

Argentina
programa


Programación Orientada a Objetos con JAVA

Guía IV

Ejercicios

EJERCICIOS DE APRENDIZAJE

En este módulo de POO, vamos a empezar a ver cómo dos o más clases pueden relacionarse entre si mediante una herencia de clases.

	VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.
---	--

Importante!!!

Hacer diagrama UML que muestre esta realidad esta realidad representando correctamente las relaciones de asociación, dependencia y herencia.

1. Tenemos una clase padre Animal junto con sus 3 clases hijas Perro, Gato, Caballo. La clase Animal tendrá como atributos el nombre, alimento, edad y raza del Animal.
Crear un método en la clase Animal a través del cual cada clase hija deberá mostrar luego un mensaje por pantalla informando de que se alimenta. Generar una clase Main que realice lo siguiente:

```
1 public class Main {
2
3     public static void main(String[] args) {
4
5         //-->Declaracion del objeto PERRO
6         Animal perro = new Perro("Stich","Carnivoro",15,"Doberman");
7         perro.Alimentarse();
8         //-->Declaracion de otro objeto PERRO
9         Perro perro1 = new Perro("Teddy","Croquetas",10,"Chihuahua");
10        perro1.Alimentarse();
11
12
13        //-->Declaracion del objeto Gato
14        Animal gato = new Gato("Pelusa","Galletas",15,"Siames");
15        gato.Alimentarse();
16        //-->Declaracion del objeto Caballo
17        Animal caballo = new Caballo("Spark","Pasto",25,"Fino");
18        caballo.Alimentarse();
19
20    }
21 }
```

2. Crear una superclase llamada Electrodoméstico con los siguientes atributos: precio, color, consumo energético (letras entre A y F) y peso.

Los constructores que se deben implementar son los siguientes:

- Un constructor vacío.
- Un constructor con todos los atributos pasados por parámetro.

Los métodos a implementar son:

- Métodos getters y setters de todos los atributos.
- Método **comprobarConsumoEnergetico(char letra)**: comprueba que la letra es correcta, sino es correcta usara la letra F por defecto. Este método se debe invocar al crear el objeto y no será visible.

- Método **comprobarColor(String color)**: comprueba que el color es correcto, y si no lo es, usa el color blanco por defecto. Los colores disponibles para los electrodomésticos son blanco, negro, rojo, azul y gris. No importa si el nombre está en mayúsculas o en minúsculas. Este método se invocará al crear el objeto y no será visible.
- Método **precioFinal()**: según el consumo energético y su tamaño, aumentará el valor del precio. Esta es la lista de precios:

LETRA	PRECIO
A	\$1000
B	\$800
C	\$600
D	\$500
E	\$300
F	\$100

PESO	PRECIO
Entre 1 y 19 kg	\$100
Entre 20 y 49 kg	\$500
Entre 50 y 79 kg	\$800
Mayor que 80 kg	\$1000

A continuación se debe crear una subclase llamada Lavadora, con el atributo carga, además de los atributos heredados.

Los constructores que se implementarán serán:

- Un constructor vacío.
- Un constructor con la carga y el resto de atributos heredados. Recuerda que debes llamar al constructor de la clase padre.

Los métodos que se implementara serán:

- Método get y set del atributo carga.
- Método **precioFinal()**: este método será heredado y se le sumará la siguiente funcionalidad. Si tiene una carga mayor de 30 kg, aumentará el precio en \$500, si la carga es menor o igual, no se incrementará el precio. Este método debe llamar al método padre y añadir el código necesario. Recuerda que las condiciones que hemos visto en la clase Electrodoméstico también deben afectar al precio.

Se debe crear también una subclase llamada Televisor con los siguientes atributos: resolución (en pulgadas) y sintonizador TDT (booleano), además de los atributos heredados.

Los constructores que se implementarán serán:

- Un constructor vacío.
- Un constructor con la resolución, sintonizador TDT y el resto de atributos heredados. Recuerda que debes llamar al constructor de la clase padre.

Los métodos que se implementara serán:

- Método get y set de los atributos resolución y sintonizador TDT.

- Método **precioFinal()**: este método será heredado y se le sumará la siguiente funcionalidad. Si el televisor tiene una resolución mayor de 40 pulgadas, se incrementará el precio un 30% y si tiene un sintonizador TDT incorporado, aumentará \$500. Recuerda que las condiciones que hemos visto en la clase Electrodomestico también deben afectar al precio.

Finalmente, en el main debemos realizar lo siguiente:

Vamos a crear una Lavadora y un Televisor y llamar a los métodos necesarios para mostrar el precio final de los dos electrodomésticos.

3. Se plantea desarrollar un programa que nos permita calcular el área y el perímetro de formas geométricas, en este caso un círculo y un rectángulo. Ya que este cálculo se va a repetir en las dos formas geométricas, vamos a crear una Interfaz, llamada `calculosFormas` que tendrá, los dos métodos para calcular el área, el perímetro y el valor de PI como constante.

Desarrollar el ejercicio para que las formas implementen los métodos de la interfaz y se calcule el área y el perímetro de los dos. En el main se crearán las formas y se mostrará el resultado final.

*Área círculo: $PI * radio^2$ / Perímetro círculo: $PI * diámetro$.*

*Área rectángulo: $base * altura$ / Perímetro rectángulo: $(base + altura) * 2$*

4. En un nuevo proyecto Java se pide:

a) Crear una clase de nombre **Position** con los siguientes atributos:

x de tipo entero.

y de tipo entero.

Sus métodos getter and setters

Constructor con todos los atributos.

b) Crear una clase abstracta de nombre **Personaje** con los siguientes atributos:

-ubicación de tipo `Position`

-nick de tipo `String`

-vidas de tipo `int` inicializado en 3.

-energía de tipo `int` inicializado en 100.

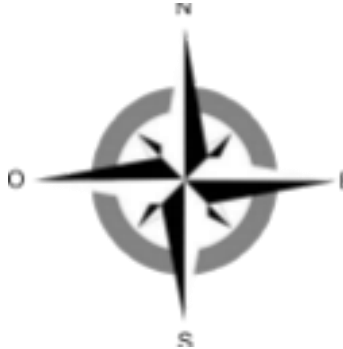
-orientación de tipo `char` ('N' norte, 'S' sur, 'E' este, 'O' oeste). Inicializado en 'N'

-un constructor que permita inicializar su ubicación y Nick.

Con los métodos:

Disparar que consume 10 unidades de energía y podrá disparar hasta que consuma toda su energía.

Girar que cambia secuencialmente de orientación N, E, S, O y vuelve nuevamente N,E,S,O.....



Avanzar que si el personaje está mirando hacia el norte: crece en 1 el valor Y de su posición; si está mirando al Sur: decrece en 1 el valor de Y, si mira al Este: crece en 1 el valor de X, y si mira al Oeste: decrece en 1 el valor de X.

c) Crear una clase de nombre **Guerrero** que es un **Personaje** con los siguientes atributos adicionales:

-caballo de tipo bool.

Con los métodos:

Sobreescribir el comportamiento de **Avanzar** para que si tiene un caballo avanzará de a 10 pasos caso contrario se comportará como el método de la clase padre.

Sobreescribir el comportamiento de **Disparar** para que si tiene menos de 30 unidades de energía pierda el caballo.

d) Luego en una clase **TestHerencia**, desde su método main se pide:

- Crear un Guerrero de nombre “Thor” en la posición X=100, Y=200
- Hacerlo girar hasta que mire al Oeste
- Hacerlo Avanzar 5 pasos.
- Hacerlo disparar 8 veces.