GRAU D'ENGINYERIA INFORMÀTICA

PROGRAMACIÓ II

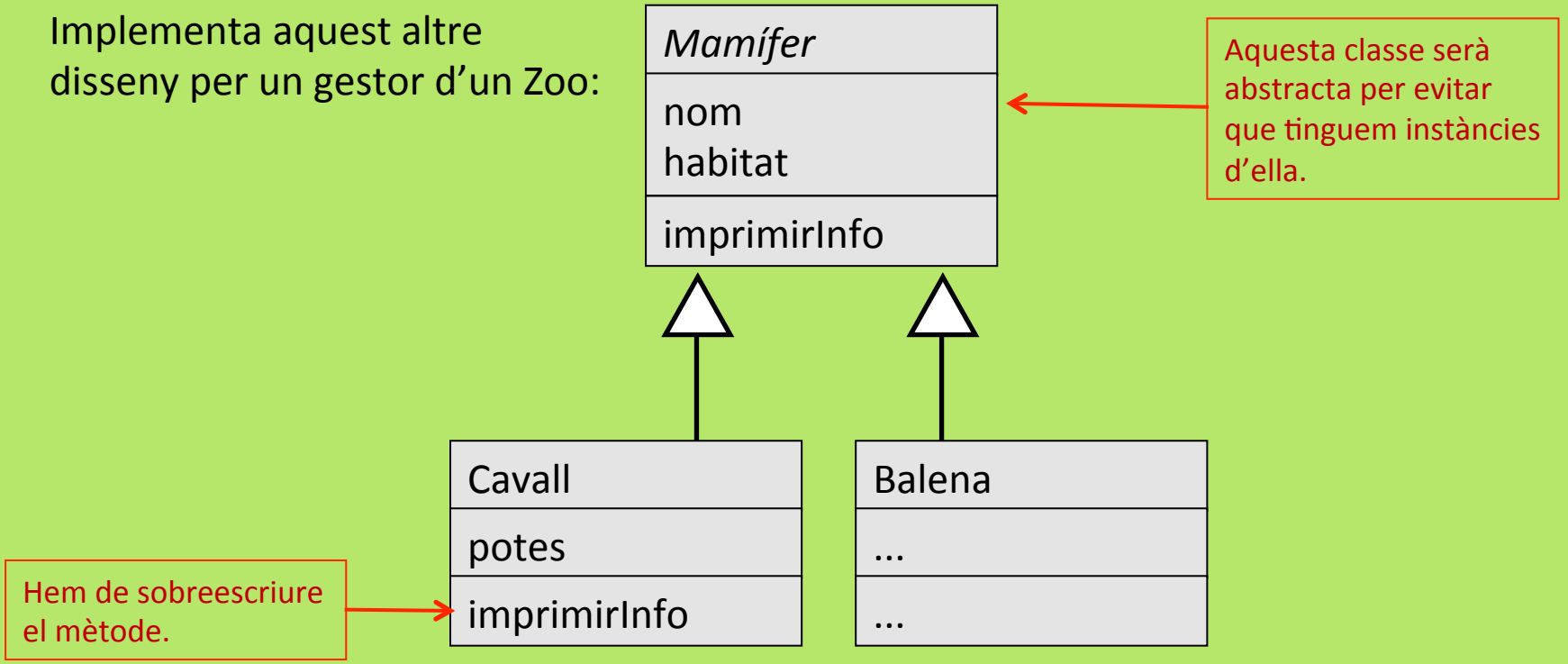
Programació Orientada a Objectes (exercicis herència)

Sergio Sayago (basat en material de Laura Igual)
Departament de Matemàtiques i Informàtica
Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Herència i jerarquia de classes

Exercici Sobreescriptura

Implementa aquest altre disseny per un gestor d'un Zoo:



Mamífer serà una classe abstracta, ja que hi ha informació d'aquesta classe que no es pot conèixer sense especificar més (saber més sobre l'animal).
Exemple: l'habitat de l'animal: terrestre o marí.

Mamifer.java

```
public abstract class Mamifer {
    private String nom, habitat;
    public Mamifer(String nom) {
          this.nom = nom;
          this.habitat = "Desconegut"
    }
    public void imprimirInfo() {
          System.out.println("Nom: " + nom + ", habitat " + habitat");
   }
}
```

Cavall.java

```
public class Cavall extends Mamifer {
    private int potes;
    public Cavall(String nom){
          super(nom);
          potes=4;
          setHabitat("Terrestre");
    }
    public void imprimirInfo() {
          super.imprimirInfo();
          System.out.println("Té " + potes + " potes");
    }
}
```

Balena.java

```
public class Balena extends Mamifer {
    public Balena(String nom){
          super(nom);
          setHabitat("Maritim");
    }
}
```

Exercici

Què surt per pantalla?

