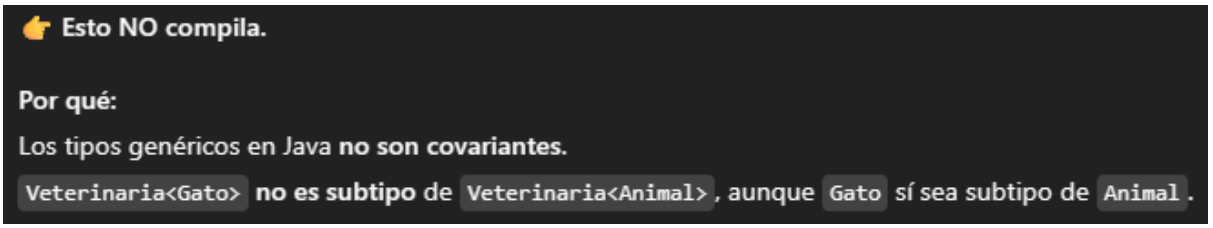
2.- Considere la siguiente clase:

```
public class Veterinaria<E> {
    private E animal;

    public void setAnimal(E x) {
        animal = x;
    }
    public E getAnimal() {
        return animal;
    }
}

public class Animal{
}

public class Gato extends Animal {
}

public class Perro extends Animal {
}
```

a) Indicar cual es el resultado de las siguientes operaciones:
i) Veterinaria <Animal> vet = new Veterinaria <Gato>();



ii) Veterinaria <Gato> vet = new Veterinaria <Animal>(); No
Veterinaria <? Super Gato> vet = new Veterinaria <Animal>(); Si que anda, le dice que vet puede ser de cualquier superclase de gato
iii) Veterinaria <?> vet = new Veterinaria<Gato>();
vet.setAnimal(new Gato());
solo se pueden hacer operaciones de lectura
iv) Veterinaria vet = new Veterinaria ();
vet.setAnimal(new Perro());Bien

👉 **Esto compila, pero no es seguro.**

Por qué:

- Acá se está usando un **raw type** (tipo sin genérico).
- No hay chequeo de tipos en compilación.
- Podés poner cualquier cosa dentro:

```java
vet.setAnimal(new Perro());
vet.setAnimal("Hola");   // también compila
```

v) Veterinaria vet = new Veterinaria <?>();

👉 ❌ **Esto NO compila.**

Por qué:

- No podés usar `<?>` con el operador `new`.
- `<?>` no es un tipo concreto — denota **una familia de tipos**, no una clase instanciable.
- Es igual que intentar `new List<?>();` → no se puede.

vi) Veterinaria <? extends Animal> vet = new Veterinaria<Gato>();Bien

Por qué:

- `Gato` **extiende** `Animal`.
- `Veterinaria<? extends Animal>` significa "una Veterinaria de algún tipo que es subtipo de `Animal`".
- `Gato` entra perfectamente en ese rango.

Pero atención ⚠️:

- Con `? extends T` no podés usar `setAnimal` con nada distinto de `null`:

```java
vet.setAnimal(new Gato());  // ❌ NO compila
vet.setAnimal(null);        // ✅
```

Ej4.

La expresión `Alumno::getNota` **significa**:

> "Usá el método `getNota()` de la clase `Alumno`
> como función que toma un Alumno y devuelve un int".

🧠 Cómo leerlo mentalmente

Sintaxis	Se lee como	Qué hace
`Alumno::getNota`	"el método getNota del Alumno"	se pasa como función que extrae la nota
`System.out::println`	"el método println de System.out"	se pasa como función que imprime
`String::toUpperCase`	"el método toUpperCase de String"	convierte un string a mayúsculas