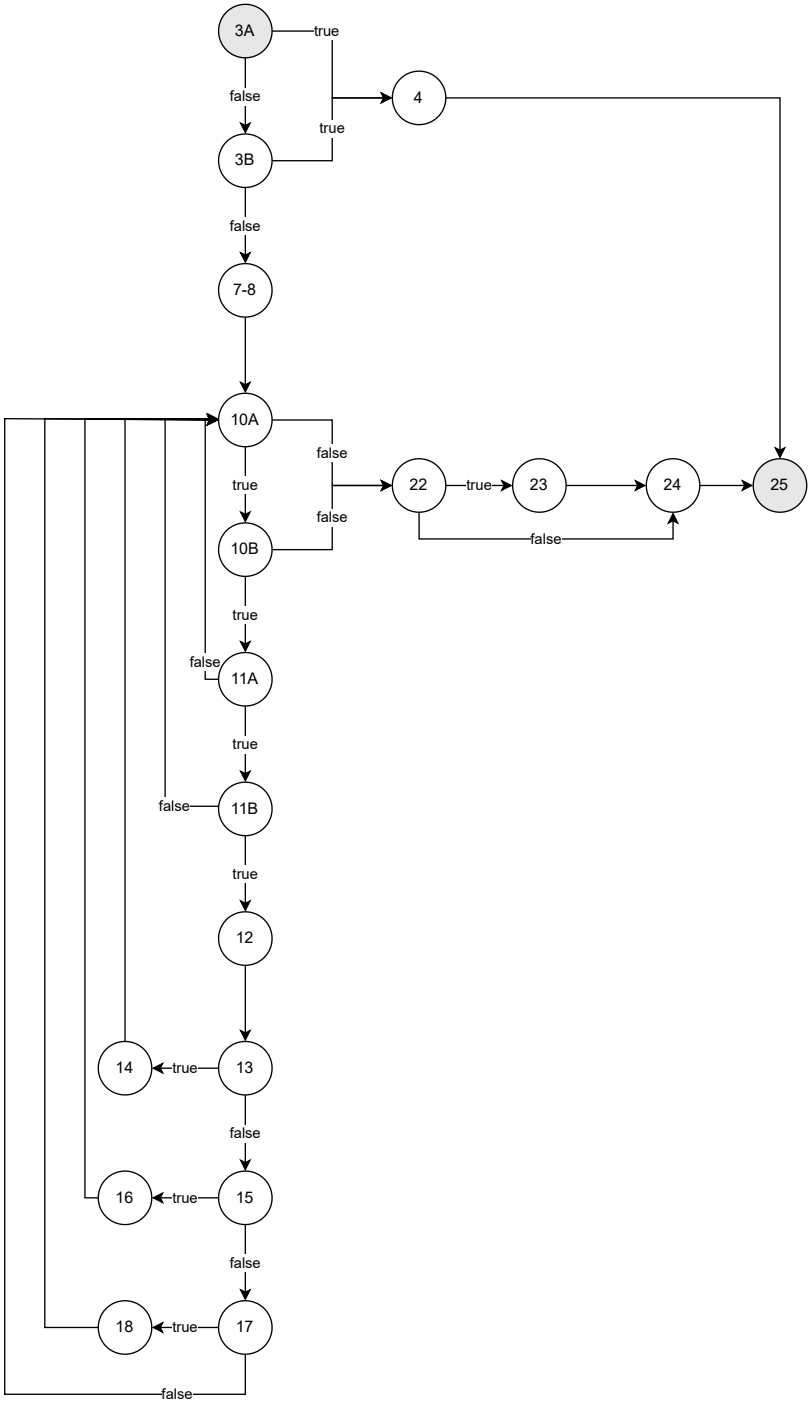

Exercici 1:

Graf del flux de Control

Important:

- un únic node inicial.
- un únic node final, tot i que tenim dues sortides (excepció i retorn) cal posar un node fictici final.
- utilitzar fletxes per indicar el sentit del flux.
- és la línia 10 (for) qui indica quan és surt del bucle.

Complexitat Ciclomàtica

Hi ha 10 nodes predicats (més d'una fletxa de sortida) i també hi ha 10 àrees tancades.

Complexitat és 10+1=11

Camins de 0 voltes

Camí A: 3A → 4 → 25
Camí B: 3A → 3B → 4 → 25
Camí C: 3A → 3B → 7-8 → 10A → 22 → 23 → 24 → 25
El camí de 0 voltes on 10B sigui fals no és lògic, el descartem.

