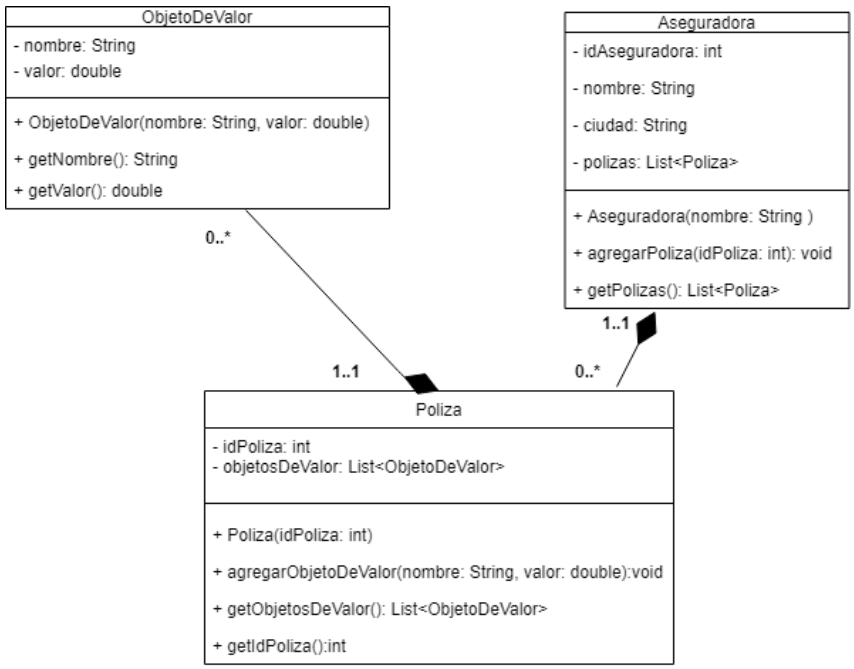
Examen 111Mil – Compañía de Seguros

Esteban ha empezado a asegurar varios objetos que tiene en su casa para estar tranquilo ante cualquier siniestro, como por ejemplo robo, incendio, inundación o tormenta eléctrica. Como Esteban cursó el 111Mil, se le ocurrió hacer una aplicación que lo ayude a gestionar sus objetos asegurados y las pólizas asociadas para buscar facilmente la cobertura qué cada compañía de seguros le hace.
Como sabe que cursamos el programa 111Mil nos invitó a ser parte del desarrollo.

Ejercicio 1. Implementar desde el diagrama de clases

Esteban realizo el diagrama UML de la aplicación, implemento las clases *Aseguradora y ObjetoDeValor*, y nos pide que implementemos la clase *Póliza* (Una póliza es un contrato que indica los objetos asegurados por un cliente). Para ello tenga en cuenta que:

a) La lista de objetos de valor debe ser inicializada vacía dentro del constructor.
b) El método *agregarObjetodeValor* debe crear y agregar un nuevo objeto de valor utilizando el nombre y el valor pasados por parámetro.



Ejercicio 2. Implementar un método a partir de un enunciado

Esteban nos pidió que programemos una funcionalidad que obtenga el valor de un “objeto de valor” dado el número de poliza y el nombre del objeto de valor. 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 zzz(Aseguradora a){
        int vvv=0;
        for (Iterator<Poliza> iterator= a.getPolizas().iterator();
iterator.hasNext();) {
            Poliza poliza= iterator.next();
            if(poliza.getIdPoliza()%2==0)
                vvv++;
            }
        return vvv;
    }
```

¿Qué imprimirá el programa al ejecutar el siguiente código? Tenga en cuenta que Esteban nos indicó que el método *agregarPoliza* de la clase *Aseguradora* crea y agrega en la lista *polizas* una nueva póliza.

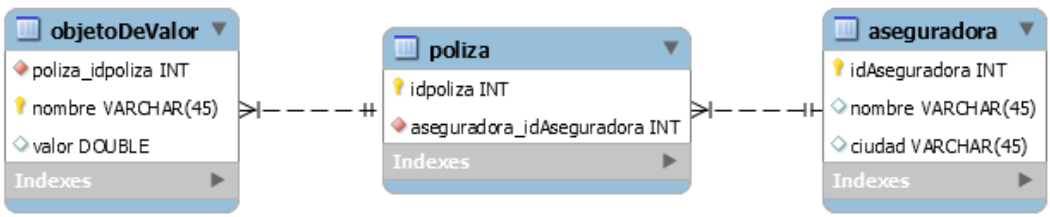
```
        Aseguradora a1=new Aseguradora("Sancor Seguros");
        a1.agregarPoliza (1);
        a1.agregarPoliza(2);
        a1.agregarPoliza(3);

