    File f2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input2.txt");
    tester.gaseste_interval(f2);
    File expected_file2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected2.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file2, actual_file));

    //3
    File f3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input3.txt");
    tester.gaseste_interval(f3);
    File expected_file3 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected3.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file3, actual_file));

    //4
```

```
    File f4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input4.txt");
    tester.gaseste_interval(f4);
    File expected_file4 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected4.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file4, actual_file));

    //5
    File f5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input5.txt");
    tester.gaseste_interval(f5);
    File expected_file5 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected5.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file5, actual_file));

    //6
    File f6 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input6.txt");
    tester.gaseste_interval(f6);
    File expected_file6 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected6.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file6, actual_file));

    //7
    File f7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input7.txt");
    tester.gaseste_interval(f7);
    File expected_file7 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected7.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file7, actual_file));
```

```
    //8
    File f8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input8.txt");
    tester.gaseste_interval(f8);
    File expected_file8 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected8.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file8, actual_file));

    //9
    File f9 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input9.txt");
    tester.gaseste_interval(f9);
    File expected_file9 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected9.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file9, actual_file));

    //10
    File f10 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input10.txt");
    tester.gaseste_interval(f10);
    File expected_file10 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected10.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file10, actual_file));

    //11
    File f11 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_input\\input11.txt");
    tester.gaseste_interval(f11);
    File expected_file11 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\CC_expected\\expected11.txt");
```

