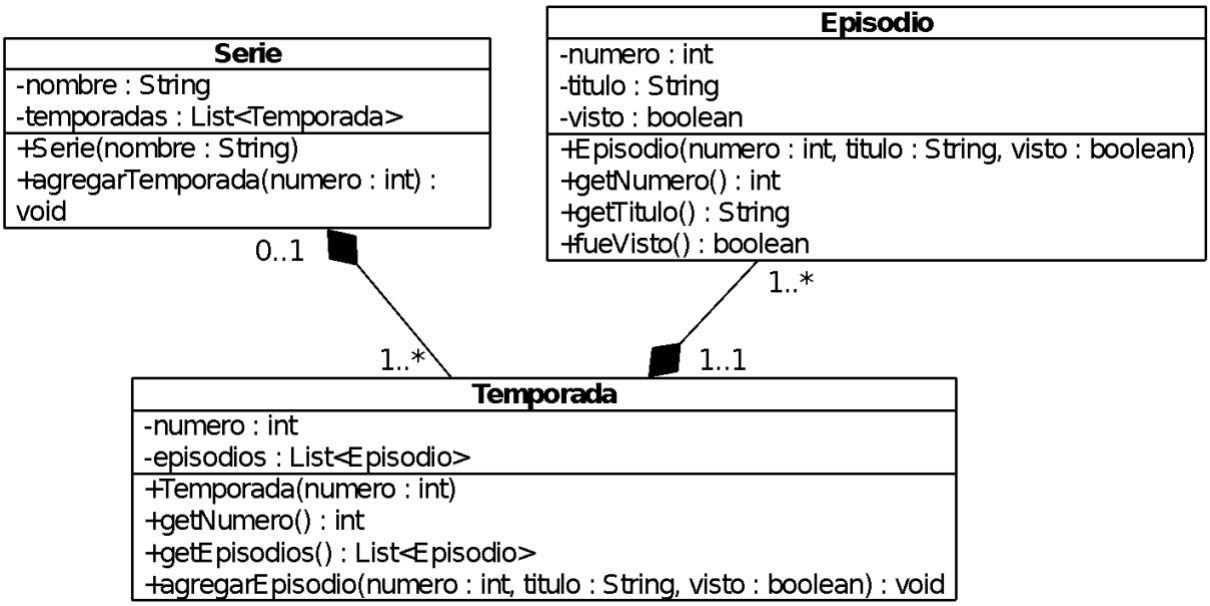
Examen 111Mil – Series de TV

Dada la cantidad de series que cada persona mira habitualmente, es muy difícil llevar registro de qué episodios fueron vistos y cuáles no. Como Montserrat cursó el 111Mil, se le ocurrió hacer una aplicación que ayude con este problema. La aplicación permitirá llevar un registro para cada temporada de una determinada serie de qué episodios fueron vistos y cuáles no el usuario de la aplicación.
Como sabe que cursamos el programa 111Mil nos invitó a ser parte del desarrollo.

Ejercicio 1. Implementar desde el diagrama de clases

Montserrat realizo el diagrama UML de la aplicación, implemento las clases *Temporada* y *Episodio*, y nos pide que implementemos la clase *Serie*. Para ello tenga en cuenta que:

a) La lista de temporadas debe ser inicializada vacía dentro del constructor.
b) El método *agregarTemporada* debe crear y agregar una nueva temporada utilizando el número de temporada recibido por parámetro. Puede suponer que no se agregarán temporadas con el mismo número. Es decir, no es necesario controlar esa situación.



Ejercicio 2. Implementar un método a partir de un enunciado

Montserrat nos pidio que programemos una funcionalidad de obtener un episodio dado el número de temporada en el cual fue emitido y el número de episodio dentro de esa temporada. Implemente la funcionalidad indicando a qué clase corresponde el/los método/s que satisface/n dicha funcionalidad.