        Aseguradora a2=new Aseguradora("Seguros Rivadavia");

        a2.agregarPoliza(1);
        a2.agregarPoliza(2);
        a2.agregarPoliza(4);

        System.out.println(zzz(a1));
        System.out.println(zzz(a2));
```

Ejercicio 4. Javadoc

Elaborar la documentación técnica utilizando Javadoc del método *agregarObjetodeValor* 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 calcule la suma total de lo valores de los objetos asegurados en la póliza '9938'.



SELECT SUM(o.valor) FROM poliza p INNER JOIN objetodeValor o ON p.idpoliza=o.poliza_idpoliza;

Ejercicio 6. Hibernate

Dado el diagrama de entidad-relación presentado en el ejercicio anterior y el diagrama UML presentado en el ejercicio 1, muestre cómo sería el mapeo del atributo polizas de la clase Aseguradora usando Hibernate.

Solución

Ejercicio 1. Implementar desde el diagrama de clases

```
public class Poliza{
   private int idPoliza;
   private List<ObjetoDeValor> objetosDeValor;

   public Poliza(int identificador) {
      this.idPoliza = identificador;
      this.objetosDeValor = new ArrayList<>();
   }

   public void agregarObjetoDeValor(String nombre, double valor){
      objetosDeValor.add(new ObjetoDeValor(nombre, valor));
   }

   public List<ObjetoDeValor> getObjetosDeValor (){
      return this.objetosDeValor;
   }

   public int getIdPoliza (){
     return this.idPoliza;
   }
}
```

Ejercicio 2. Implementar un método a partir de un enunciado

En la clase Aseguradora:

```
public double obtenerCobertura(int idPoliza, String nombre){
      for (Iterator<Poliza> iterator= this.polizas.iterator(); iterator.hasNext(); ) {
        Poliza poliza = iterator.next();
         if(poliza.getIdPoliza()==idPoliza)
           return poliza.obtenerValorCobertura(nombre);
}
 return 0;
}
```

En la clase Poliza:

```
public double obtenerValorCobertura(String nombre)
{
       for (Iterator<ObjetoDeValor> iterator=this.objetosDeValor.iterator(); iterator.hasNext();) {
              ObjetoDeValor objeto=iterator.next();
              if(objeto.getNombre().equals(nombre))
                 return objeto.getValor();
      }
      return 0;
   }
```

Ejercicio 3. Seguimiento de código

Imprime 1 y 2

Ejercicio 4. Javadoc

```
/**
```

```
* crea y agrega un nuevo objeto de valor utilizando el nombre del objeto y su valor recibidos por parámetro
*
*@param nombre nombre del objeto de valor a agregar
*@param valor valor del objeto de valor a agregar
*/
```

Ejercicio 5. Consultas SQL

```
SELECT SUM(o.valor) FROM poliza p INNER JOIN objetodeValor o ON p.idpoliza=o.poliza_idpoliza
WHERE p.idpoliza=9938 GROUP BY p.idpoliza;
```

Ejercicio 6. Hibernate

XML

```
<set name="polizas" table="poliza">
	<key>
		<column name="idAseguradora" not-null="true" />
	</key>
	<one-to-many class="Poliza" />
</set>
```