CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNITAT 9
PROGRAMACIÓ ORIENTADA A OBJECTES II
EXERCICIS

PROGRAMACIÓ
CFGS DAW

Autors:
Carlos Cacho y Raquel Torres
Revisat per:
Lionel Tarazon - lionel.tarazon@ceedcv.es
Fco. Javier Valero – franciscojavier.valero@ceedcv.es
José Manuel Martí - josemanuel.marti@ceedcv.es

2021/2022

EXERCICI 1 - PRODUCTE

Suposem una classe Producte amb dos atributs:

- String nom
- int quantitat

Implementa aquesta classe amb un constructor (amb paràmetres) a més dels getters i setters dels seus dos atributs. No és necessari comprovar les dades introduïdes.

A continuació, en el programa principal fes el següent:

1. Crea 5 instàncies de la Classe Producte.
2. Crea un ArrayList.
3. Afig les 5 instàncies de Producte al ArrayList.
4. Visualitza el contingut del ArrayList utilitzant Iterator.
5. Elimina dos element del ArrayList.
6. Inserida un nou objecte producte enmig de la llista.
7. Visualitza de nou el contingut de ArrayList utilitzant Iterator.
8. Elimina tots els valors del ArrayList.

EXERCICI 2 - ASTRES

Define una jerarquia de classes que permeta emmagatzemar dades sobre els planetes i satèl·lits (llunes) que formen part del sistema solar.

Alguns atributs que necessitarem emmagatzemar són:

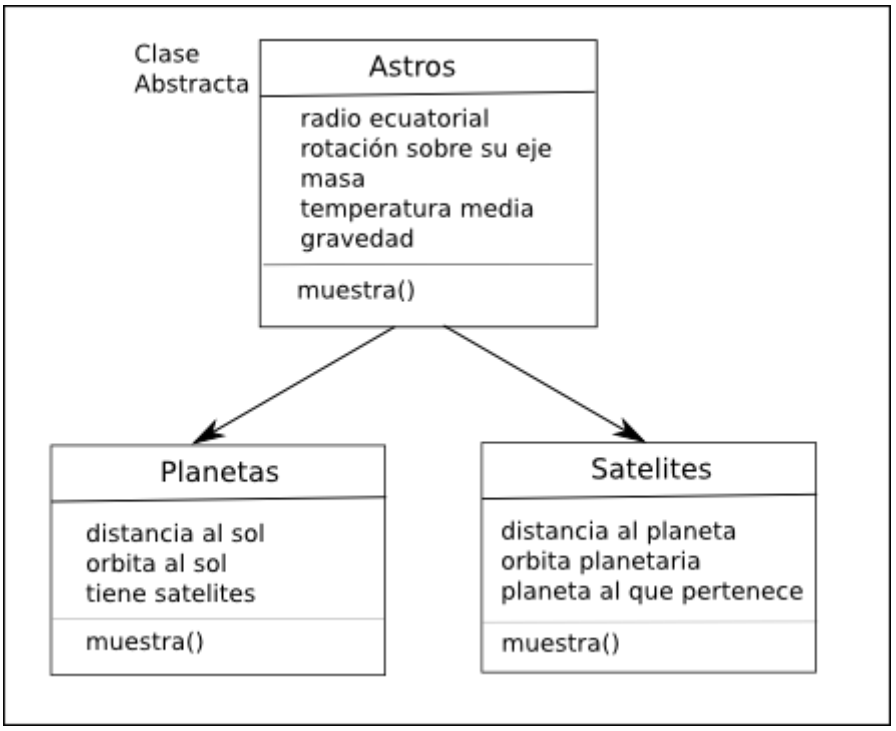
- Massa del cos.
- Diàmetre mitjà.
- Període de rotació sobre el seu propi eix.
- Període de translació al voltant del cos que orbiten.
- Distancia mitjana a aqueix cos.
- etc.

Defineix les classes necessàries contenint:

- Constructors.
- Mètodes per a recuperar i emmagatzemes atributs.
- Mètodes per a mostrar la informació de l'objecte.

Defineix un mètode, que donat un objecte del sistema solar (planeta o satèl·lit), imprimisca tota la informació que es disposa sobre el mateix (a més de la seua llista de satèl·lits si els tinguera).

