Cuaderno de ejercicios de la UP05

Realiza un plan de pruebas de los siguientes enunciados que trabajaste en la unidad de programación 3

Sigue estas instrucciones

- Diseña un plan de pruebas coherente, teniendo en cuenta
    - Pruebas unitarias: diseña escenarios en los que usarías pruebas tipo JUnit (caja negra, basadas en casos de uso). Es necesario representarlas en forma de tabla y solo es necesario hacer 4, pero que sean relevantes.
    - Pruebas de integración: diseña escenarios en los que usarías pruebas tipo Mockito y GitHub Actions (simulación de componentes y tareas automatizadas en cada cambio).
    - Pruebas de aceptación: diseña escenarios usando pruebas tipo Cucumber, utilizando lenguaje Gherkin para definir los criterios del cliente.
    - Pruebas de seguridad y otros tipos: diseña escenarios donde se ponga a prueba la robustez y protección del sistema ante fallos o ataques.

Ejercicio 1: Gestor de Liga de Fútbol

Se ha encargado desarrollar un gestor para una liga de fútbol. La liga debe estar identificada por un nombre y una temporada, definida por el año de inicio y el de finalización. La liga estará compuesta por un máximo de 22 equipos.

Cada equipo debe incluir información como su nombre, el número de partidos ganados, empatados y perdidos. A partir de estos datos, se podrá calcular la puntuación total de cada equipo según el sistema estándar de puntos (3 puntos por victoria, 1 por empate, 0 por derrota). Además, cada equipo debe tener entre 18 y 24 futbolistas.

Cada futbolista debe tener datos que lo identifiquen: nombre, nacionalidad, un número de identificación único, su posición en el campo (portero, defensa, centrocampista o delantero), el número de goles marcados y el número de partidos jugados.

Se requiere que solo exista una instancia de la liga (patrón singleton). Debes implementar la funcionalidad para añadir y eliminar equipos, así como para añadir y eliminar futbolistas de los

equipos. Además, se debe poder acceder a la información completa de cada jugador, equipo o la liga misma.

El sistema debe proporcionar las siguientes funcionalidades:

- Mostrar los equipos en posiciones de descenso (los 4 últimos).
- Mostrar los equipos en posiciones de clasificación para competiciones europeas (los 4 primeros).
- Calcular los goles a favor de cada equipo.
- Identificar al máximo goleador ("pichichi") de la liga.

Ejercicio 2: Gestor de Aeropuertos

Se te ha pedido desarrollar un sistema para gestionar la operativa de un aeropuerto internacional. El aeropuerto debe estar identificado por un nombre, un código IATA de tres letras, y la ciudad donde se encuentra.

Cada aeropuerto gestiona varios vuelos. Un vuelo está identificado por un número único, la aerolínea operadora, el destino y el origen. También debe incluir la hora de salida, la hora de llegada estimada y el estado del vuelo (en hora, retrasado, cancelado).

Cada vuelo tiene una tripulación asignada, formada por al menos un piloto y dos auxiliares de vuelo. Los tripulantes deben incluir información como su nombre, nacionalidad, y su número de identificación único.

Se debe poder:

- Añadir y eliminar vuelos.
- Asignar o eliminar tripulantes de los vuelos.
- Consultar el estado de cualquier vuelo en cualquier momento.
- Filtrar los vuelos por su destino, origen o estado.
- Calcular el tiempo estimado de llegada para los vuelos en base a su hora de salida.

El sistema debe permitir acceder a la información completa de cada vuelo y cada tripulante.

Tabla para las pruebas de unidad:

ID	Nombre de prueba	método	descripción (precondiciones y pasos)	entrada	salida esperada	salida obtenida
1						
2						
3						
4						

Ejercicio de Debug

Dado el siguiente código:

```java
public class EjercicioDebug {
    public static void main(String[] args) {
        int[] n = {8, 1, 6, 2, 3};
        int r = 0;
        for (int i = 0; i < n.length; i++) {
            r += procesaNumero(n[i]); // Colocar breakpoint con condición: n[i] > 5
        }
        System.out.println("Resultado: " + r);
        muestraInfoExtra(n);
    }
    static int procesaNumero(int x) {
        int y = 0;
        if (x % 3 == 0) {
            y = (x * 2)/3;
        } else {
            y = x * 3;
        }
        System.out.println("Procesando: " + x);
        System.out.println("Temporal: " + y);
        System.out.println("-----");
        return y;
    }

    static void muestraInfoExtra(int[] arr) {
        int s = 0;
        for (int i = 0; i < arr.length; i++) {
            s += arr[i];
        }
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] % 3 == 0) {
                System.out.println(arr[i] + " es divisible entre 3");
            } else {
                System.out.println(arr[i] + " no es divisible entre 3");
            }
        }
        System.out.println("Suma total: " + s);
    }
}
```

Coloca un punto de ruptura en la línea indicada con la condición establecida y realiza las siguientes pruebas:

- En el primer punto de ruptura, ejecuta un step over y continúa hasta la siguiente ruptura.
- La segunda vez que pare, ejecuta un step into y ejecuta step over 3 veces. Después ejecuta step out.

  Refleja el resultado en una traza con tabla de seguimiento de variables y de pasos (como la pila de llamadas del debugger).