CreaExemple.java

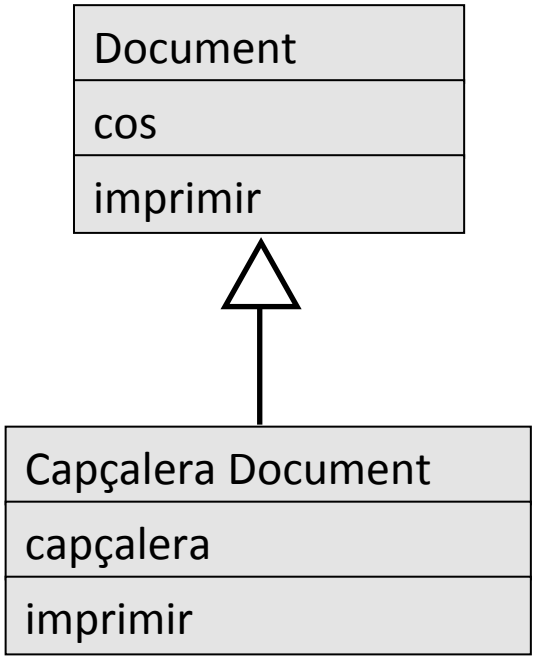
```
public class CreaExemples {
   public static void main(String [] args) {
        Cavall bobi = new Cavall("Bobi");
        bobi.imprimirInfo();
        Balena bal= new Balena("Nemo");
        bal.imprimirInfo(); /*Està a la classe Mamífer*/
   }
}
```

→ Nom Bobi, habitat Terrestre
Té 4 potes.
Nom Nemo, habitat Maritim

MES EXEMPLES

Exemple

- Herència amb sobreescriptura de mètodes:

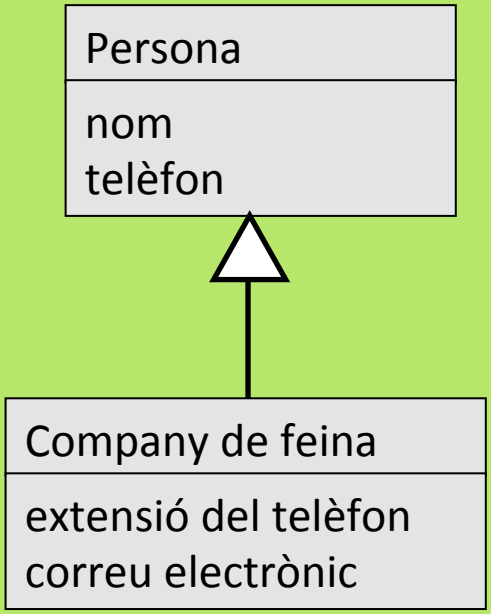


Entenem DocumentCapçalera com un tipus específic de document que té a més d'un cos de document una capçalera.

Les funcions a realitzar pel mètode imprimir ara seran diferents, ja que tenim una informació diferent emmagatzemada.

Exercici

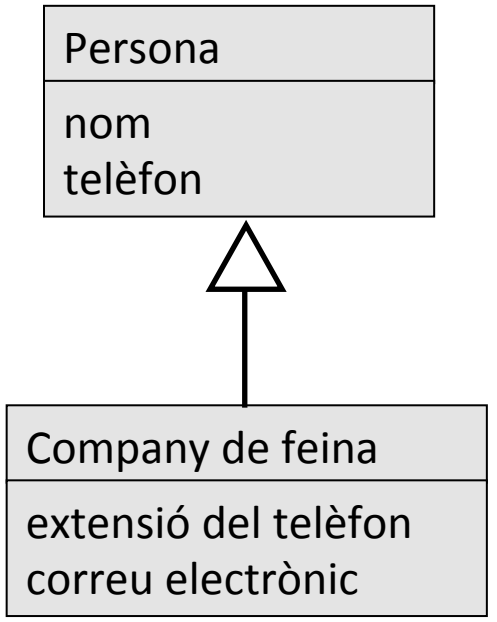
- Implementar el següent disseny:



Solució:

Persona.java

```
public class Persona {
    private String nom;
    private String telefon;
    // constructor
    public Persona (String pNom, String pTelefon) {
       nom = pNom;
       telefon = pTelefon;
    }
    // Getters i setters
    public String getNom() {
       return nom;
    }
    public String getTelefon() {
       return telefon;
    }
    public void setNom(String pNom)  {
       nom = pNom;
    }
    public void setTelefon(String pTelefon) {
       telefon = pTelefon; }
}
```