El diagrama UML seria:



Una possible solució seria crear una llista d'objectes, inserir els planetes i satèl·lits (directament mitjançant codi o sol·licitant-los per pantalla) i després mostrar un xicotet menú que permeta a l'usuari imprimir la informació de l'astre que trie.

EXERCICI 3 - MASCOTES

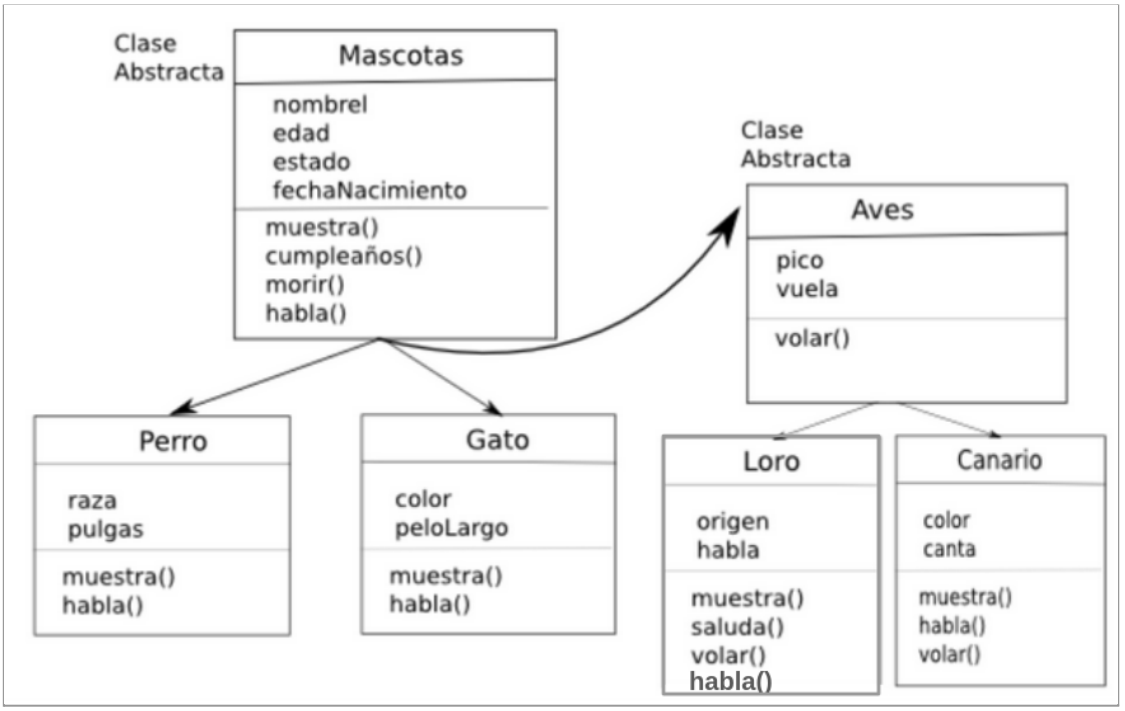
Implementa una classe anomenada **Inventari** que utilitzarem per a emmagatzemar referències a tots els animals existents en una botiga de mascotes.

Aquesta classe ha de complir amb els següents requisits:

- A la botiga existiran 4 tipus d'animals: gossos, gats, lloros i canaris.
- Els animals han d'emmagatzemar-se en un ArrayList privat dins de la classe **Inventari**.
- La classe ha de permetre realitzar les següents accions:
    - Mostrar la llista d'animals (sols tipus i nom, 1 línia per animal).
    - Mostrar totes les dades d'un animal concret.
    - Mostrar totes les dades de tots els animals.
    - Insertar animals en l'inventari.
    - Eliminar animals de l'inventari.
    - Buidar l'inventari.

Implementa les altres classes necessàries per a la classe Inventari.

El diagrama UML seria:



EXERCICI 4 – BANC

Farem una aplicació que simule el funcionament d'un banc.

Crea una classe **CompteBancari** amb els atributs: **IBAN** i **saldo**. Implementa mètodes per a :

- Consultar els atributs.
- Ingressar diners.
- Retirar diners.
- Traspassar diners d'un compte a una altre.

