CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

TEMA9.
PROGRAMACIÓN ORIENTADA A OBJETOS (II)

PROGRAMACIÓN
CFGS DAW

EJERCICIOS

Carlos Cacho
Raquel Torres
carlos.cacho@ceedcv.es
raquel.torres@ceedcv.es
Versión:180214.2042

Licencia

UD09. PROGRAMACIÓN ORIENTADA A OBJETOS (II)

EJERCICIOS

1. Supongamos una clase Producto con dos atributos:
   - nombre: String.
   - cantidad: int.

Definir esta clase, con dos constructores (uno con parámetros y otro sin) y métodos para acceder a sus atributos y modificarlos.

A continuación, en el programa principal:

- Define 5 instancias de la Clase Producto.
- Define un ArrayList.
- Añade las 5 instancias de Producto en ArrayList, utiliza los dos métodos add().
- Visualiza el contenido de ArrayList utilizando Iterator.
- Elimina un elemento de ArrayList.
- Visualiza de nuevo el contenido de ArrayList utilizando Iterator.
- Elimina todos los valores del ArrayList.

2. Define una jerarquía de clases que permita almacenar datos sobre los planetas y satélites que forman parte del sistema solar (junto con el sol).

   Algunos atributos que puede ser interesante recoger son:

   - la masa del cuerpo.
   - su diámetro medio.
   - el período de rotación sobre el propio eje.
   - período de traslación alrededor del cuerpo que orbitan.
   - distancia media a ese cuerpo, ...

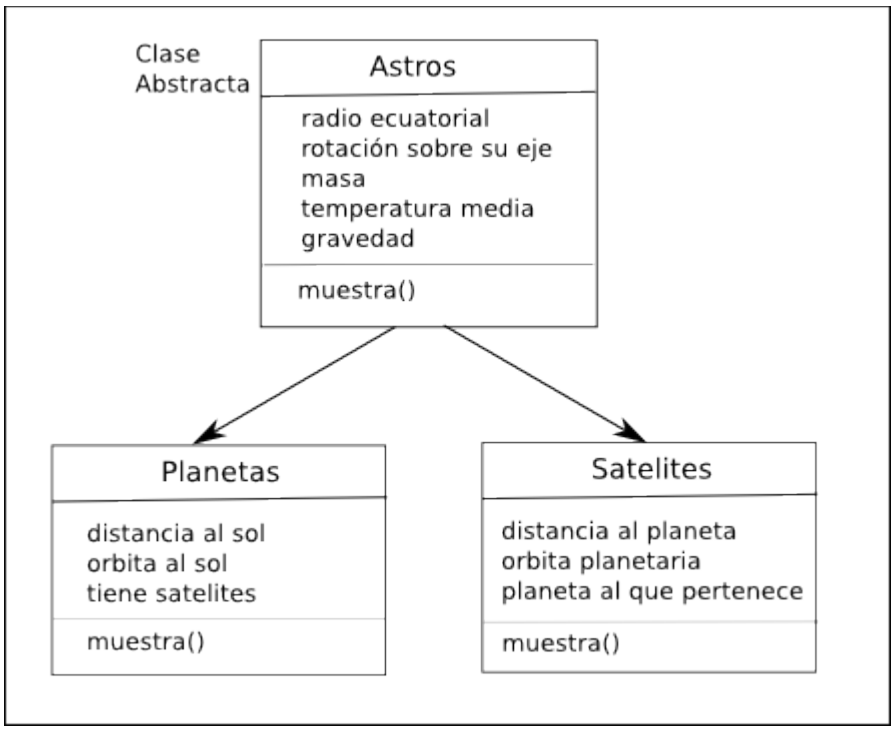
   Define las clases necesarias conteniendo:

   - constructores.
   - métodos para recuperar y almacenas atributos
   - método para mostrar la información del objeto.

Define un método, que dado un objeto del sistema solar (planeta o satélite), imprima toda la información de que se dispone sobre el mismo.

Una posible solución sería, crear una lista de objetos, introducir los planetas y satélites, bien sea directamente o solicitándolos por pantalla (no es necesario escribirlos todos) y solicitar el nombre de un Astro y muestre su información.

