CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNITAT 9
POO (II)
EXEMPLES

PROGRAMACIÓ
CFGS DAW

Autors:
Joan Vicent Cassany – jv.cassanycoscolla@edu.gva.es
Revisat per:

2022/2023

Llicència

Exemple 01

Utilitzem ArrayList i tots els seus mètodes apresos.

PROGRAMA PRINCIPAL

```
package UF09_Exemple01;

// Importem les classes ArrayList i Itinerator
import java.util.ArrayList;
import java.util.Iterator;
/**
 * UF09 Exemple 01: Creació i gestió d'un ArrayList de Strings
 */
public class UF09_Exemple01 {

    public static void main(String[] args) {

        // Creación de l'ArrayList i ús del mètode size per a vore el tamany
            ArrayList<String> colors = new ArrayList<String>();
            System.out.println("Nombre d'elements: " + colors.size());

        // Afegim els primers elements amb add i mostrem tamany amb size
            colors.add("roig");
            colors.add("verd");
            colors.add("blau");
            System.out.println("Nombre d'elements: " + colors.size());


        // Mostrar la llista de forma bàsica
            System.out.println("Contingut de la llista: " + colors);

        // Afegim més elements i mostrem tamany
            colors.add("blanc");
            colors.add("groc");
            System.out.println("Nombre d'elements: " + colors.size());

        // Mostrar la llista de forma bàsica
            System.out.println("Contingut de la llista: " + colors);
```

```
// Afegim un element en una posició determinada
    colors.add(1, "violeta");
    System.out.println("Contingut de la llista afegint un element en posició 1: " + colors);

// Mostrem els elements que hi ha en una posició amb el mètode get
    System.out.println("L'element que hi ha en la posició 0 és " + colors.get(0));
    System.out.println("L'element que hi ha en la posició 3 és " + colors.get(3));

// Verifiquem el contingut de l'ArrayList amb el mètode contains
    if (colors.contains("blanc")) {
        System.out.println("El blanc està en la llista de colors");
    }

// Verifiquem en quina posició de l'ArrayList està un element amb el mètode indexOf
   System.out.println("El blanc està en la posició " + ((int)colors.indexOf("blanc") + 1));


// Eliminem un element pel contingut i un altre per la posició amb el mètode remove
    colors.remove("blanc");
    System.out.println("Contingut de la llista sense el blanc: " + colors);
    colors.remove(2);
    System.out.println("Contingut de la llista sense el color de la posició 2: " + colors);

// Canviem el color de una posició determinada amb el mètode set
    colors.set(2, "turquesa");
System.out.println("Contingut de la llista amb posició 3 (ind.2) actualitzada: " + colors);

// Mostrar la llista utilitzant un for
    System.out.print("Contingut de la lista utilitzant FOR: ");
    for(int i=0; i<colors.size(); i++) {
        System.out.print(colors.get(i) + " ");
    }
    System.out.println("");

// Mostrar la llista utilitzant un for a l'estil foreach
    System.out.print("Contingut de la lista utilitzant FOREACH: ");
    for(String color: colors) {
        System.out.print(color + " ");
    }
    System.out.println("");
```

```
            // Recorrer la llista utilitzant l'objecte Iterator
                System.out.print("Contingut de la lista utilitzant ITERATOR: ");
                Iterator iter = colors.iterator();      // Creem el Iterator a partir de la llista
                while(iter.hasNext()) {                 // Mentre hi haja següent en la llista
                    System.out.print(iter.next() + " ");
                }
                System.out.println("");
        }
}
```

- Estudia el programa.
- Crea una nova versió utilitzant els mesos de l'any a partir del seguen esquelet de programa. Intenta recordar les instruccions i fes proves fins que dones amb la sol·lució, evitant copiar-les.

```
package UF09_Exemple01B;

// Importem les classes ArrayList i Itinerator

/**
 * UF09 Exemple 01B: Creació i gestió d'un ArrayList de Strings
 */
public class UF09_Exemple01B {

    public static void main(String[] args) {

        // Creación de l'ArrayList i ús del mètode size per a vore el tamany

        // Afegim els primers elements amb add i mostrem tamany amb size

        // Mostrar la llista de forma bàsica

        // Afegim més elements i mostrem tamany


        // Mostrar la llista de forma bàsica

       // Afegim un element en una posició determinada

      // Mostrem els elements que hi ha en una posició amb el mètode get
```

```
        // Verifiquem el contingut de l'ArrayList amb el mètode contains

        // Verifiquem en quina posició de l'ArrayList està un element amb el mètode indexOf

       // Eliminem un element pel contingut i un altre per la posició amb el mètode remove

        // Canviem el color de una posició determinada amb el mètode set

      // Mostrar la llista utilitzant un for

     // Mostrar la llista utilitzant un for a l'estil foreach

     // Recorrer la llista utilitzant l'objecte Iterator

       }
}
```

