



Argentina
programa

Programación Orientada a Objetos con JAVA


Guía III

Ejercicios

EJERCICIOS DE APRENDIZAJE

En este módulo de POO, vamos a empezar a ver cómo dos o más clases pueden relacionarse entre sí, a través de asociación o dependencia.

Nota: instalar el easyUML por si queremos ver nuestros proyectos como diagramas UML. Encontrarán como instalarlo en Moodle.

	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.
---	--

1. Realizar un programa para que una Persona pueda adoptar hasta tres Perros. Vamos a contar de dos clases. Perro, que tendrá como atributos: nombre, raza, edad y tamaño, un constructor que permita inicializar todos sus atributos y los respectivos métodos getter y setter de sus atributos; y la clase Persona con atributos: nombre, apellido, edad, documento y tres atributos de tipo Perro, un constructor que sólo permita inicializar los atributos: nombre, apellido, edad y documento. La Persona cuenta además con los siguientes métodos:

adoptarPerro() Este método recibe por parámetro un Perro y lo asigna en alguna de las variables vacías; si todas las variables atributos Perro están ocupadas, mostrará un mensaje “Ya no puedo adoptar”.

perroMasGrande() Este método retornará el Perro de mayor edad.

Ahora deberemos en el main crear una Persona y 4(cuatro) Perros. Después, vamos a hacer que persona adopte cada uno de esos Perros (luego al ejecutar veremos que pasa), por último pediremos a la Persona que nos diga cual es el Perro más viejo.

2. Un Kiosco tiene 3 Empleados. Vamos a contar de dos clases. Kiosco, que tendrá los atributos: dirección, nombre, cuit y 3 atributos de tipo Empleado; y la clase Empleado posee los atributos: nombre, apellido, fecha de ingreso y dni. Usted deberá pensar en que funcionalidad incorporar en la clase **Kiosoco** para que podamos incorporar sólo hasta 3 Empleados y poder mostrar cual es el Empleado con mayor antigüedad.

Ahora en el método main de la clase principal usted creará un Kiosco y 3 Empleados, luego incorporará al Kisoco esos Empleados y le solicitará al Kiosco que muestre los datos del Empleado de mayor antigüedad.

3. Nos piden modelar un **Auto** con las siguientes características y comportamiento para poder ser anexado al proyecto.

A- El auto debe tener: color, patente y combustible con una carga inicial de 50lts. El auto sólo tendrá como comportamiento:

- **Avanzar:** Al que le indicaremos la cantidad de metros que deseamos avance y deberemos tener en cuenta que por cada 10 mts consume 1lt de combustible y solo podrá avanzar si hay combustible suficiente.

- **Retroceder:** Al que le indicaremos la cantidad de metros que deseamos retroceda y deberemos tener en cuenta que por cada 10 mts consume 1lt de combustible y solo podrá avanzar si hay combustible suficiente.

- **LlenarTanque:** Llenará el tanque de este auto nuevamente con 50lts de combustible.

B- Hacer la clase **Rueda**, con atributos: marca y presión, Luego el auto debe tener 4 ruedas, además Rueda tendrá los siguientes comportamientos:

- Inflar: la presión regresa a 28.0 PSI
- Pinchar: se debe reducir la presión al mínimo.
- Desinflar: reduce la presión de la rueda -0.5 PSI

C- Desde el método **main** crear las instancias, relacionarlas, e inflar/desinflar sus ruedas, avanzar/retroceder, y llenarTanque.

Nota: La medida en PSI (libra por pulgada cuadrada). Generalmente los turismos ligeros suelen llevar una presión recomendada comprendida entre los 28 a 30 PSI.

4. Disponemos de 3 objetos conocidos: un **Hombre**, un **Robot** y una **Bateria**; sabemos que el Robot reconoce las órdenes:

- Avanzar(cantidad de pasos)
- Retroceder(cantidad de Pasos)
- Dormir()
- Despertar()
- Recargar()
- bateriaLLena():boolean
- bateriaVacía() :boolean
- energiaActual() :int



También sabemos que un **Robot** tiene una batería con 1000 unidades de energía y que cada vez que avanza o retrocede por cada 100 pasos pierde 10 unidades además si damos la orden al robot de dormir, no responderá al avanzar o retroceder hasta despertarlo.

Para volver a recargar las baterías del robot, bastará con ordenarle que recargue.

La **Bateria** tiene como atributos carga que es un valor entero, posee un constructor que permite inicializar su carga y los métodos get y set para dicho atributo.

Por otro lado el **Hombre** sabe:

- **JugarConRobot(Robot)** Este método recibe por parámetro un Robot y con hace lo siguiente:

- Hacer que el Robot Avance 500 pasos.
- El robot Retroceda 20 pasos.
- Informe por pantalla cuanta energía tiene el robot Actualmente.
- Ponga el robot a dormir.

Luego desde el método main de una clase **TestJuego**, se pide:

- Crear un Robot.
- Crear un Hombre y pasarle el Robot creado en el ítem anterior.
- Hacer que el Hombre juegue con el Robot.
- Crear otro Hombre y pasarle el mismo Robot
- Hacer que el último Hombre creado juegue con el Robot,

Extra:

Modificar al método jugar con robot para que reciba por teclado los pasos que desea que avance o retroceda el robot las veces que quiera y finalice sólo cuando ponga a dormir al robot. (Puede utilizar la clase Scanner para leer desde el teclado).