Ejercicio 3. Seguimiento de código

Considerando el siguiente método:

```
    public int yyy(Temporada t){
        int z=0;
        for (Iterator<Episodio> iterator= t.getEpisodios().iterator();
iterator.hasNext();) {
            Episodio next = iterator.next();
            if(next.fueVisto())
                z++;
            else
                return z;
        }
        return z;
    }
```

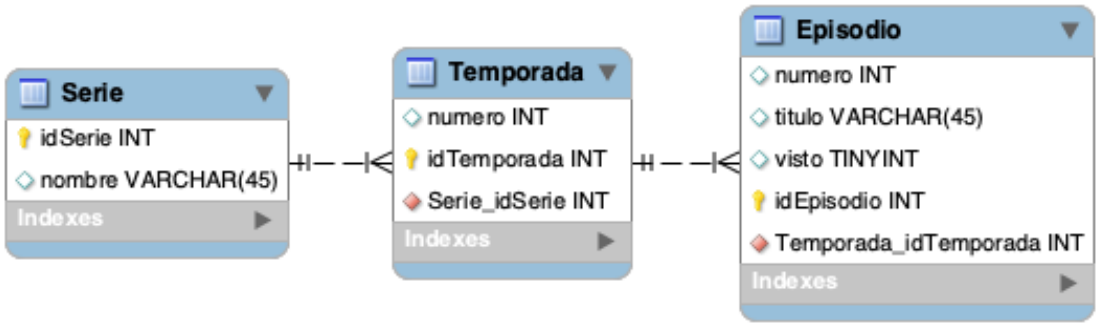
¿Qué imprimirá el programa al ejecutar el siguiente código? Tenga en cuenta que Montserrat nos indicó que el método *agregarEpisodio* de la clase *Temporada* crea y agrega en la lista *episodios* un nuevo episodio.

```
Temporada t1=new Temporada(1);
t1.agregarEpisodio(1, "t1e1",true);
t1.agregarEpisodio(2, "t1e2",true);
t1.agregarEpisodio(3, "t1e3",false);

Temporada t2=new Temporada(2);
t2.agregarEpisodio(1, "t2e1",true);
t2.agregarEpisodio(2, "t2e2",false);
t2.agregarEpisodio(3, "t2e3",true);

System.out.println(yyy(t1));
System.out.println(yyy(t2));
```

Ejercicio 4. Javadoc

Elaborar la documentación técnica utilizando Javadoc del método *agregarTemporada* implementado en el ejercicio 1. Incluya tanto la descripción del método como los tags que correspondan.

Ejercicio 5. Consultas SQL



Dado el diagrama de entidad-relación, complete la siguiente consulta para que liste solamente el número total de temporadas de la serie “111Mil”

SELECT COUNT(*) FROM Temporada

Ejercicio 6. Hibernate

Dado el diagrama de entidad-relación presentado en el ejercicio anterior y el diagrama UML presentado en el ejercicio 1, escriba la línea del archivo de mapeo de Hibernate correspondiente al mapeo del atributo “titulo”.

Soluciones

Ejercicio 1. Implementar desde el diagrama de clases

```
public class Serie {
   private String nombre;
   private List<Temporada> temporadas;

   public Serie(String nombre) {
      this.nombre = nombre;
      this.temporadas = new ArrayList<>();
   }

   public void agregarTemporada(int numero){
      temporadas.add(new Temporada(numero));
   }

}
```

Ejercicio 2. Implementar un método a partir de un enunciado

En la clase Serie:

```
  public Episodio obtenerEpisodio(int nroTemporada, int nroEpisodio){
      Iterator<Temporada> itTemporadas= temporadas.iterator();
      while (itTemporadas.hasNext()) {
         Temporada temporada = itTemporadas.next();
         if(temporada.getNumero()==nroTemporada){
            return temporada.obtenerEpisodioPorNro(nroEpisodio);
         }
      }
      return null;
   }
```

En la clase Temporada:

```
  public Episodio obtenerEpisodioPorNro(int nroEpisodio) {
      Iterator<Episodio> itEpisodios = episodios.iterator();
      while (itEpisodios.hasNext()) {
         Episodio episodio = itEpisodios.next();
         if (episodio.getNumero() == nroEpisodio) {
            return episodio;
         }
      }
      return null;
   }
```

Ejercicio 3. Seguimiento de código

Imprime 2 y 1

Ejercicio 4. Javadoc

```
/**
* crea y agrega una nueva temporada utilizando el número de temporada recibido por parámetro
```

*
*@param numero número de la tempora a agregar
*/

Ejercicio 5. Consultas SQL

SELECT COUNT(*) FROM Temporada t INNER JOIN Serie s ON (t.Serie_idSerie=s.idSerie) where s.nombre='111Mil'

Ejercicio 6. Hibernate

<property name=”titulo” type=”java.lang.String”/>