16. Programación Orientada a Objetos(II)

16.6 Relación de herencia.

Queremos hacer un SuperGuerrero, que es igual a un Guerrero, pero tiene un método ataqueEspecial que el guerrero normal no tiene, además tiene un atributo int numAtaquesEspeciales, que indica el número de ataques especiales que puede hacer hasta que se le agoten.

Para no escribir casi todo el código de nuevo y que los cambios en guerrero se reflejen en SuperGuerrero podemos hacer una subclase.

```
public class SuperGuerrero extends Guerrero{

    // Por defecto un SuperGuerrero puede hacer 5 ataques especiales sin recargar.
    static final int ATAQUES_ESPECIALES_DEFAULT = 5;
    static final int DAÑO_ESPECIAL_MAX = 3;

    int ataquesEspeciales;

    public SuperGuerrero(String nombre, int fuerza, int defensa, int destreza, int ataquesEspeciales) {

        // LLamamos al constructor de la superclase, un Superguerrero es un guerrero.
        super(nombre, fuerza, defensa, destreza);
        // Inicializamos el atributo propio.
        this.ataquesEspeciales = ataquesEspeciales;

    }



    public SuperGuerrero(String nombre) {
        super(nombre);
        this.ataquesEspeciales = ATAQUES_ESPECIALES_DEFAULT;
    }


    public void ataqueEspecial(Guerrero g) {

        //El ataque especial hace un ataque normal y después quita un valor aleatorio de daño mágico.

        this.atacar(g); //Atacar se hereda de la superclase, lo hereda de guerrero.

        // Si le quedan ataques especiales lo realiza
        if (ataquesEspeciales>0) {
            g.setVida(g.getVida()-(int)(DAÑO_ESPECIAL_MAX * Math.random()));
            ataquesEspeciales--;
        }

    }
```

```java
        //El ataque especial hace un ataque normal y después quita un valor aleatorio de daño mágico.

        this.atacar(g); //Atacar se hereda de la superclase, lo hereda de guerrero.

        // Si le quedan ataques especiales lo realiza
        if (ataquesEspeciales>0) {
            g.setVida(g.getVida()-(int)(DAÑO_ESPECIAL_MAX * Math.random()));
            ataquesEspeciales--;
        }

    }


    // Si no sobreescribimos el toString seguiría escribiendo solo los datos del guerrero normal.

    @Override
    public String toString() {
        return "SuperGuerrero [ataquesEspeciales=" + ataquesEspeciales + ", getNombre()=" + getNombre()
                + ", getFuerza()=" + getFuerza() + ", getDefensa()=" + getDefensa() + ", getDestreza()=" + getDestreza()
                + ", getVida()=" + getVida() + ", toString()=" + super.toString() + ", getClass()=" + getClass()
                + ", hashCode()=" + hashCode() + "]";
    }




    static public void main(String args[]) {

        Guerrero yasser = new Guerrero("Yasser", 18, 15, 22);
        SuperGuerrero chenard = new SuperGuerrero("Chenard");



        // El superguerrero puede usar el método atacar heredado como su método ataqueEspecial.
        chenard.atacar(yasser);
        chenard.ataqueEspecial(yasser);


        yasser.atacar(chenard);


        System.out.println(yasser);
        System.out.println(chenard);

    }


}
```

Habíamos visto la referencia **this,** que hacía referencia al propio objeto, ahora podemos usar también la referencia **super**, que hará referencia a la superclase.

```java
// Si no sobreescribimos el toString seguiría escribiendo solo los datos del guerrero normal.

@Override
public String toString() {
    return "SuperGuerrero [ataquesEspeciales=" + ataquesEspeciales + ", getNombre()=" + getNombre()
            + ", getFuerza()=" + getFuerza() + ", getDefensa()=" + getDefensa() + ", getDestreza()=" + getDestreza()
            + ", getVida()=" + getVida() + ", toString()=" + super.toString() + ", getClass()=" + getClass()
            + ", hashCode()=" + hashCode() + "]";
}



static public void main(String args[]) {

    Guerrero yasser = new Guerrero("Yasser", 18, 15, 22);
    SuperGuerrero chenard = new SuperGuerrero("Chenard");



    // El superguerrero puede usar el método atacar heredado como su método ataqueEspecial.
    chenard.atacar(yasser);
    chenard.ataqueEspecial(yasser);


    yasser.atacar(chenard);


    System.out.println(yasser);
    System.out.println(chenard);

}
```

Ejercicio 16.1

Crea una subclase de Guerrero que sea GuerreroSanador, con método sanar(Guerrero), que restaure la vida del guerrero al 100%.

Ejercicio 16.2

Crea una clase rectángulo con los métodos area() y perímetro(). Crea una subclase cuadrado que no necesite programar los métodos área y perímetro, sino que los herede.

Ejercicio 16.3

Crea la clase ***Vehiculo***, así como las clases ***Bicicleta*** y ***Coche*** como subclases de

la primera. Para la clase ***Vehiculo***, crea los atributos de clase ***modelo*** y **kilometrosTotales**, así como el atributo de instancia **kilometrosRecorridos**. Crea también algún método específico para cada una de las subclases. Prueba las clases creadas mediante un programa con un menú como el que se muestra a continuación:

```
VEHÍCULOS

=========

1. Anda con la bicicleta

2. Haz el caballito con la bicicleta

3. Anda con el coche

4. Quema rueda con el coche

5. Ver kilometraje de la bicicleta

6. Ver kilometraje del coche

7. Ver kilometraje total

8. Salir

Elige una opción (1-8):
```