Exemple 02

Utilitzem ArrayList i tots els seus mètodes apresos, però en aquest cas ho farem amb objectes.
Ens crearem la classe Gos que ja coneguem i alguns dels seus mètodes.
Farem ús dels mètodes d'ArrayList en el programa principal.
Finalment farem una ordenació de l'ArrayList. Aiçò ens portarà a definir un nou mètode en la classe Gos per a determinar per quin camp anem a ordenar.

CLASSE ENUMERADA

```
package UF09_Exemple02;

/**
 * UF09 Exemple 02: Classe enumerada Sexe
*/
public enum Sexe {
   FEMELLA, MASCLE
}
```

CLASSE GOS

```
package UF09_Exemple02;

/**
 * UF09 Exemple 02: Classe Gos
 */
public class Gos implements Comparable <Gos> {

   // Atributs de classe
   private static int totalPes=0, totalEdat=0;

   // Atributs d'instància
   private final String nom, color, raça;
   private final Sexe sexe;
   private int edat, pes;



   // Mètode constructor: sempre té el mateix nom que la classe
```

```
    public Gos (String nom, String color, String raça, Sexe sexe, int edat, int pes) {
        this.nom = nom;
        this.color = color;
        this.raça = raça;
        this.sexe = sexe;
        this.edat=edat;
        totalEdat++;
        this.pes=pes;
        totalPes++;
    }

    // Mètodes getter
    public String toString() {
        return this.nom + " és " + this.sexe + " de color " + this.color + " i raça " + this.raça + ",
pesa " + this.pes + " Kg i té " + this.edat + "anys.";
    }

    public String diMeNom (){
        return this.nom;
    }

    // Mètode per a poder fer l'ordenació d'ArrayList
    public int compareTo(Gos altreGos) {
        return (this.nom).compareTo(altreGos.diMeNom());
    }

}
```

PROGRAMA PRINCIPAL

```
package UF09_Exemple02;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Collections;
/**
 * UF09 Exemple 02: Creació i gestió d'un ArrayList d'objectes
 */
public class Exemple02 {

    public static void main (String[] args){
```

```
        // Creación de l'ArrayList i ús del mètode size per a vore el tamany
            ArrayList<Gos> gossos = new ArrayList<Gos>();
            System.out.println("Nombre de gossos: " + gossos.size());

        // Afegim els primers elements amb add i mostrem tamany amb size
            gossos.add(new Gos("Scooby", "marró", "dogo", Sexe.MASCLE, 5, 40));
            gossos.add(new Gos("Laika", "clapejat", "mestís", Sexe.FEMELLA, 3, 4));
            gossos.add(new Gos("Snoopy", "blanc", "beagle", Sexe.MASCLE, 6, 5));
            gossos.add(new Gos("Dama", "marró", "cocker", Sexe.FEMELLA, 4, 4));
            gossos.add(new Gos("Rodamón", "gris", "mestís", Sexe.MASCLE, 6, 6));
            System.out.println("Nombre de gossos: " + gossos.size());

        // Mostrar la llista de forma bàsica
            mostraGossos(gossos);

        // Afegim més elements i mostrem tamany
            gossos.add(new Gos("Pongo", "clapejat", "dalmata", Sexe.MASCLE, 7, 30));
            gossos.add(new Gos("Perdita", "clapejat", "dalmata", Sexe.FEMELLA, 6, 25));
            gossos.add(new Gos("Milú", "blanc", "fox terrier", Sexe.MASCLE, 8, 4));
            System.out.println("Número d'elements: " + gossos.size());

        // Mostrar la llista de forma bàsica
            mostraGossos(gossos);

       // Afegim un element en una posició determinada
            gossos.add(1, new Gos("Pluto", "ocre", "caça", Sexe.MASCLE, 8, 4));
            gossos.add(5, new Gos("Patán", "marró", "pointer", Sexe.MASCLE, 8, 4));
            System.out.println("Contingut de la lista després d'afegir dos gossos en les posicions
1 i 5: ");
            mostraGossos(gossos);

        // Ordenem la llista per nom tal i com ho haviem definit en la classe
            Collections.sort(gossos);
            System.out.println("Contingut de la llista després d'ordenar: ");
            mostraGossos(gossos);

        // Mostrem els elements que hi ha en una posició amb el mètode get
            System.out.println("El gos que hi ha en la posició 0 és " +
gossos.get(0).diMeNom());
            System.out.println("El gos que hi ha en la posició 3 és " + gossos.get(3));
```

```
        // Verifiquem si un element està en l'ArrayList i donem la posició
            Gos buscar = new Gos ("Milú", "blanc", "fox terrier", Sexe.MASCLE, 8, 4);
            if (gossos.contains(buscar)){
                System.out.println("El gos " + buscar.diMeNom() + " es troba en la posició " +
    ((int)gossos.indexOf(buscar)+1));
            }

        // Eliminem un element pel contingut el mètode remove
            gossos.remove(buscar);
            System.out.println("Contingut de la llista sense Milú: ");
            mostraGossos(gossos);


        // Canviem el color de una posició determinada amb el mètode set
            gossos.set(2, buscar);
            System.out.println("Contingut de la llista amb la posició 3 (index 2) actualitzada: ");
            mostraGossos(gossos);

      }

      public static void mostraGossos(ArrayList<Gos> gossos) {
        // Recorrer la llista utilitzant l'objecte Iterator
            System.out.print("Contingut de la lista utilitzant ITERATOR: ");
            Iterator<Gos> iter = gossos.iterator();      // Creem el Iterator a partir de la llista
            while(iter.hasNext()) {                      // Mentre hi haja següent en la llista
                System.out.print (iter.next().diMeNom() + " ");
            }
            System.out.println("");
        }
}
```

