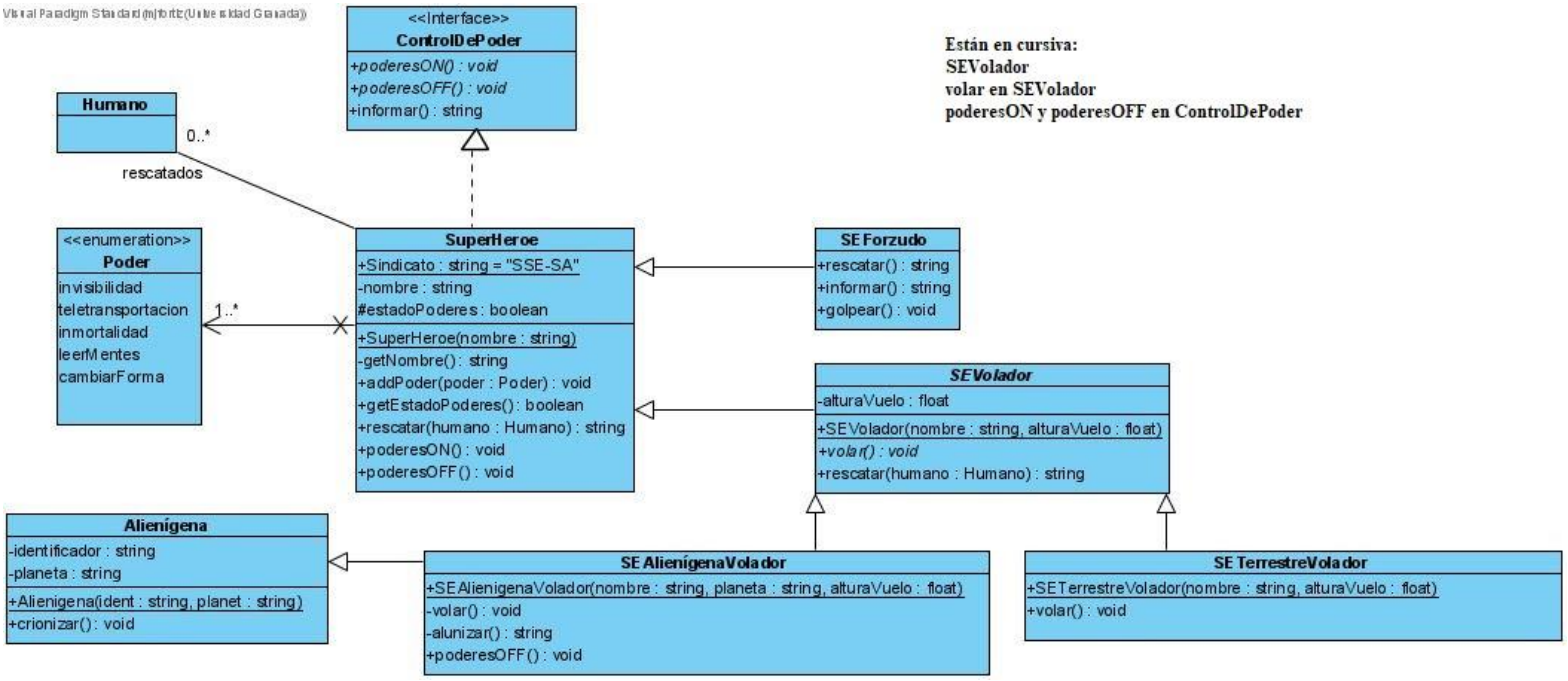

1. Indica verdadero o falso:

- La relación entre ControlDePoder y SuperHeroe es de dependencia
- SEVolador es una interfaz porque está en cursiva, aunque no lleve <<interface>>, es otra forma de representarla
- En el enumerado Poder hay un atributo de la clase SuperHeroe
- En la interface ControlDePoder podría añadirse una constante, por ejemplo, DuracionPoderes, siempre que sea protected
- El método volar está redefinido en las clases que heredan de SEVolador
- El método informar está sobrecargado en SEForzudo
- Un objeto de la clase SEForzudo no puede consultar su atributo nombre en Java
- A los objetos de la clase SEVolador se les puede enviar el mensaje informar
- La clase SEAlienigenaVolador hereda todos los atributos de SEVolador excepto los privados
- Si SETerrestreVolador fuera una clase abstracta necesitaría añadir al menos un método abstracto
- Si en SuperHeroe estuviera el atributo identificador, habría conflicto de nombres en SEAlienigenaVolador
- Si en la clase SEVolador estuviera el método crionizar(), habría conflicto de nombres en la clase SEAlienigenaVolador
- La clase SEVolador tiene un atributo de referencia que es una colección de objetos del enumerado Poder
- Si en Java, en la clase SEVolador se cambiara el valor del atributo Sindicato con un método, ese valor cambiaría también en su clase padre y sus clases hijas
- En el siguiente código:

```
SuperHeroe batman= new SETerrestreVolador("Batman", "Tierra", 1.2);
batman.rescatar();
```

hay ligadura dinámica en el método rescatar que se debe ejecutar se decide en tiempo de ejecución dependiendo de la clase de la variable batman

- Hay error en este código Ruby en el método alunizar de SEAlienigenaVolador (donde otro es un objeto conocido de la clase SuperHeroe y nombre es el consultor del atributo con el mismo nombre)

```
puts "el nombre de mi amigo es:  " + otro. nombre + " y
rescatamos humanos en peligro"
```

- Hay error en este código Java en el método rescatar de SuperHeroe

```