Camins d'1 volta

Detallo només els camins lògics del punt de vista del codi, no només considerant el graf.
Hi ha dos camins d'1 volta que no complexin el condicional 11:
Camí D: 3A → 3B → 7-8 → 10A → 10B → 11A → 10A → 22 → 23 → 24 → 25
Camí E: 3A → 3B → 7-8 → 10A → 10B → 11A → 11B → 10A → 22 → 23 → 24 → 25
De camí d'una volta que complexi el condicional 11 sempre complirà el condicional 13:
Camí F: 3A → 3B → 7-8 → 10A → 10B → 11A → 11B → 12 → 13 → 14 → 10A → 22 → 23 → 24 → 25
Observem que no pot existir cap camí d'1 volta on 10B i 13 siguin falsos.

Exercici 2:

Conjunt Bàsic

Important, hem vist que hi ha arrestes per on no hem passat amb només una o cap volta, per això anem a buscar més camins:

Busquem un camí que doni dues voltes i passi per l'aresta 13 fals.

Camí G: 3A → 3B → 7-8 → 10A → 10B → 11A → 11B → 12 → 13 → 14 →

10A → 10B → 11A → 11B → 12 → 13 → 15 → 16 →10A → 22 → 23 → 24 → 25

Busquem un camí que doni tres voltes i que passi per les arestes 17 true, 10B false i 22 true.

Camí H: 3A → 3B → 7-8 → 10A → 10B → 11A → 11B → 12 → 13 → 14 →

10A → 10B → 11A → 11B → 12 → 13 → 15 → 16 →

10A → 10B → 11A → 11B → 12 → 13 → 15 → 17 → 18 →10A → 10B → 22 → 24 → 25

El conjunt bàsic és el mínim nombre de camins que donin cobertura màxima (passar per totes les arestes), però en aquest cas no existeix cap camí lògic que 17 sigui falsa, perquè necessitaríem fer 4 voltes i 10B no ens permet fer-ho. Aquí caldria observar que 10B és incorrecte, que si volem els mínims valors de l'array i si aquesta no està ordenada, cal observar tots els valors.

Conjunt Bàsic està representat per 4 camins, dos camins de 0 voltes on 3A true i 3B true, i dos camins que seran el camí H i un camí que passi per 10A false i 12 true (que poden ser els camins G, F, E, D o C).

per exemple: CB = { camí A, camí B, camí H, camí C}

Camins de 0 voltes ={camí A, camí B, camí C}

Conjunt de Casos de Prova = { camí A, camí B, camí H, camí C}

Si heu escollit un altre CB, el conjunt de Casos de Prova serà de 5 camins.

File - MultiplesCorrect.java

```java
package CapaDomini;

import java.util.Arrays;

public class MultiplesCorrect {
    public static int[] trobaTresMultiplesMinims(int divisor, int[] nombres
) {
        if (nombres == null || divisor <= 0) {
            throw new IllegalArgumentException("Paràmetre d'entrada null o
divisor no vàlid");
        }

        int[] resultats = {Integer.MAX_VALUE, Integer.MAX_VALUE, Integer.
MAX_VALUE};
        int count = 0;

        /* ERROR: suposem que l'array nombres no està ordenat i cal eliminar
count < 3*/

        for (int i = 0; i < nombres.length /*&& count < 3*/; i++) {
            if (nombres[i] >= 0 && nombres[i] % divisor == 0) {  // Comprova
que sigui positiu i múltiple
                count++;
                if( nombres[i] < resultats[0] ) {
                    resultats[2] = resultats[1]; //ERROR: cal que resultats
estigui ordenat
                    resultats[1] = resultats[0];
                    resultats[0] = nombres[i];
                }else if (nombres[i] < resultats[1] ) {
                    resultats[2] = resultats[1];
                    resultats[1] = nombres[i];
                }else if (nombres[i] < resultats[2] )
                    resultats[2] = nombres[i];
            }
        }

        if( count < 3)
            resultats = null; // Si no troba 3 múltiples, retorna null
        return resultats; //ERROR: calia retornar resultats
    }

}