Per als tres últims mètodes pot utilitzar-se internamente un mètode privat més general anomenat **afegir(...)** que afija una quantitat (positiva o negativa) al saldo.

També hi haurà un atribut comú a totes les instàncies anomenat **interesAnualBasic**, que en principi pot ser constant.

La classe ha de ser ***abstracta*** i ha de tindre un mètode **calcularInteressos()** que es deixarà sense implementar.

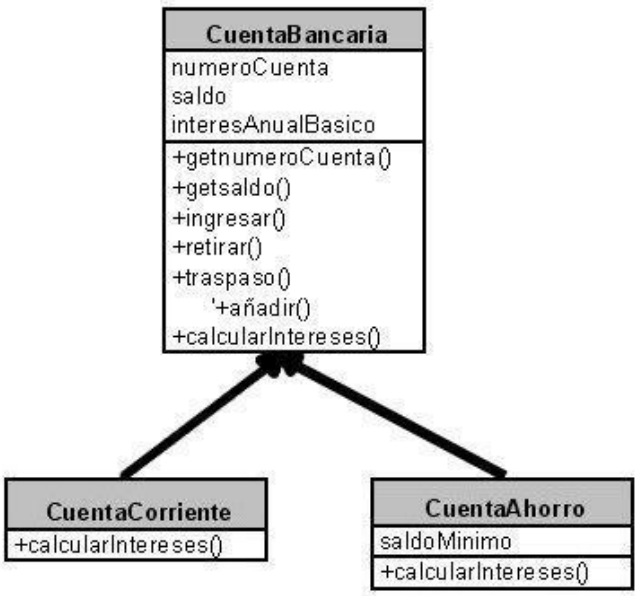
També pot ser útil implementar un mètode per a mostrar les dades del compte.

D'aquesta classe heretaran dues subclasses: **CompteCorrent** i **CompteEstalvi**. La diferència entre ambdues serà la manera de calcular els interessos:

- A la primera se li incrementarà el saldo tenint en compte l'interés anual bàsic.
- La segona tindrà una constant de classe anomenada **saldoMinim**. Si no s'arriba a aquest saldo l'interés serà la meitat de l'interés bàsic. Si se supera el saldo mínim l'interés aplicat serà el doble de l'interés anual bàsic.

Implementa una classe principal amb funció main per a provar el funcionament de les tres classes: Crea diversos comptes bancaris de diferents tipus, poden estar en un ArrayList si ho desitges; prova de realitzar ingressos, retirades i transferències; calcula els interessos i mostra'ls per pantalla; etc.

El diagrama UML seria:



EXERCICI 5 – EMPRESA I EMPLEATS

Implementarem dues classes que permetan gestionar dades d'empreses i els seus empleats.

Els **empleats** tenen les següents característiques:

- Un empleat té nom, DNI, sou brut (mensual), edat, telèfon i adreça.
- El nom i DNI d'un objecte no pot variar.
- És obligatori que tots els empleats tinguen almenys definit el seu nom, DNI i el sou brut. Les altres dades no són obligatoris.
- Será necessari un mètode per a imprimir per pantalla la informació d'un empleat.
- Serà necessari un mètode per a calcular el sou net d'un empleat. El sou net es calcula descomptant del sou brut un percentatge que depén de l'IRPF. El percentatge de l'IRPF depén del sou brut anual de l'empleat (sou brut x 12 pagues).**(*)**

Sou brut anual	**IRPF**
Inferior a 12.000 €	20%
De 12.000 a 25.000 €	30%
Més de 25.000 €	40%

Per exemple, un empleat amb un sou brut anual de 17.000 € tindrà un 30% d'IRPF.

Per a calcular el seu sou net mensual es descomptarà un 30% al seu sou brut mensual.

Les **empreses** tenen les següents característiques:

- Una empresa té nom i CIF (dades que no poden variar), a més de telèfon, adreça i empleats. Quan es crea una nova empresa aquesta manca d'empleats.
- Serán necessaris mètodes per a:
  - Afegir i eliminar empleats a l'empresa.
  - Mostrar per pantalla la informació de tots els empleats.
  - Mostrar per pantalla el DNI, sou brut i net de tots els empleats.
  - Calcular la suma total de sous bruts de tots els empleats.
  - Calcular la suma total de sous nets de tots els empleats.

