**PRIMER PARCIAL**

**INF310 SX**‒ **Estructuras de Datos II. Gestión 1-2019.**

**Subgrupo: A-F**

Árbol Binario

1. En la class Arbol (desordenado pero sin duplicados), escriba el procedimiento

**public void delHojas(int nivel)** //Asuma que nivel > 0

el cual elimine del árbol las hojas que se encuentran en el **nivel** dado. Si no existen hojas en ese nivel, este procedimiento no realiza ninguna acción.

**Por ejemplo**: (En el gráfico, no se dibujan los punteros null)

**Árbol A**

**Nivel 1**

**Nivel 2**

**Nivel 3**

**Nivel 4**

|  |  |
| --- | --- |
| **A.delHojas(5);** | El nivel 5 no existe: “No pasa nada” (el árbol queda igual) |
| **A.delHojas(2);** | No hay hojas en el nivel 2: “No pasa nada” (el árbol queda igual) |
| **A.delHojas(3);**  El 50 es la única hoja del nivel 3. Entonces, ésta será eliminada. | **Árbol A**  **Nivel 1**  **Nivel 2**  **Nivel 3**  **Nivel 4** |
| **A.delHojas(4);**  Las hojas del nivel 4 son eliminadas. | **Árbol A**  **Nivel 1**  **Nivel 2**  **Nivel 3** |

**Listas**

2. En un videojuego se desea llevar la cuenta de cuántas “vidas” le quedan a un personaje (los personajes se identifican con un número entero único). Usando Single List’s (puede usar más de una), implemente esta situación a través de una class Lista, cuyas operaciones son:

**public Lista()**

//**Constructor**. La lista de personajes está vacía.

**public void add(int nroPersonaje, int cantVidas)** //Asuma que cantVidas > 0

//Si nroPersonaje **NO** existe en la Lista, se lo inserta con la **cantVidas** especificado.

//Si nroPersonaje ya está en la Lista, su cantidad de vidas actual **se cambia** por **cantVidas**.

**public void shoot(int nroPersonaje)**

//Si nroPersonaje **NO** existe en la Lista, este procedimiento **no hace nada**.

//Si nroPersonaje ya está en la Lista, su cantidad de vidas se decrementa en uno. Si la cantidad de vidas queda en cero,

// este personaje es eliminado de la Lista.

**public void mostrar()**

//Muestra en consola (System.out.println) los personajes con sus respectivas vidas restantes. Se sugiere que se muestre

// una tira de elementos NroPersonaje/cantVidas.

**Por ejemplo (en el main):** Como usuario, imaginamos a la Lista P como una secuencia de pares NroPersonaje/cantVidas.

Lista P = new Lista(); //P=(vacía)

P.add(20, 5); //P=[**20/5**] (El personaje 20 se inserta con 5 vidas)

P.add(10, 2); //P=[20/5, **10/2**] (El personaje 10 se inserta con 2 vidas)

P.add(80, 6); //P=[20/5, 10/2, **80/6**] (El personaje 80 se inserta con 6 vidas)

P.mostrar(); //(En la consola se muestra): **20/5, 10/2, 80/6**

//***No es necesario que su procedimiento* mostrar() *separe los elementos con comas, lo importante es que se vean.***

P.shoot(10); //P=[20/5, **10/1**, 80/6] (El personaje 10 recibió un disparo, por tanto se le quita una vida)

P.add(20, **3**); //P=[**20/3**, 10/1, 80/6] (El personaje 20 ya existe, por tanto solo se le modifica su cantidad de vidas)

P.shoot(50); //P=[20/3, 10/1, 80/6] (Se le disparó al personaje 50. Como el 50 **NO existe**, “no pasa nada”)

P.shoot(10); //P=[20/3, 80/6] (El 10 recibió un disparo, por tanto se le quita una vida. Como su cantidad de vidas es cero, el 10

// es eliminado de la Lista P)

P.mostrar(); //(En la consola se muestra): **20/3, 80/6**