- Estudia el programa.
- Crea una nova versió utilitzant algun dels objectes vists fins ara (cotxe, participant, etc.). Intenta recordar les instruccions i fes proves fins que dones amb la sol·lució, evitant copiar-les.

Exemple 03

Creació d'una composició de classes.
A partir del programa de participació en una cursa anem a introduir informació més completa dels participants. Per a fer-ho utilitzarem la classe Persona que ja coneguem i incorporarem l'objecte persona en la classe Participant per a incloure totes les dades de la persona.

CLASSE PERSONA

```
package UF09_Exemple03;
import java.util.regex.Pattern;
/**
 * UF09 Exemple 03: Classe Persona
 */
public class Persona {

    private final String dni;
    private String nom;
    private String cognoms;
    private int edat;

    public Persona(String dni, String nom, String cognoms, int edat) {
        this.dni = dni;
        this.nom = nom;
        this.cognoms = cognoms;
        this.edat = edat;
    }

    public String getDni() {
        return dni;
    }

    public String getNom() {
        return nom;
    }

    public String getCognoms() {
        return cognoms;
    }
```

```
public int getEdat() {
    return edat;
}

public void setNom(String nom) {
    this.nom = nom;
}

public void setCognoms(String cognoms) {
    this.cognoms = cognoms;
}

public void setEdat(int edat) {
    this.edat = edat;
}

// Rertorna si és menor d'edat
public boolean esMenor() {
    return this.edat < 18;
}

// Retorna si està jubilat
public boolean esJubilat() {
    return this.edat >= 65;
}

//Devuelve la diferencia de edat entre este objeto y el recibido
public int diferenciaEdat(Persona p) {
    return this.edat - p.edat;
}

public void mostra() {
    System.out.println("DNI: " + this.dni);
    System.out.println("Nom: " + this.nom);
    System.out.println("Cognoms: " + this.cognoms);
    System.out.println("Edat: " + this.edat);
}
```

```
    // Funció que rep un DNI i retorna si és vàlid o no.
    public static boolean validarDNI(String valor) {
        // Expresió regular que indica 8 dígits i al final alguna de les lletres que se permeten
        // rangs A-H J-N P-T V-Z (recordeu, algunes no se permeten)
        String dniRegexp = "\\d{8}[A-HJ-NP-TV-Z]";
        // Retorna true si se compleix la expresió regular
        return Pattern.matches(dniRegexp, valor);
    }

}
```