return "me llamo " + nombre + " y rescato humanos en peligro"
```

- En el siguiente código en Java, el tipo estático de la variable superman es SEAlienigenaVolador y su tipo dinámico SEAlienigenaVolador

```
SEVolador superman= new SEAlienigenaVolador("superman", "Krypton", 5.8);
```

2.Indica en qué líneas no hay error, hay error de compilación o hay error de ejecución.

```
SuperHeroe sh2= new SEAlienigenaVolador("capitanZ", "Smirk", 3.8);
SEVolador sh3= new SEVolador("Anaceto", 1);
ControlDePoder sh4= new Alienigena("ET", "Micasa");
SEVolador sh5= new SETerrestreVolador("JunLee", 2);
sh5= SEAlienigenaVolador("Crushi", "Crush", 200)
ArrayList<SEVolador> coleccion= new ArrayList<>();
coleccion.add(sh5);
coleccion.add(sh2);
coleccion.get(0).alunizar();
```

3.Implementar en Ruby el método rescatar de SEVolador para que llame a volar y si la altura del vuelo es mayor de 50 llame luego al método rescatar de la superclase

4.Implementa en Java las siguientes clases e interfaces completas, incluyendo la implementación interna de los constructores que aparezcan en el diagrama:
- SuperHeroe
- ControlDePoder (donde el método informar devuelve el string "tengo el poder")

5. Crea una clase parametrizada llamada Libreria que permita gestionar libros de diferentes tipos. Como atributos tiene el nombre de tipo String y una lista de libros del tipo del parámetro. Crea un método para añadir un libro que recibe como parámetro en la primera posición de la lista libros y otro para consultar el último libro de dicha lista. La cabecera de la clase sería:

```
public class Libreria<T extends Libro>
```

6. Dada la siguiente declaración en una clase del mismo paquete que las del diagrama:

```
SuperHeroe sh3= new SEAlienigenaVolador("R74", "Marte", 1.5);
```

Indica en qué líneas hay error de compilación o de ejecución y cómo se corregirían

```
sh3.getNombre();
sh3.getEstadoPoderes();
sh3.alunizar();
((SETerrestreVolador)sh3).golpear();
```

7. Indica cómo harías para hacer copia profunda de los objetos de la clase SuperHeroe.