Agenda anual

Dispones de la implementación de un sistema para gestionar una agenda correspondiente al año 2024. La agenda se compone de semanas en las cuales se pueden registrar citas los días laborables. Las citas tienen una duración (en horas) y una descripción. La clase que se utiliza para diseñar las citas está correctamente implementada. Los desarrolladores de las clases asociadas a los días y las semanas las han implementado pero no las han testeado. Está es tu labor: diseñar un conjunto de tests que permita chequear ambas implementaciones.

Los requisitos de nuestra implementación para la clase Dia son los siguientes:

- Los días se identifican por el número del día del año (1-366).
- Cada día almacena las citas correspondientes a cada hora disponible para tal efecto: primera cita a las 9:00 y última cita a las 17:00 de la tarde.
- Una cita puede tener una duración superior a una hora, por lo que puede estar asignada a varias horas consecutivas de un día.
- La clase presenta varios métodos:

1. `int buscaSlot(int duracion)`: devuelve la primera hora disponible para una cita de la duración que recibe como parámetro. En caso de no encontrar un hueco, devolverá -1.
2. `boolean asignarCita(int hora, Cita cita)`: asigna la cita a las horas necesarias para la misma empezando en la hora indicada si hay hueco libre. Devuelve un boolean que indica si la operación se ha podido realizar.
3. `Cita getCita(int hora)`: devuelve la Cita asignada a la hora que se recibe como parámetro. En caso de estar libre esa hora, devuelve null.
4. `String muestraCita(int hora)`: devuelve un String que describe la Cita asociada a la hora indicada como parámetro

    a. “No existe” si no hay Cita asignada a la hora recibida como parámetro.
    b. “Hora no valida” si la hora indicada no es asignable a una cita.
    c. “hora: 00 Descripción de la Cita”.
5. `boolean validaHora(int hora)`: indica si la hora recibida como parámetro es asignable para una cita o no.

6. `public boolean huecoLibre(int hora, int duracion)` indica si esta libre el hueco que abarca desde la hora indicada tantas horas como duración tiene la cita.

Los requisitos de nuestra implementación para la clase Semana son los siguientes:

- Una semana se identifica por el número de semana del año (1-52).
- Cada semana almacena los días disponibles para asignación de citas (lunes-viernes) esa semana.
- La clase presenta varios métodos:

1. `String mostrarCita(int hora, int diaSemana)`: devuelve un String que describe la Cita asociada al día de la semana y la hora indicadas como parámetros

    a. "No existe" si no hay Cita asignada a la hora recibida como parámetro.
    b. "Hora no valida" si la hora indicada no es válida.
    c. "hora:00 Descripción de la Cita".

2. `Dia getDia(int diaSemana)`: devuelve el Dia correspondiente al día de la semana que se recibe como parámetro. En caso de recibir 6 o 7 (sábado o domingo), devuelve null.

3. `public int getNumeroSemana()`: devuelve el número de semana.

4. `public String primerHueco(int duracion)`. Devuelve un String que indica el primer día de la semana con disponibilidad y la hora de inicio de la cita. En otro caso devuelve "No hay disponibilidad".

5. `public String diaSemana(int dia)`. Devuelve el nombre del día de la semana (Lunes, Martes…Viernes). En otro caso muestra "No citable".

6. `boolean validaDia(int diaSemana)`: indica si el día recibido como parámetro es válido para citar.

Para diseñar los tests deberás considerar diferentes criterios de cobertura para los métodos relacionados a continuación.

```
D1) public Dia(int diaNumero)

D2) public int buscaSlot(int duracion)

D3) public boolean asignarCita(int hora, Cita cita)

D4) public Cita getCita(int hora)

D5) public String muestraCita(int hora)

D6) public boolean validaHora(int hora)

D7) public boolean huecoLibre(int hora, int duracion)
```

```
S1) public Semana(int numeroSemana)

S2) public String mostrarCita(int hora, int diaSemana)

S3) public Dia getDia(int diaSemana)

S4) public String primerHueco(int duracion)

S5) public String diaSemana(int dia)
```

a) Cobertura Correlated Active Clause Coverage CACC para los métodos D1, D2, D3, D5, D7, S1, S4.
b) Cobertura Prime Path Coverage (PPC) para los métodos D2 y S5.
c) Cobertura Base Choice Coverage (BCC) para los métodos D4, D5, D6 (en este caso solo se necesita una característica), S2, S3.

Debes aplicar los casos de tests a las clases mutadas en el proyecto AgendaBreak. Cada error detectado deberá ser corregido, hasta conseguir que se pasen todos los tests diseñados.

Se deberán recoger en un documento los errores detectados (método e instrucción), así como el/los tests que han fallado, el tipo de cobertura que ha dado lugar a dicho test y la corrección aplicada.

Entrega:

1) Clase de Tests correspondientes a las coberturas indicadas previamente.
2) Documento con el desarrollo del diseño de los casos de tests de cada criterio.
3) Fichero ***errores.pdf*** con la información recogida en el proceso de Mutation Testing aplicado a las clases del proyecto AgendaBreak.