CLASSE PARTICIPANT

```
package UF09_Exemple03;

/**
 * UF09 Exemple 03: Classe Participant
 */
public class Participant {

    private Persona persona;
    private final int dorsal;    //Quan es crea no es pot modificar
    private static int contadorDorsal=0; //atribut de classe compartit amb objectes

    public Participant (String dni, String nom, String cognoms, int edat){
        this.persona = new Persona (dni, nom, cognoms, edat);
        contadorDorsal++;
        this.dorsal=contadorDorsal;
    }

    public String dadesParticipant(){
  return "El participant " + this.persona.getNom() + " amb DNI " +
                this.persona.getDni() + " porta el dorsal " + this.dorsal;
    }

    public static String dadesParticipacio(){
  return "Hi ha un total de " + contadorDorsal + " participants";
    }
}
```

PROGRAMA PRINCIPAL

```
package UF09_Exemple03;

/**
* UF09 Exemple 03: Programa que crea participants
 */
public class ParticipantPrograma {

    public static void main (String[] args){

        Participant p1 = new Participant("1234567A", "Pere", "Sanxís", 24) ;
        Participant p2 = new Participant("9876543B", "Mireia", "Llopis", 32) ;

        System.out.println(p1.dadesParticipant());
        System.out.println(p2.dadesParticipant());
        System.out.println(Participant.dadesParticipacio());
    }
}
```

- Utilitza altres mètodes de la classe Persona per a mostrar més informació del participant.
- Inclou informació addicional com pot ser el Sexe de la persona (a més pots crear una classe enumerada per al Sexe).
- Fes ús de la validació del DNI al crear un nou participant.

Exemple 04

Creació d'una jerarquia de classes.
Anem a crear una jerarquia de classes per als animals.
Tindrem una superclasse Animals que contemplarà atributs i mètodes comuns a tota la fauna.
A continuació crearem una subclasse especialitzada per a les Aus, on afegirem alguns atributs addicionals d'aquestes. Finalment crearem dos subclasses específiques a l'Estruç i el Pingüí.
A partir d'ací ja podrem generar i gestionar objectes en el programa principal.
Per a alguns atributs utilitzarem les classes enumerades que ja conguem.

CLASSES ENUMERADES

```
package UF09_Exemple04;
/**
* UF09 Exemple 04: Herència, classe enumerada Tipus
*/
public enum Tipus {
   VERTEBRAT,INVERTEBRAT,DESCONEGUT
}
```

```
==========================================================
package UF09_Exemple04;
/**
* UF09 Exemple 03: Herència, classe enumerada Sexe
*/
public enum Sexe {
   MASCLE, FEMELLA, ASEXUAL
}
```

```
==========================================================
package UF09_Exemple04;
/**
* UF09 Exemple 04: Herència, classe enumerada Cobert
*/
public enum Cobert {
   PELS, PLOMES, ESCATES
}
```

SUPERCLASSE ANIMAL

```
package UF09_Exemple04;
/**
* UF09 Exemple 04: Herència, superclasse Animal
*/
public class Animal {

   // Atributs de classe
   private Tipus tipus;       // No el poden heretar les subclasses
   Sexe sexe;                 // L'hereten les subclasses

   // Constructor bàsic
   public Animal () {
      this.tipus = Tipus.DESCONEGUT;
   }

   // Constructor complet
   public Animal (Tipus tipus, Sexe sexe) {
      this.tipus=tipus;
      this.sexe=sexe;
   }

   public Tipus dimeTipus() {
      return tipus;
   }

   public String toString() {
      return "És un animal: " + this.tipus + " " + this.sexe;
   }

   // És un dels mètodes de la superclasse que heretaran totes les subclasses
   public void despaçament () {
      System.out.println ("Com sóc un animal puc desplaçar-me.");
   }

   // És un dels mètodes de la superclasse que no heretaran les subclasses
   private void metodesDesplaçament () {
      System.out.println ("Els animal es deplacen tant si tenen com si no tenen extremitats.");
   }
}
```