El diagrama UML sería:

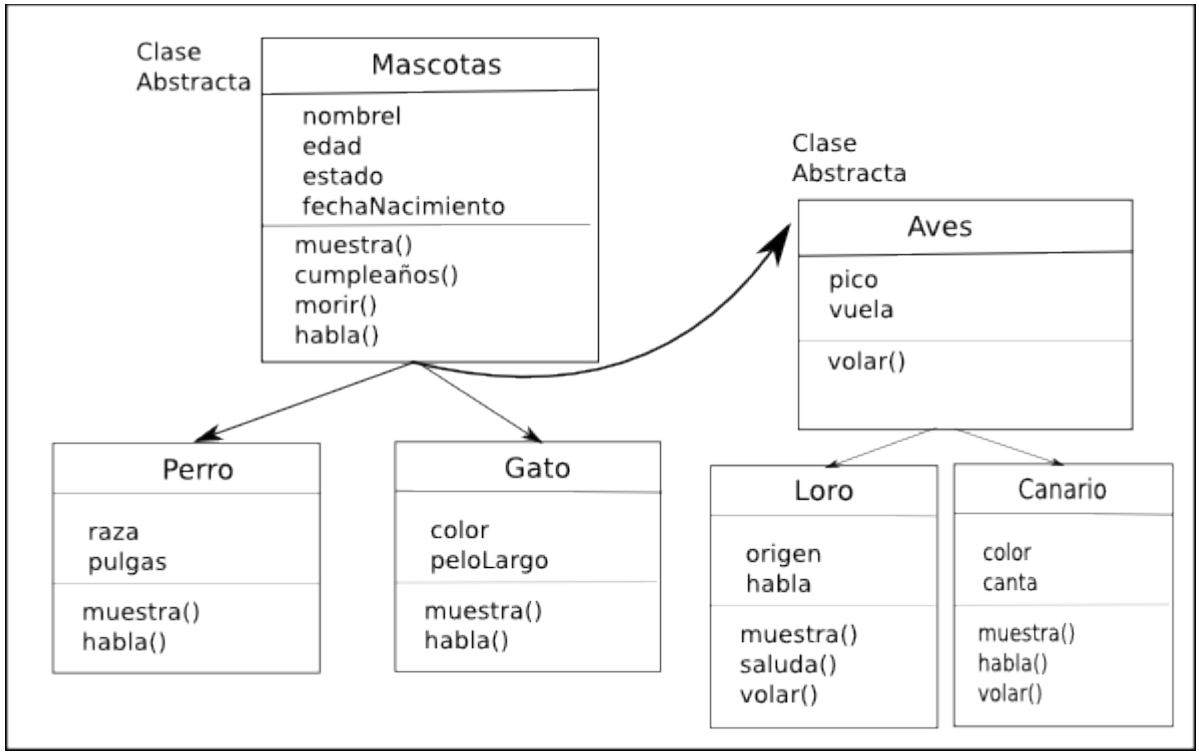


3. Escribe una clase llamada **Inventario**, para almacenar referencias a todos los animales existentes en una tienda de mascotas.

   Esta clase debe cumplir con los siguientes requisitos:

   - En la tienda existirán diferentes tipos de animales: perros, gatos, loros y canarios.
   - Los animales deben almacenarse en un ArrayList privado dentro de la clase **Inventario**.
   - La clase debe tener un menú que permita:
     - inicializa el inventario a vacío.
     - método para insertar animales del inventario.
     - método para eliminar animales del inventario.
     - método que imprima los datos de todos los animales de la tienda.
     - método que imprima los datos de todos los perros de la tienda.

   El diagrama UML sería:



4. Vamos a hacer una aplicación que modelice el funcionamiento de un banco.

   Crear una clase **CuentaBancaria** con los atributos:

   - **numeroCuenta** (que se pasará en el constructor)
   - **saldo**.

   Implementad los métodos:
   - para consultar estos atributo.
   - para ingresar.
   - retirar dinero.
   - para hacer un traspaso de una cuenta a otra.

   Para los tres últimos métodos puede utilizarse un **método privado**:
   añadir(double cantidad)

   que añada una cantidad (positiva o negativa) al saldo.

   También habrá un atributo común a todas las instancias **interesAnualBasico**, que en principio puede ser una constante.

   La clase tiene que ser ***abstracta*** y debe tener un método

calcularIntereses() que se dejará sin implementar. De esta clase heredarán dos: **CuentaCorriente** y **CuentaAhorro**. La diferencia entre ambas será la manera diferente de calcular los intereses. A la primera se le incrementará el saldo teniendo en cuenta el interés anual básico. La segunda tendrá una constante de clase saldoMinimo. Si no se llega a este saldo el interés será la mitad del interés básico, si se supera el doble.

Además, define un método abstracto mostrar() en la clase CuentaBancaria para mostrar los datos del objeto.

El diagrama UML sería: