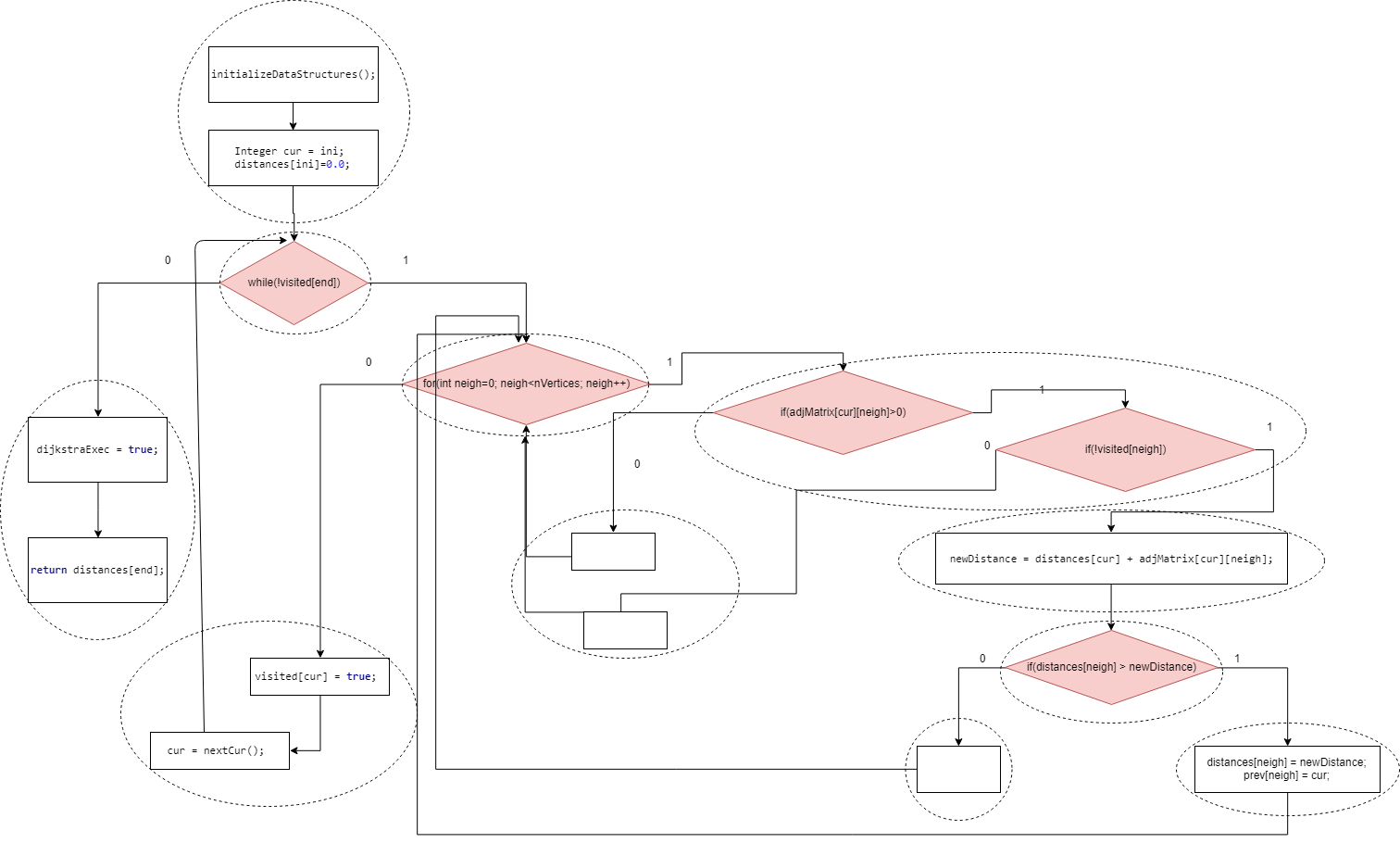
Hemos usado trazas, que son pequeños puntos de control que se añaden al código para saber por qué línea ha entrado la función al ejecutarse.