```
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file11, actual_file));
}
```

### 3. Testarea circuitelor independente

Pentru un graf complet conectat *G* cu *e* arce si *n* noduri, numărul de circuite linear independente este dat de formula lui McCabe: V(G) = e – n + 1.

În cazul nostru, adăugam un arc de la nodul 66 la nodul 1:

$$\left.\begin{matrix} n = 37\ de\ noduri \\ e = 51\ de\ muchii \end{matrix}\right\} \Rightarrow V(G) = 15$$

Deci, *Complexitatea Ciclomatică* este V(G) = 15.

- **Circuite independente**

a) 13, 14…17, 13
b) 25, 26…29, 25
c) 34, 35 36, 34
d) 37, 38…42, 43, 44, 51, 52, 59 60, 37
e) 37, 38…42, 43, 44, 51, 52, 53…58, 59 60, 37
f) 37, 38…42, 43, 45, 46, 51, 52, 59 60, 37
g) 37, 38…42, 43, 45, 46, 51, 52, 53…58, 59 60, 37
h) 37, 38…42, 43, 45, 47, 48, 51, 52, 59 60, 37
i) 37, 38…42, 43, 45, 47, 48, 51, 52, 53…58, 59 60, 37
j) 37, 38…42, 43, 45, 47, 49, 50, 51, 52, 59 60, 37
k) 37, 38…42, 43, 45, 47, 49, 50, 51, 52, 53…58, 59 60, 37
l) 37, 38…42, 43, 45, 47, 49, 51, 52, 59 60, 37
m) 37, 38…42, 43, 45, 47, 49, 51, 52, 53…58, 59 60, 37
n) 32, 33, 34, 37, 61, 32
o) 1, 6, 7, 12, 13, 18, 19, 24, 25, 30 31, 32, 62, 66, 1
p) 1, 2…5, 6, 7, 12, 13, 18, 19, 24, 25, 30, 31, 32, 62, 66, 1
q) 1, 2…5, 6, 7, 8…11, 12, 13, 18, 19, 24, 25, 30, 31, 32, 62, 66, 1

## 4. Testarea la nivel de cale

Am facut câteva notații pentru a urmări mai ușor pe grafic:

- 13.(14:17.13)* = a
- 25.(26:29.25)* = b
- 34.(35:36.34)*= c
- 37.(38:42.43.(44.51 + 45.(46.51 + 47.(48.51 + 49.(50.51 + 51)))).52.(53:58.59:60 + 59.60).37)* = d

Obținem *expresia regulată*:

1.(2:5.6+6).7.(8:10.12+12).a.18.19.(20:23.24+24).b.30:31.32.(33.c.d.61.32)*.62.(63:65.66+66)

Pentru n = 0 și n = 1 avem:

- a = 13.(14:17.13 + null)
- b = 25.(26:29.25 + null)
- c = 34.(35:36.34 + null)
- d = 37.(38:42.43.(44.51 + 45.(46.51 + 47.(48.51 + 49.(50.51 + 51)))).52.(53:58.59:60 + 59.60).37 + null)

Obținem expresia regulată:

1.(2:5.6+6).7.(8:10.12+12).a.18.19.(20:23.24+24).b.30:31.32.(33.c.d.61.32+null).62.(63:65.66+66)

Calculăm *numărul de căi:*

a = 1*(1 + 1) = 2

b = 1*(1 + 1) = 2

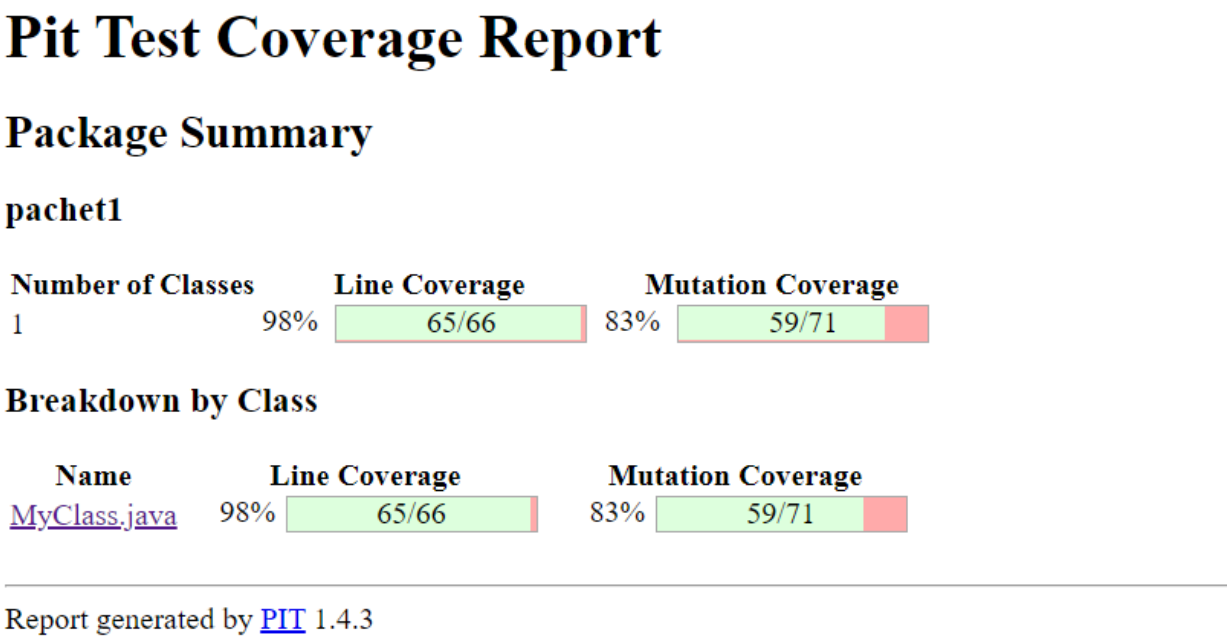
c = 1*(1 + 1) = 2

d = 1*(1*(1+1*(1+1*(1+1*(1+1))))*1*(1+1)*1+1) = 1*(1*(1+1*(1+1*(1+1*2)))*1*2*1+1) = 11

Nr căi = 1*(1+1)*1*(1+1)*1*(1+1)*1*(1+1)*1*(1+1)*1*(1*1*(1+1)*11 + 1)*1*(1+1) = 1472

**5. Genearea mutanților**

Am rulat seturile de test de la punctele 1, 2 și 3 contra mutanților generați și am obținut raportul:



# Pit Test Coverage Report

## Package Summary

### pachet1

| Number of Classes | Line Coverage | | Mutation Coverage | |
|---|---|---|---|---|
| 1 | 98% | 65/66 | 83% | 59/71 |

### Breakdown by Class

| Name | Line Coverage | | Mutation Coverage | |
|---|---|---|---|---|
| MyClass.java | 98% | 65/66 | 83% | 59/71 |

Report generated by PIT 1.4.3

Au supraviețuit mutanți, deoarece testele nu acopereau anumite situații. Una dintre ele este cea în care, dacă intervalele Mihaelei nu erau ordonate, primul interval de intersecție valid găsit nu era și cel mai mic. O altă situație neacoperită este cea în care capetele din stânga erau egale, iar capătul din dreapta al intervalului Mihaelei era mai mare decât capătul din dreapta al Decanului. Am dat următoarele date de test:

| **Intrări** | | | | | **Rezultat afișat** |
|---|---|---|---|---|---|
| **k** | **n** | **S** | **m** | **T** | |
| 3 | 3 | {(5, 9), (1, 4), (10, 12)} | 3 | {(1, 4), (5, 9), (11, 15)} | 1 4 |
| 4 | 1 | {(1, 8)} | 1 | {(1, 7)} | 1 5 |

```
@Test
public void killMutants() throws IOException {
    File actual_file = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\src\\pachet1\\licentaOut.txt");

    //1
    File f1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\Mutant_input\\input1.txt");
    tester.gaseste_interval(f1);
    File expected_file1 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\Mutant_expected\\expected1.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file1, actual_file));

    //2
    File f2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\Mutant_input\\input2.txt");
    tester.gaseste_interval(f2);
    File expected_file2 = new File("D:\\An 3\\SEM 2\\Testare
software\\PROIECT\\tests\\pachet2\\Mutant_expected\\expected2.txt");
    assertTrue("The files are different!",
FileUtils.contentEquals(expected_file2, actual_file));

}
```

Am obținut în final acest raport:

# Pit Test Coverage Report

## Project Summary

| Number of Classes | Line Coverage | | Mutation Coverage | |
|---|---|---|---|---|
| 1 | 98% | 65/66 | 86% | 61/71 |

## Breakdown by Package

| Name | Number of Classes | Line Coverage | | Mutation Coverage | |
|---|---|---|---|---|---|
| pachet1 | 1 | 98% | 65/66 | 86% | 61/71 |

Report generated by PIT 1.4.3