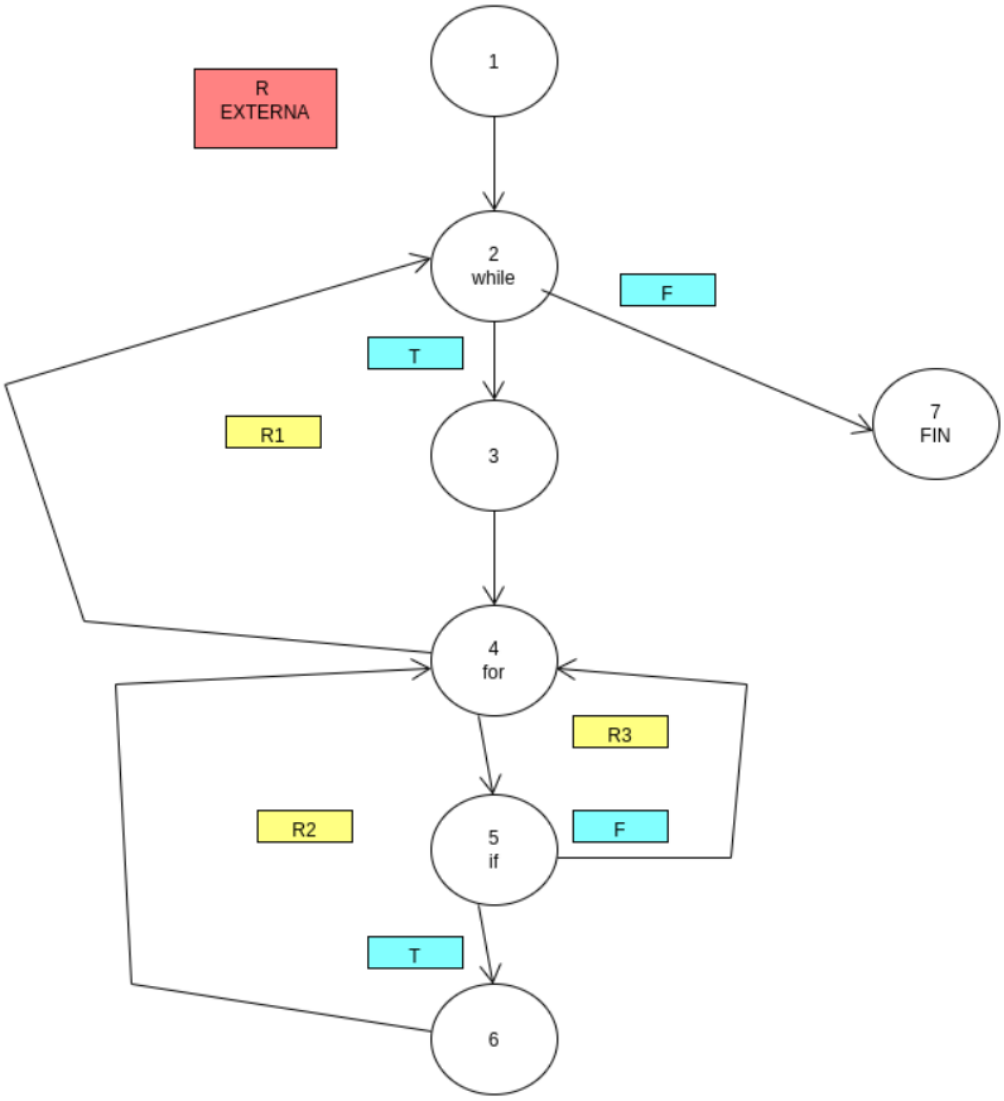
Soluciones Extraordinario Junio de Entornos de Desarrollo. Curso 2019/20.

1a) Cualquier fallo del tipo que falte una condición o esté mal configurado el flujo o la estructura for, while o if hacen que el ejercicio sea incorrecto. Si faltase el nodo fin o inicial, podría tenerse en cuenta el resto si está correcto.



2b)

V(G) = Nº regiones + 1 externa = 3 + 1 = 4

V(G) = Condiciones + 1 = 3 + 1 = 4

V(G) = aristas - nodos + 2 = 9 - 7 + 2 = 4

Esta es una solución que viene del grafo correcto . Si el grafo está incorrecto, no valdría una solución que diese lo mismo, pues el grafo seguiría estando incorrecto.

2c)

Camino | Paso por nodos

==========

1 | 1 2 3 4 5 6 4 2 7

2 | 1 2 3 4 5 4 2 7

3 | 1 2 3 4 2 7

4 | 1 2 7

2d)

Camino | hacerMas | n | x[i] > x [i + 1] | Resultado esperado

==========

1 | true | i < n | elemento es mayor que el siguiente | realizamos el cambio de elementos

2 | true | i < n | elemento es menor que el siguiente | no hay cambios de elementos

