Objetos inmutables en Java

Un objeto inmutable es aquel cuyo estado no se puede cambiar una vez construido. Todos sus atributos han sido definidos como **`final`** y/o utiliza ***copia defensiva*** para protegerse frente a cambios desde el código cliente.

Por ejemplo, los objetos `String` e `Integer` de Java son inmutables. Si echas un vistazo a la documentación Java de estas clases, verás que todos los métodos que en principio alterarían el estado interno de uno de estos objetos, en realidad **devuelven** un nuevo objeto. Por ejemplo, en los métodos `concat()`, `replace()` o `trim()` de la clase `String`, el objeto original no es alterado.

```
String h = "HOLA";
h.concat(" Y ADIOS");
```

deja `h` inalterado. ¿por qué? Para modificar `h`,

```
h = h.concat(" Y ADIOS”);
```

Entonces, ¿cuántos objetos `String` se crean a continuación?

```
h = "HOLA";
h = h + " Y ADIOS";
```

Objetos inmutables en Java

Crear una clase inmutable

Para crear una clase inmutable (desde el punto de vista del código cliente) deberíamos seguir estas reglas:

1. No proporcionar ningún método que permita modificar el objeto (como p. ej. los ***`setter`***).

2. Usar visibilidad **privada** para los atributos

3. Definir los atributos como **`final.`** Aunque puede no ser estrictamente necesario si no hay ***`setters`***, pero si sería conveniente, ya que también previene la modificación interna.

4. Usar ***copia defensiva*** para las **referencias internas** a objetos mutables. Esto implica:

    1. No inicializar nunca referencias internas simplemente copiando una referencia externa, por ejemplo, en los constructores.

    2. No devolver nunca una referencia interna tal cual, por ejemplo, en los ***`getters`***.

5. Declarar la clase como **`final`**

Objetos inmutables en Java

Por ejemplo, supongamos que tenemos las clases **Punto** y **Circulo** y queremos que **Circulo** sea inmutable.

```
class Circulo {
  public Circulo(Punto p, double r) {
    origen = p;
    radio  = r; }
  public Punto   getOrigen() {return origen;}
  public double  getRadio()  {return radio;}
  private Punto  origen;
  private double radio;
}
```

¿Es esta clase inmutable?

Comprueba si cumple las reglas anteriores o intenta crear un Circulo y modificar su centro o su radio; si puedes hacerlo, entonces Circulo no es inmutable.

Solución:

```
final class Circulo {       //regla 5: clase 'final'
public Circulo(Punto p, double r) {
  origen = new Punto(p);   // regla 4.1: copia defensiva
  radio = r;    // no es una referencia, no hay problema.}
public Punto getOrigen() {
  return new Punto(origen); // regla 4.2: copia defensiva
}
public double getRadio() { return radio;}
final private Punto origen;    // reglas: 2 y 3
final private double radio;    // reglas: 2 y 3
                               // regla 1: no hay setters
}
```

Ref. https://www.dlsi.ua.es//asignaturas/prog3/Objetos_inmutables.html