SUBCLASSE AU

```
package UF09_Exemple04;
/**
* UF09 Exemple 04: Herència, subclasse Au
*/
class Au extends Animal {

    // Atributs de classe
    Cobert cobert;

    // Constructor bàsic
    public Au(Tipus tipus, Sexe sexe) {
        super();
        this.sexe=sexe;
    }

    // Constructor complet
    public Au(Tipus tipus, Sexe sexe, Cobert cobert) {
        super(tipus, sexe);
        this.sexe=sexe;
        this.cobert=Cobert.PLOMES;
    }

    public String toString() {
        return "Sóc un au: " + this.dimeTipus() + " " + this.sexe + " i tinc el cos cobert de " +
this.cobert;
    }

   // Mètode per a volar
    public void volar() {
        System.out.println("Hi ha aus que volen i altres no.");
    }

    // Mètode per a les plomes
    public final void plomes() {
        System.out.println("Totes les aus es netegen les plomes.");
    }
}
```

SUBCLASSE ESTRUÇ

```
package UF09_Exemple04;
/**
* UF09 Exemple 04: Herència, subclasse Au
*/
public class Estruç extends Au {
    // Atributs de classe
    String especie;
    double km_hora;

    // Constructor bàsic
    public Estruç(Tipus tipus, Sexe sexe) {
        super(tipus, sexe);
    }

    // Constructor complet
    public Estruç(Tipus tipus, Sexe sexe, Cobert cobert, String especie, double km_hora) {
        super(tipus, sexe, cobert);
        this.especie=especie;
        this.km_hora=km_hora;
    }

    // L'atribut Tipus és privat i no l'heretem, necessitem un mètode getter
    public String toString() {
        return "Sóc " + this.especie + ": " + this.dimeTipus() + " " + this.sexe +
", tinc el cos cobert de " + this.cobert + " i alcance una velocitat de " + this.km_hora + "
Km/h.";       }

    // L'estruç no pot volar
    @Override
    public void volar() {
        System.out.println("Els estruços no podem volar però a correr no ens guanya ningú.");
    }

    // Mètode per a les plomes (El mètode és final en la superclasse, no es pot sobreescriure)
    // @Override
    // public void plomes() {
    //    System.out.println("Els estruços es netegen les plomes.");
    //}
}
```

SUBCLASSE PINGÜÍ

```
package UF09_Exemple04;
/**
* UF09 Exemple 04: Herència, subclasse Au
*/
public class Pingui extends Au {
    // Atributs de classe
    String especie;
    double nusos;

    // Constructor bàsic
    public Pingui(Tipus tipus, Sexe sexe) {
        super(tipus, sexe);       }

    // Constructor complet
    public Pingui(Tipus tipus, Sexe sexe, Cobert cobert, String especie, double nusos) {
        super(tipus, sexe, cobert);
        this.especie=especie;
        this.nusos=nusos;
    }

    // L'atribut Tipus és privat i no l'heretem, necessitem un mètode getter
    public String toString() {
        return "Sóc un " + this.especie + ": " + this.dimeTipus() + " " + this.sexe + ", tinc el cos
cobert de " + this.cobert +
                " i alcance una velocitat de " + this.nusos + " nusos.";
    }

    // El pingüí no pot volar
    @Override
    public void volar() {
        System.out.println("Els pingüins no podem volar però som la canya nedant.");
    }

    // Mètode per a les plomes (El mètode és final en la superclasse, no es pot sobreescriure)
    // @Override
    // public void plomes() {
    //    System.out.println("Els pingüins es netegen les plomes.");
    //}
}
```

