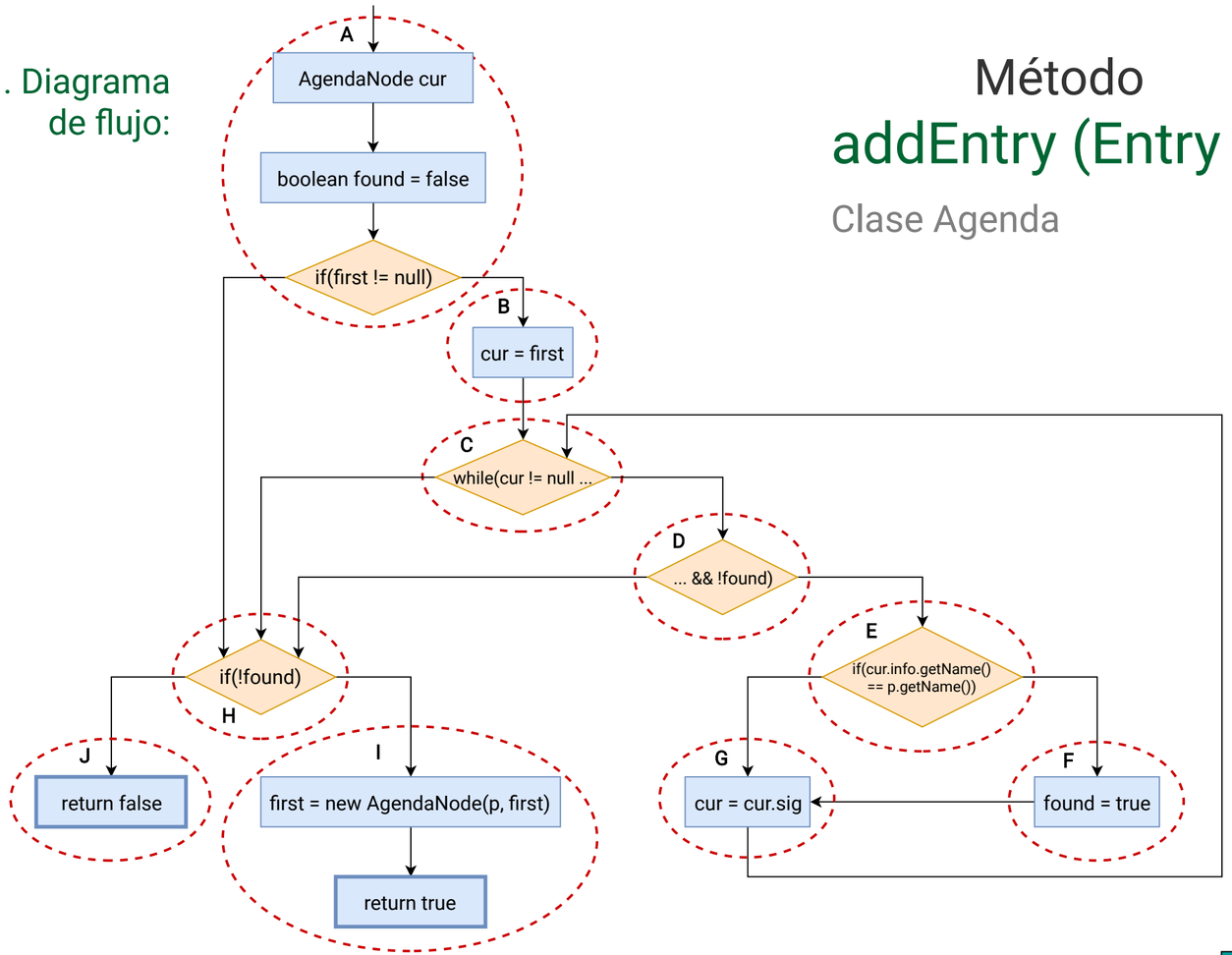
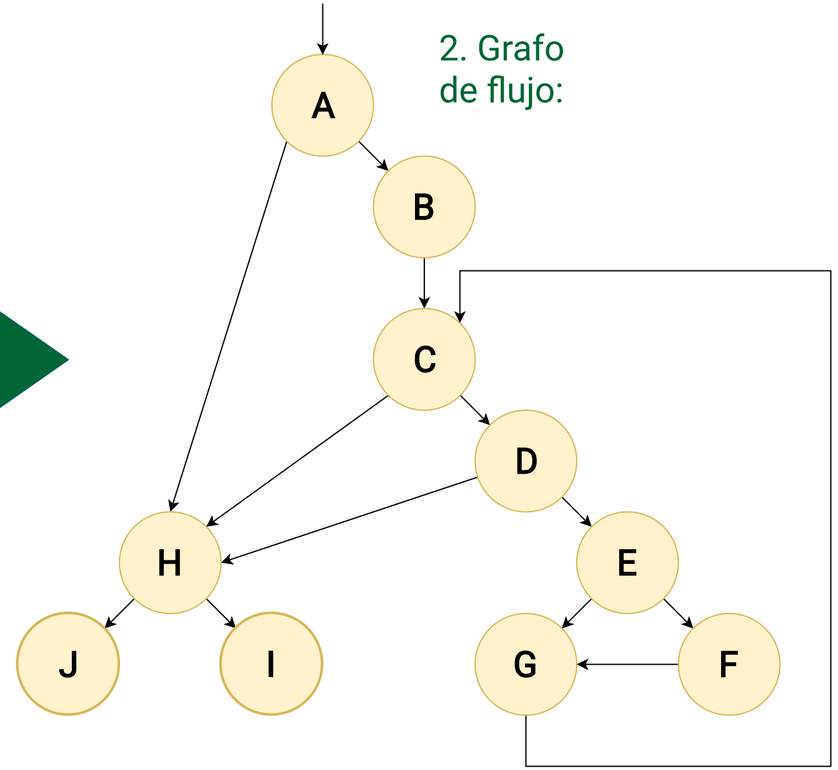
Método
addEntry (Entry p)