Clase Dijkstra



Complejidad: Rombos + 1 = Areas = 5

Camino 1: La función no entra por el while inicial

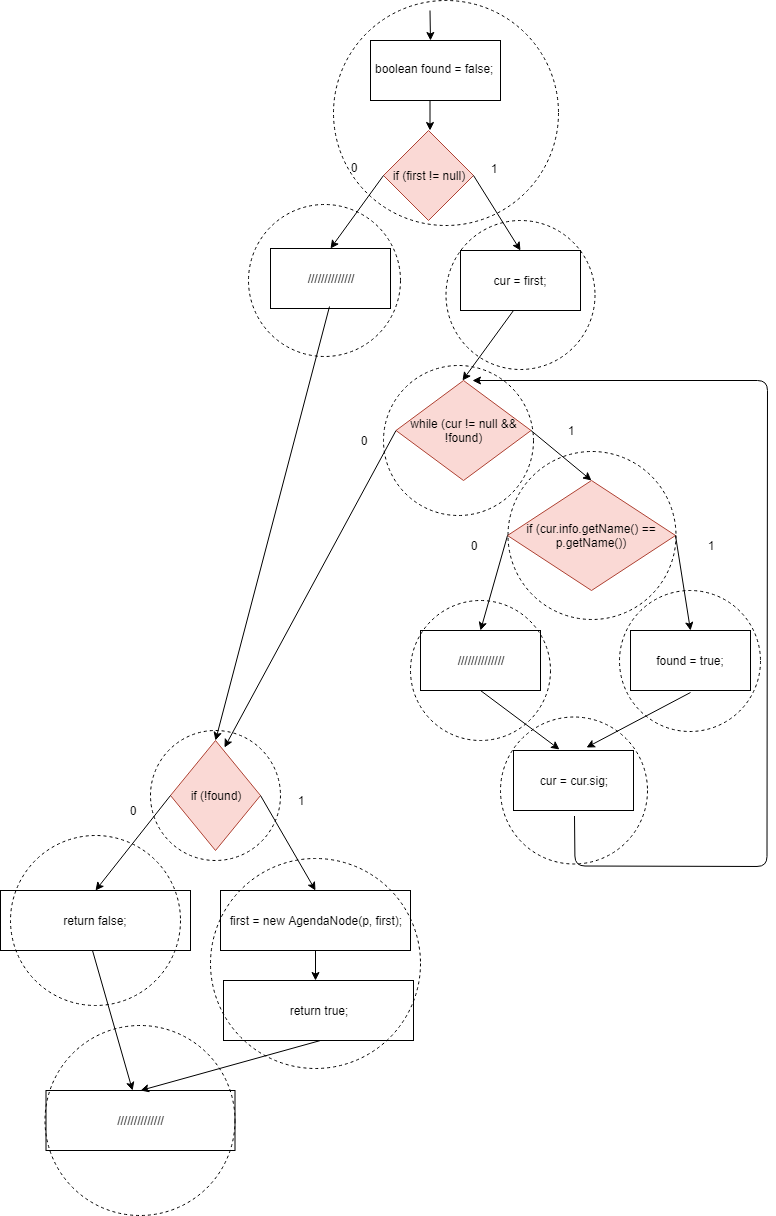
Camino 2: La función entra por el while inicial pero no por el for siguiente

Camino 3: La función entra por el while inicial, el for pero no entra por el if

Camino 4: La función entra por el while inicial, el for, el if que hay a continuación y después sin entrar por el último if.

Camino 5: La función entra por el while inicial, el for, el if que hay a continuación y después entra por el último if.

Clase Agenda



En la clase Agenda hemos añadido el String traza que sirve para hacer las trazas del código para las pruebas, hemos creado el nuevo método setFirst, que recibe por parámetro un AgendaNode, lo que hará será asignar al primer AgendaNode el que le pases por parámetro.

Para la realización de las pruebas, hemos utilizado assertEquals, comparando lo que tendría que dar la traza de la ejecución de esa prueba con la obtenida, a través del nuevo método getTraza, que te devuelve el valor de la traza.

En el método addEntry, hemos modificado lo que recibe por parámetro, añadiendo un nuevo parámetro de tipo booleano, que será el valor de found, y modificando la interface con el parámetro nuevo, en lugar de inicializar siempre found a false. Por último, para los casos de prueba CP31 y CP33, necesitábamos que entrara en la condición if (first != null), y que después cuando se llegara al while first valiera null, esto lo hemos conseguido creando una nueva condición, que comprueba si el nombre de la entrada que queremos añadir es CP31 o CP33 para saber si estamos en ese caso de prueba, y entonces asigna a first el valor null.

Complejidad: Rombos + 1 = Areas = 5

**Camino 1:** La función no entra en if (first != null) y no entra en if(!found)

**Camino 2:** La función no entra en if (first != null) y entra en if(!found)

**Camino 3**

En este camino al tratarse de un while con 2 condiciones, para asegurarnos del funcionamiento total de este while con todas las posibles opciones, hemos dividido este camino en 3:

**CP31:** La función entra en if (first != null), no entra en while (cur != null && !found) ya que cur == null y found == true, te da igual si entra o no en el último if (!found).

