EL CLUB DE LOS POETAS MUERTOS

"No dejen que les digan que no valen nada, porque los sueños son para los poetas. Solo ustedes pueden cambiar su destino."

Tenemos los siguientes estudiantes:

REYES	JUANJO	ZULEMA	PEDRO	CAROLINA
ALEJANDRO	LEANDRO	CHRISTIAN	PABLO	ROBERTO
ANYORLINA	DIEGO	HECTOR	JORGE	LUCAS
RAUL	JORDAN	EMMA	IVAN	ALVARO
JAVI	DAVID	ERICK	ADRI	INMA

Para cada persona consideramos las siguientes características:

- Apodo[1]: no nulo, no vacío, no blanco, longitud >=3, no modificable (mutador privado)

Tenemos los siguientes poetas:

NUMERO	NOMBRE	APE1	PAIS
1	PABLO	NERUDA	CHILE
2	FEDERICO	GARCIA	ESPAÑA
3	OCTAVIO	PAZ	MEXICO
4	ALFONSINA	STORNI	ARGENTINA
5	MARIO	BENEDETTI	URUGUAY
6	GABRIELA	MISTRAL	CHILE
7	GUSTAVO ADOLFO	DOMINGUEZ	ESPAÑA
8	CESAR	VALLEJO	PERU
9	JUAN RAMON	JIMENEZ	ESPAÑA
10	ANTONIO	MACHADO	ESPAÑA

[1] Los apodos que aparecen en el enunciado son ficticios y cualquier parecido con la realidad es pura coincidencia ☺

→ Se deben usar colecciones cuando proceda ←

Crear las siguientes clases:

1. Principal
2. Estudiante: tendrá como atributo el apodo
3. Poeta: con los atributos necesarios
4. Generador: clase "utility" con dos métodos estáticos
    a. generarEstudiante(): devuelve un objeto de la clase Estudiante. Los apodos deben estar almacenados en una lista (ArrayList o LinkedList). Cada vez que se invoque al método debe devolver un estudiante empezando por REYES y terminando por INMA. **Sólo debe devolver 1 estudiante cada vez**. Si se invoca más veces del número de estudiantes existentes, debe devolver la cadena "NO SOY UN ESTUDIANTE".
    b. generarPoetas(): devuelve una lista de objetos de la clase Poeta. Los poetas estarán almacenados en una colección (ArrayList o LinkedList)

SE PIDE (2EV)

1) [4p] Definir:
    a) [0,5p] Clase **Estudiante**
        I. [0,25p] Dos estudiantes son iguales si tienen el mismo apodo
        II. [0,25p] El método **toString** en la clase Estudiante devuelve el apodo
    b) [1p] Clase **Poeta**
        I. [0,25p] Número de poeta: autoincrementado a partir de 1
        II. [0,5p] Dos poetas son iguales si tienen el mismo nombre, ape1 y país.
        III. [0,25p] El método **toString** en la clase poeta devuelve: [*numero*] *nombre*, *apellido* (*pais*)
    c) [2,5p] Clase **Generador**
        I. [1,5p] **generarEstudiante**()
        II. [0,5p] **generarPoetas**()
        III. [0,25p] Justifica por qué has elegido la colección ArrayList o LinkedList para los estudiantes. ¿Has necesitado usar alguna colección auxiliar? ¿Para qué?
        IV. [0,25p] Idem para los poetas
2) [4,5p] Realizar las siguientes acciones en la clase **Principal**:
    a) [1p] Usando una estructura de control iterativa crear una lista en la que estén todos los estudiantes. Se debe invocar al método **generarEstudiante** de la clase **Generador**.
    b) [0,25p] Mostrar la lista de estudiantes separados por coma por pantalla.
    c) [1p] Crear un conjunto de estudiantes llamado **estu10** que tenga 10 estudiantes elegidos aleatoriamente de la lista de estudiantes creada en el apartado a. No puede haber estudiantes repetidos
    d) [0,5p] Mostrar el conjunto de estudiantes **estu10** por pantalla separados por un salto de línea cada uno. Se debe usar un iterador

e) [0,25p] Crear una lista llamada **poetas10** invocando al método **generarPoetas** de la clase **Generador**

f) [1p] Crear un mapa que tenga 10 pares (estudiante, poeta). El mapa se llamará **mapa10** y tendrá como claves los estudiantes del conjunto **estu10** y como valores los poetas de la lista **poetas10**.

g) [0,5p] Usando un iterador mostrar el **mapa10** por pantalla con los pares separados por salto de línea

3) [1,5p] Estructura general del programa (validación de parámetros, legibilidad, escalabilidad, rendimiento, uso de tipos de datos)

SE PIDE (1REC)

1) [8p] Clase Poeta
   a) [2p] Atributos
   b) [0,5p] Restricciones: definir restricciones adecuadas para los atributos y especificarlas en los comentarios
   c) [1p] Constructor
   d) [0,5p] Accesores
   e) [3p] Mutadores y validación de parámetros. Control de restricciones.
   f) [1p] Método toString()
2) [2p] Estructura general del programa (legibilidad, control de errores, abstracción, encapsulación, ocultación, …) .

REQUISITOS DE OBLIGADO CUMPLIMIENTO

- Copiar en el examen tiene una calificación de 0 y supondrá la adopción de medidas disciplinarias con el alumnado implicado
- El profesor indicará al alumnado con qué ordenador realizará el examen y el sitio que ocupará en el aula para realizarlo
- Entregar la carpeta COMPLETA del proyecto IntelliJ de todas las versiones solicitadas por el profesor durante el examen
- La entrega se debe realizar en el tiempo y forma indicados por el profesor
- No disponer ni usar conexión a internet ni de red local durante el examen
- Para aquellos exámenes sospechosamente parecidos entre ellos habrá un examen oral para demostrar la autoría del mismo
- Realizar copia de la carpeta del proyecto cuando el profesor lo indique
- Respetar los conceptos del paradigma orientado a objetos (legibilidad, ocultación de datos, reusabilidad, robustez, ...)