Clase Agenda

1. Diagrama de flujo:



2. Grafo de flujo:



3. Complejidad ciclomática:

Número de rombos + 1 =
5 + 1 = 6

4. Caminos independientes:

#1: A - H - J
#2: A - H - I
#3: A - B - C - H - I
#4: A - B - C - D - H - J
#5: A - B - C - D - E - G - C - H - I
#6: A - B - C - D - E - F - G - C - H - J

Existen otros caminos independientes equivalentes

5. Casos de prueba:

Camino	(Entry) p	(AgendaNode) first	Salida	Factible
#1	["David"]	null	false	NO
#2	["Alex"]	null	true	SÍ
#3	["Alex"]	["David"] → null	true	NO
#4	["David"]	["Alex"] → null	false	NO
#5	["Alex"]	["David"] → null	true	SÍ
#6	["David"]	["David"] → null	false	SÍ

Para probar aquellos casos de prueba que no son factibles se ha modificado el código del método para forzar el flujo del camino asociado

6. jUnit:

Para los caminos factibles (#2, #5 y #6) se ha realizado la prueba en jUnit en base al método original, addEntry.

Para los no factibles (#1, #3 y #4) se han creado nuevos métodos copiados del original, añadiendo la sentencia necesaria para forzar la condición del if o while y así seguir dicho camino:

- **addEntryV2** para el camino #1
- **addEntryV3** para el camino #3
- **addEntryV4** para el camino #4

En ambos tipos de caminos se ha llevado a cabo la prueba con los datos de entrada mostrados en la tabla anterior, comprobando la igualdad entre el camino en concreto y el contenido de la variable traza.

Esta variable, de tipo string, es creada para mostrar los nodos del grafo de flujo por donde pasa el flujo del programa, modificando el método para ello.

También se comprueba la igualdad en la salida real de la función y la esperada (véase tabla de casos prueba).