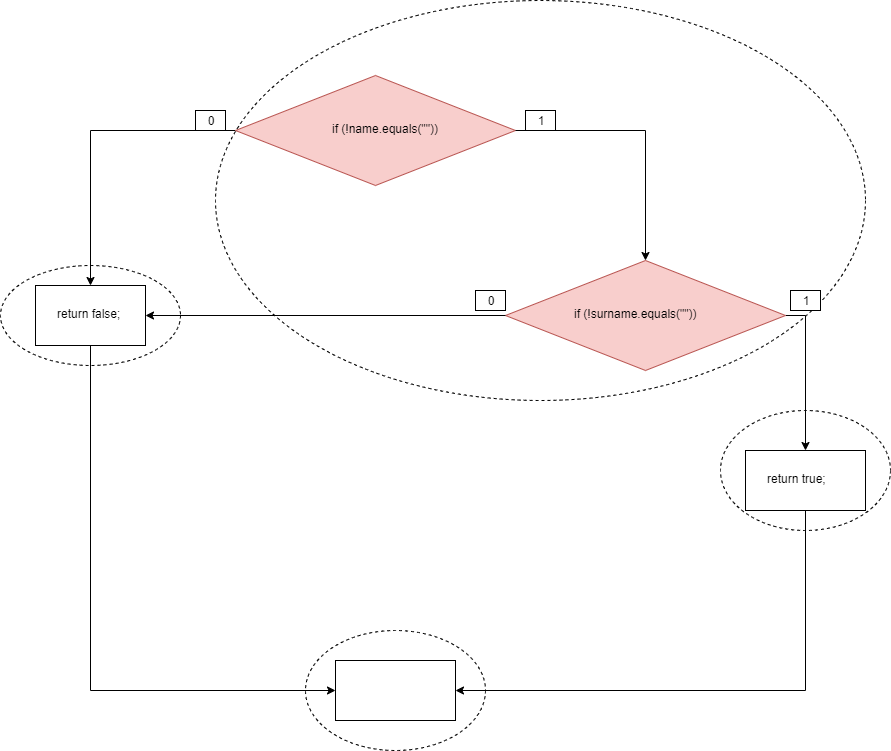
**CP32:** La función entra en if (first != null), no entra en while (cur != null && !found) ya que cur =! null y found == true, te da igual si entra o no en el último if (!found).

**CP33:** La función entra en if (first != null), no entra en while (cur != null && !found) ya que cur =! null y found == false, te da igual si entra o no en el último if (!found).

**Camino 4:** La función entra en if (first != null), entra en while (cur != null && !found), no entra en if(cur.into.getName() == p.getName()), te da igual si entra o no en el último if (!found).

**Camino 5:** La función entra en if (first != null), entra en while (cur != null && !found), entra en if(cur.into.getName() == p.getName()), te da igual si entra o no en el último if (!found).

Clase AgendaEntry

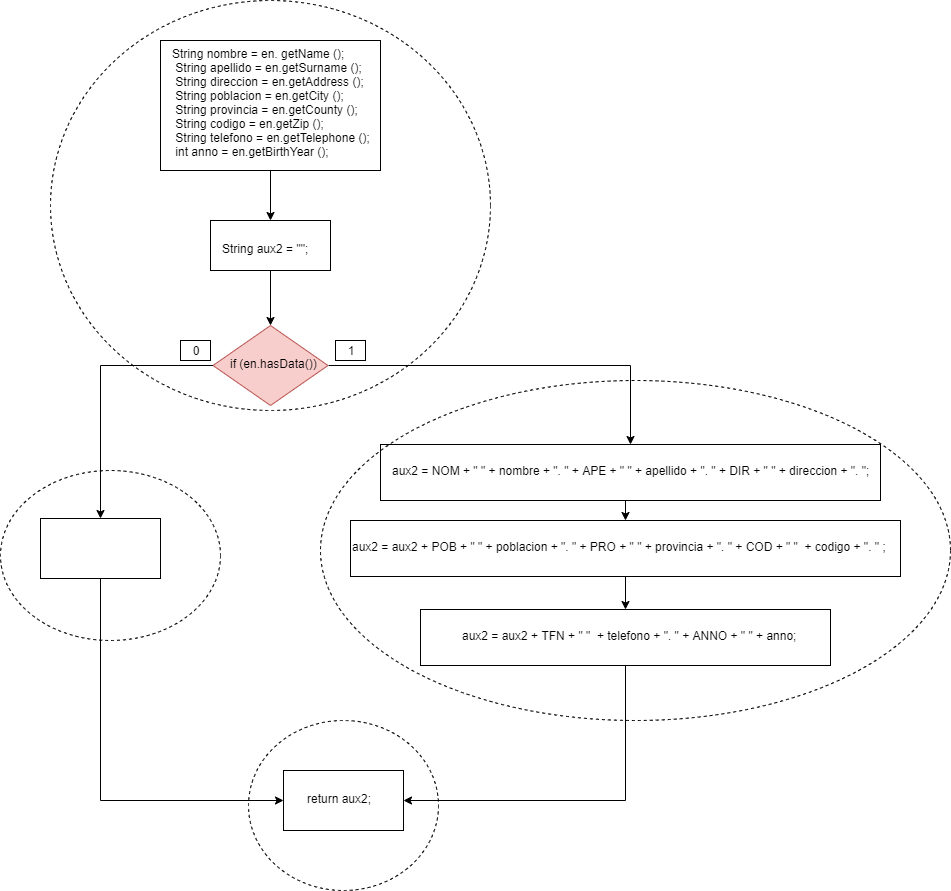


Complejidad: Rombos + 1 = Áreas = 2

Caminos mínimos:

1. La función no entra en el if.
2. La función entra en el if.

Clase AgendaParser



Complejidad: Rombos + 1 = Áreas = 2

Camino mínimos:

1. La función no entra en el if.
2. La función entra en el if.