Implementa les classes Emprat i Empresa amb els atributs oportuns, un constructor, els getters/setters oportuns i els mètodes indicats. Pots afegir més mètodes si ho veus necessari. Aquestes classes no han de realitzar cap mena d'entrada per teclat.

Implementa també una classe Programa amb una funció main per a realitzar proves: Crear una o diverses empreses, crear empleats, afegir i eliminar empleats a les empreses, llistar tots els empleats, mostrar el total de sous bruts i nets, etc.

(*) L'IRPF realment és més complex però s'ha simplificat per a no complicar massa aquest exercici.

EXERCICI 6 - VEHICLES

És molt aconsellable fer el disseny UML abans de començar a programar.

Deus crear diverses classes per a un programari d'una empresa de transport. Implementa la jerarquia de classes necessària per a complir els següents criteris:

- Els vehicles de l'empresa de transport poden ser terrestres, aquàtics i aeris. Els vehicles terrestres poden ser cotxes i motos. Els vehicles aquàtics poden ser vaixells i submarins. Els vehicles aeris poden ser avions i helicòpters.
- Tots els vehicles tenen matrícula i model (dades que no poden canviar). La matrícula dels cotxes terrestres han d'estar formada per 4 números i 3 lletres. La dels vehicles aquàtics per entre 3 i 10 lletres. La dels vehicles aeris per 4 lletres i 6 números.
- Els vehicles terrestres tenen un nombre de rodes (dada que no pot canviar).
- Els vehicles aquàtics tenen eslora (dada que no pot canviar).
- Els vehicles aeris tenen un nombre de seients (dada que no pot canviar).
- Els cotxes poden tindre aire condicionat o no tindre'n.
- Les motos tenen un color.
- Els vaixells poden tindre motor o no tindre'n.
- Els submarins tenen una profunditat màxima.
- Els avions tenen un temps màxim de vol.
- Els helicòpters tenen un nombre d'hèlices.
- No es permeten vehicles genèrics, és a dir, no es deuen poder instanciar objectes que siguen vehicles sense més. Però ha de ser possible instanciar vehicles terrestres, aquàtics o aeris genèrics (és a dir, que no siguen cotxes, motos, vaixells, submarins, avions o helicòpters).
- El disseny ha d'obligar al fet que totes les classes de vehicles tinguen un mètode imprimir() que imprimisca per pantalla la informació del vehicle en una sola línia.

Implementa totes les classes necessàrias amb: atributs, constructor amb paràmetres, getters/setters i el mètode imprimir. Utiliza **abstracció** i **herència** de la forma més apropiada.

Implementa també una classe Programa per a fer algunes proves: Instància diversos vehicles de tota mena (cotxes, motos, vaixells, submarins, avions i helicòpters) així com vehicles genèrics (terrestres, aquàtics i aeris). Crea un ArrayList i afegis tots els vehicles. Recorre la llista i crida al mètode imprimir de tots els vehicles.

EXERCICI 7 - FIGURES

Implementa una **interfície** anomenada **iFigura2D** que declare els mètodes:

- double perimetre(): Per a retornar el perímetre de la figura
- double area(): Per a retornar l'àrea de la figura
- void escalar(double escala): Per a escalar la figura (augmentar o disminuir la seua grandària). Només cal multiplicar els atributs de la figura per l'escala (> 0).
- void imprimir(): Per a mostrar la informació de la figura (atributs, perímetre i àrea) en una sola línia.

Existeixen 4 tipus de figures.

- **Quadrat**: Els seus quatre costats són iguals.
- **Rectangle**: Tenen ample i alt.
- **Triangle**: Tenen ample i alt.
- **Cercle**: Tenen ràdi.

Crea les 4 classes de figures de manera que implementen la interfície iFigura2D. Defineix els seus mètodes.

Crea una classe ProgramaFiguras amb un main en el qual realitzar les següents proves:

1. Crea un ArrayList figures.
2. Afig figures de diversos tipus.
3. Mostra la informació de totes les figures.
4. Escala totes les figures amb escala = 2.
5. Mostra de nou la informació de totes les figures.
6. Escala totes les figures amb escala = 0.1.
7. Mostra de nou la informació de totes les figures.