3 | false | i >= n | sale del for | no hay cambios de elementos

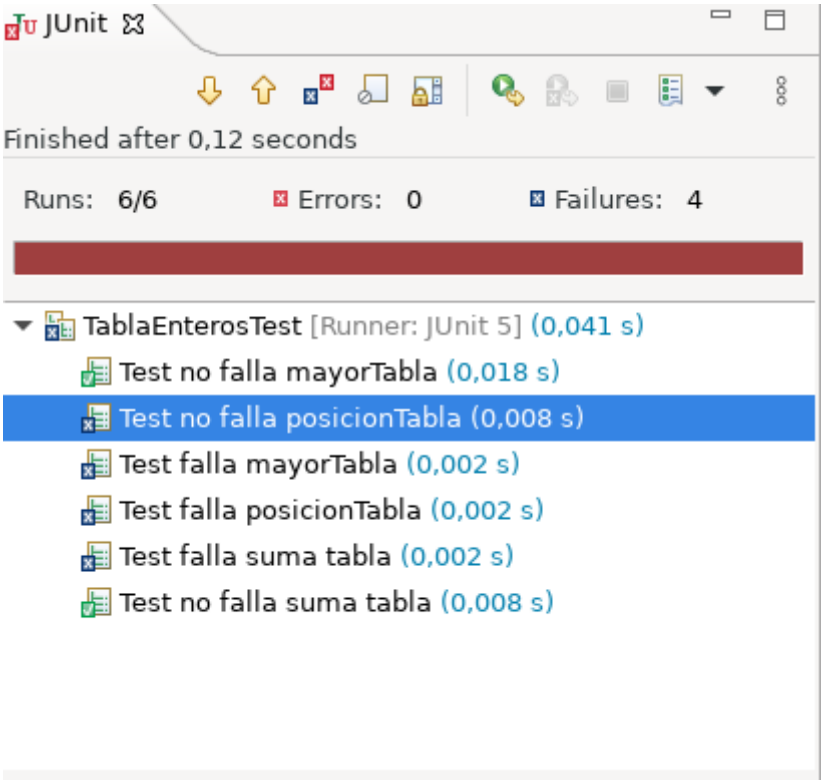
4 | false | ------- | | no hay cambios de elementos

3d) Se crean los siguientes Tests. Se crea primero un objeto vector v que tenga una serie de elementos. sumaTabla suma todos los elementos, pues creamos entonces dos pruebas, una para que la sume falle y otro para que no falle y comprobamos que está bien.

Igualmente lo hacemos para el método mayorTabla que halla el mayor del vector. Utilizamos también dos valores uno que falle y otro que no y las pruebas salen correctas.

Mientras que en el método de posicionTabla sí que vemos que hay fallo cuando equiparamos a un valor que va a fallar pero también falla en caso de que sea correcto. El número 2 debe de devolver la posición 1. Y no es así. Con lo cual detectamos un fallo. Esto quiere decir que puede existir un problema de codificación del código internamente. Vemos que el problema es en la línea de código

`if (tabla[i] != n)` que deberíamos cambiar con `if (tabla[i] == n)` para que no falle en la prueba válida.




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

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

class TablaEnterosTest {

    Integer v[] = {1,2,3,4,5,6,7};

    TablaEnteros tabla = new TablaEnteros(v);

    @DisplayName("Test no falla suma tabla")
    @Test
    void testSumaTabla() {

        assertEquals(28, tabla.sumaTabla());

    }

    @DisplayName("Test falla suma tabla")
    @Test
    void testSumaTabla2() {

        assertEquals(29, tabla.sumaTabla());

    }

    @DisplayName("Test no falla mayorTabla")
    @Test
    void testMayorTabla() {
        assertEquals(7, tabla.mayorTabla());
    }

    @DisplayName("Test falla mayorTabla")
    @Test
```

```
    void testMayorTabla2() {
        assertEquals(8, tabla.mayorTabla());
    }

    @DisplayName("Test no falla posicionTabla")
    @Test
    void testPosicionTabla() {

        assertEquals(1, tabla.posicionTabla(2));
    }
    @DisplayName("Test falla posicionTabla")
    @Test
    void testPosicionTabla2() {

        assertEquals(1, tabla.posicionTabla(2));
    }

}
```

4. Aquí debe de declararse los paquetes e importarlos en las respectivas clases para que pueda compilarse de forma correcta por terminal.

```
package com.gestion;
import java.util.*;

public class Gestion {

   public void print(){
       System.out.println("Departamento Gestion");
   }

}

package com.marketing;
import java.util.*;

import com.gestion.Gestion;

public class Marketing  {

   public void print() {

      System.out.println("Departamento Marketing");
   }

}
package com.it;
import java.util.*;

import com.gestion.Gestion;

public class It {

   public void print() {

      System.out.println("Departamento IT");
   }
```

```
}
package com.main;

import com.it.It;
import com.gestion.Gestion;
import com.marketing.Marketing;

public class Main {

    public static void main (String [] args) {

         It it = new It();
         Gestion gestion = new Gestion();
         Marketing marketing = new Marketing();

         it.print();
         gestion.print();
         marketing.print();
    }

}
```

```
$ javac com/main/Main.java
$ tree com
com
├── gestion
│   ├── Gestion.class
│   └── Gestion.java
├── it
│   ├── It.class
│   └── It.java
├── main
│   ├── Main.class
│   └── Main.java
└── marketing
    ├── Marketing.class
    └── Marketing.java

4 directories, 8 files
$ java com.main.Main
Departamento IT
Departamento Gestion
Departamento Marketing
```

5. La solución es la siguiente, sin errores y se debe ejecutar perfectamente el Main.jar. También se debe mostrar el contenido y luego extraer los ficheros

```
$ jar -cmf META-INF/MANIFEST.MF Main.jar com/marketing/*.class com/it/*.class
com/gestion/*.class com/main/*.class
$ java -jar Main.jar
Departamento IT
Departamento Gestion
Departamento Marketing
$ jar -tf Main.jar
META-INF/
```

```
META-INF/MANIFEST.MF
com/marketing/Marketing.class
com/it/It.class
com/gestion/Gestion.class
com/main/Main.class
~/Descargas $ jar xf Main.jar
$ tree...
```