PROGRAMA PRINCIPAL

```
package UF09_Exemple04;

/**
* UF09 Exemple 04: Programa principal que mostra com funciona l'herència
*/
public class ProgramaHerencia {
	public static void main(String[] args) {

// PINGÜINS
// Utilitzem constructors: bàsic Pingui => bàsic Au (no passa tipus) => bàsic Animal. Per això falten dades
		Pingui pingui1=new Pingui(Tipus.VERTEBRAT, Sexe.FEMELLA);
		System.out.println ("Motrem el pingüí 1: " + pingui1);
		pingui1.volar();		// Agafa el mètpode volar() de Pingui
		pingui1.plomes();		// Hem heretat el mètode plomes() d'Au perquè és public
		pingui1.despaçament();	// Hem heretat el mètode desplaçamane() d'Animal perquè és public
		// pingui1.metodesDesplaçament();  // Donarà error perquè aquest mètode no és públic

// Utilitzem constructors: complet Pingui => complet Au (passa tipus)=> complet Animal. Per això està complet
		Pingui pingui2=new Pingui(Tipus.VERTEBRAT, Sexe.MASCLE, Cobert.PLOMES, "Pingüí Emperador", 1.35);
		System.out.println ("Motrem el pingüí 2: " + pingui2);
		pingui2.volar();		// Agafa el mètpode volar() de Pingui

// ESTRUÇOS
// Utilitzem constructors: complet Estruç => complet Au (passa tipus) => complet Animal. Per això està complet
		Estruç estruç1 = new Estruç (Tipus.VERTEBRAT,Sexe.FEMELLA, Cobert.PLOMES, "Estruç Somalí", 31);
		System.out.println ("Motrem l'estruç 1: " + estruç1);
		estruç1.volar();		// Agafa el mètpode volar() d'Estruç

// AUS
// Utilitzem constructors: bàsic Au (no passa el tipus) => bàsic Animal. Per això falten dades en l'objecte
		Au augenerica1 = new Au (Tipus.VERTEBRAT,Sexe.MASCLE);
		System.out.println ("Motrem l'au genèrica 1: " + augenerica1);
		augenerica1.volar();		// Agafa el mètpode volar() d'Au

// Utilitzem constructors: complet Au  (passa el tipus) => complet Animal. Per això està completa la informació
		Au augenerica2= new Au (Tipus.VERTEBRAT, Sexe.MASCLE, Cobert.PLOMES);
		System.out.println ("Motrem l'au genèrica 2: " + augenerica2);

// ANIMALS
// Utilitzem constructors: complet Animal. Per això està completa la informació
		Animal animalgeneric1=new Animal (Tipus.VERTEBRAT, Sexe.MASCLE);
		System.out.println ("Motrem l'animal genèric 1: " + animalgeneric1);
	}
}
```

PROGRAMA PRINCIPAL 2

Exemple de com treballar amb una llista amb diverses subclasses que pertanyen a una mateixa superclasse.

```
package UF09_Exemple04;
import java.util.ArrayList;
/**
* UF09 Exemple 04: Programa principal que mostra com gestionar un ArrayList amb diverses subclasse que
* pertanyen a  una mateixa superclasse.
*/
public class ProgramaLlista {

    public static void main (String[] args){

        ArrayList<Animal> animals = new ArrayList<Animal>();

        animals.add (new Pingui(Tipus.VERTEBRAT, Sexe.FEMELLA, Cobert.PLOMES, "Pingüí Emperador", 1.35));
        animals.add (new Estruç (Tipus.VERTEBRAT,Sexe.FEMELLA, Cobert.PLOMES, "Estruç Somalí", 31));
        animals.add (new Pingui(Tipus.VERTEBRAT, Sexe.MASCLE, Cobert.PLOMES, "Pingüí Rei", 1.12));
        animals.add (new Estruç (Tipus.VERTEBRAT,Sexe.MASCLE, Cobert.PLOMES, "Estruç Massai", 25));

        String text;
        for (int i = 0; i < animals.size(); i++) {
           Animal a = animals.get(i);
           if (a instanceof Estruç) {
              text="Informació d'un Estruç";
           } else { if (a instanceof Pingui) {
                     text="Informació d'un Pingüí";
                } else { text="Informació desconeguda"; }
           }
           System.out.println(text + "=> " + a);
        }
    }
}
```

- Com pots veure hi ha sobrecàrrega de constructors. Revisa quins constructors s'utilitzen al crear un objecte i fer ús de les jerarquies per entendre com funcionen. Amb aquest aspecte s'ha d'anar molt en compte ja que és una font molt usual d'errors.
- Tracta de descomentar l'últim mètode que hi ha en les classes Estruç i Pingui. Què passa al fer-ho? Perquè?
- Intenta utilitzar des del ProgramaHerencia el mètode metodesDesplaçament () de la classe Animal per a mostrar per consola el resultat. Què passa al fer-ho?
- Intenta accedir directament a l'atribut "tipus" de qualsevol objecte per a mostrar-lo per consola. Què passa al fer-ho? Perquè?
- Fes el mateix amb l'atribut "sexe". A què es deu aquesta diferència?
- Crea una jerarquia nova amb mamífers totes les característiques que hem revisat.
- Tracta d'incloure altres objectes en l'ArrayList del Programa Principal 2 (ProgramaLlista). Poden ser noves subclasses o qualsevol de les classes de la mateixa jerarquia.