```

File - MultiplesTest.java

```java
package CapaDomini;

import org.junit.jupiter.api.Test;

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

class MultiplesTest {
    /* Conjunt Bàsic està representat per 4 camins:
     * dos camins de 0 voltes on 3A true i 3B true,
     * i dos camins que seran el camí H i un camí que passi per 10A false i 12 true
     * (que poden ser els camins G, F, E, D o C).
     * Conjunt de Casos de Prova = {camí A, camí B, camí H, camí C}
     * No passa res si el vostre Conjunt de Caos de Prova és diferent.
     */
    /* IMPORTANT:
     * A continuació es fa la prova unitària de tots els camins,
     * però només cal fer-ne els escollits al Conjunt de Casos de Prova
     */

    @Test
    public void test_camiA()
    {
        /* Camí A: 3A → 4 → 25
         * Entrada:
         * - divisor = 2
         * - nombres = null
         * sortida: IllegalArgumentException
         */
        int divisor = 2;
        assertThrows(IllegalArgumentException.class, ()->MultiplesCorrect.trobaTresMultiplesMinims(divisor,null));
    }

    @Test
    public void test_camiB()
    {
        /* Camí B: 3A → 3B → 4 → 25
         * Entrada:
         * - divisor = -1
         * - nombres = {}
         * sortida: IllegalArgumentException
         */
        int divisor = -1;
        int[] nombres = {};
        assertThrows(IllegalArgumentException.class, ()->MultiplesCorrect.trobaTresMultiplesMinims(divisor,nombres));
    }

    @Test
    public void test_camiC()
    {
        /* Camí C: 3A → 3B → 7-8 → 10A → 22 → 23 → 24 → 25
         * Entrada:
         * - divisor = 2
         * - nombres = {}
         * sortida: null
         */
        int divisor = 2;
        int[] nombres = {};
```

```
58          assertNull(MultiplesCorrect.trobaTresMultiplesMinims(divisor,nombres
    ));
59          // aquí trobaríem un error: el que retorna.
60      }
61
62      @Test
63      public void test_camiD()
64      {
65          /* Camí D: 3A → 3B → 7-8 → 10A → 10B → 11A → 10A → 22 → 23 → 24 → 25
66           * Entrada:
67           * - divisor = 2
68           * - nombres = {-1}
69           * sortida: null
70           */
71          int divisor = 2;
72          int[] nombres = {-1};
73          assertNull(MultiplesCorrect.trobaTresMultiplesMinims(divisor,nombres
    ));
74      }
75      @Test
76      public void test_camiE()
77      {
78          /* Camí E: 3A → 3B → 7-8 → 10A → 10B → 11A → 11B →10A →  22 → 23 →
    24 → 25
79           * Entrada:
80           * - divisor = 2
81           * - nombres = {3}
82           * sortida: null
83           */
84          int divisor = 2;
85          int[] nombres = {3};
86          assertNull(MultiplesCorrect.trobaTresMultiplesMinims(divisor,nombres
    ));
87      }
88
89      @Test
90      public void test_camiF()
91      {
92          /* Camí F: 3A → 3B → 7-8 → 10A → 10B → 11A → 11B → 12 → 13 → 14 →
    10A → 22 → 23 → 24 → 25
93           * Entrada:
94           * - divisor = 2
95           * - nombres = {4}
96           * sortida: null
97           */
98          int divisor = 2;
99          int[] nombres = {4};
100         assertNull(MultiplesCorrect.trobaTresMultiplesMinims(divisor,nombres
    ));
101         //aquest camí és molt inútil perquè no podem comprovar que es guarda
     realment a 14.
102     }
103
104     @Test
105     public void test_camiG()
106     {
107         /* Camí G: 3A → 3B → 7-8 → 10A → 10B → 11A → 11B → 12 → 13 → 14 →
108                           10A → 10B → 11A → 11B → 12 → 13 → 15 → 16 → 10A →
    22 → 23 → 24 → 25
109          * Entrada:
```

```
110          * - divisor = 2
111          * - nombres = {4,6}
112          * sortida: null
113          */
114         int divisor = 2;
115         int[] nombres = {4,6};
116         assertNull(MultiplesCorrect.trobaTresMultiplesMinims(divisor,nombres
    ));
117         //aquest camí és molt inútil perquè no podem comprovar que es guarda
     realment a 14 i 16.
118     }
119 
120     @Test
121     public void test_camiH()
122     {
123         /* Camí H: 3A → 3B → 7-8 → 10A → 10B → 11A → 11B → 12 → 13 → 14 →
124                           10A → 10B → 11A → 11B → 12 → 13 → 15 → 16 →
125                           10A → 10B → 11A → 11B → 12 → 13 → 15 → 17 → 18 →
    10A → 10B → 22 →  24 → 25
126          * Entrada:
127          * - divisor = 2
128          * - nombres = {4,6,8}
129          * sortida: null
130          */
131         int divisor = 2;
132         int[] nombres = {8,4,6};
133         int[] esperat = {4,6,8};
134         assertArrayEquals(esperat, MultiplesCorrect.trobaTresMultiplesMinims
    (divisor,nombres));
135         assertArrayEquals(esperat, MultiplesCorrect.
    trobaTresMultiplesMinims_opcio2(divisor,nombres));
136         // aquí trobem errors, faig dues propostes de codi per arreglar-ho.
137     }
138 
139 }
```