Company.java

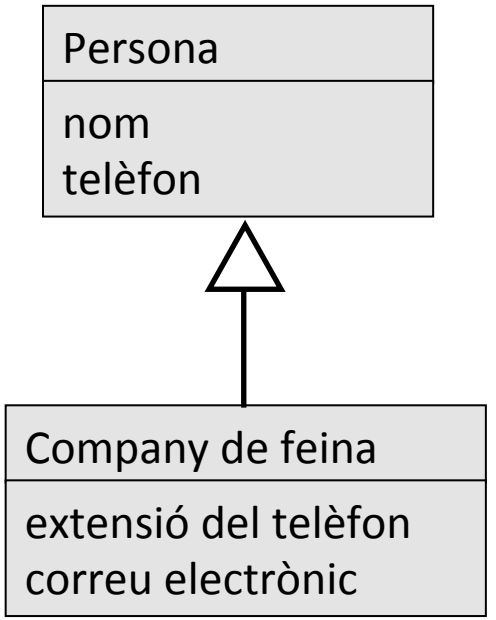
```
public class Company extends Persona {
    private String extTel;
    private String email;
    // constructors:
    public Company(String pNom, String pTelefon) {
       super(pNom, pTelefon);
       extTel = "";
       email = "";}
    public Company(String pNom, String pTelefon, String pExtTel, String pEmail) {
       super(pNom, pTelefon);
       extTel = pExtTel;
       email = pEmail;}

    // Getters i setters
    public String getExtTel() {
       return extTel;}

    public String getEmail() {
       return email;}

    public void setExtTel(String pExtTel) {
       extTel = pExtTel;}

    public void setEmail(String pEmail) {
       email = pEmail;}
}
```



Exercici: Fes de compilador!

Donat el codi anterior de les classe Persona i Company indicar si hi ha errors de compilació en les següents classes del mateix paquet:

1. Classe TestCompanys1

```
public class TestCompanys1 {
   public static void main(String[] args){
        Company nouCompany = new Company();
    }
}
```

2. Classe TestCompanys2

```
public class TestCompanys2 {
    public static void main(String[] args){
        String nom="Joan";
        String telefon="93111111";
        String telefonActual;
        Company nouCompany = new Company(nom, telefon);
        System.out.println(nouCompany.getNom());
        telefonActual = "93222222";
        nouCompany.setTelefon(telefonActual);
    }
}
```

Donarà error?

Donarà error?

Solució Exercici:

Donat el codi de les classe Persona i Company indicar si hi ha errors de compilació en les següents classes del mateix paquet:

1. Classe TestCompanys1

```
public class TestCompanys1 {
   public static void main(String[] args){
        Company nouCompany = new Company();
    }
}
```

Error de compilació:
La classe Company no té constructor sense paràmetres.

2. Classe TestCompanys2

```
public class TestCompanys2 {
    public static void main(String[] args){
        String nom="Joan";
        String telefon="93111111";
        String telefonActual;
        Company nouCompany = new Company(nom, telefon);
        System.out.println(nouCompany.getNom());
        telefonActual = "93222222";
        nouCompany.setTelefon(telefonActual);
    }
}
```

Donarà error?

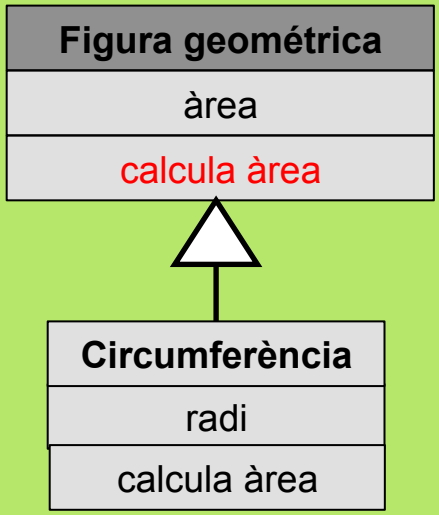
Donarà error?

No. Encara que la classe Company no té els mètodes getNom i setTelefon implementats, la superclasse Persona si que els té.

Exercici

Amplia la implementació de la classe **Cercle** que hereta de la classe abstracta Figura amb

- Un contador de cercles,
- Dos mètodes propis,
  - Un mètode d'objecte per comparar cercles i
  - Un mètode de classe per comparar cercles.



Cercle.java

```
public class Cercle extends Figura {
    static int numCercles = 0;
    public static final double PI=3.14159265358979323846;
    public double radi;

    // constructors
    public Cercle(double x, double y, double radi) {
        this.x=x; this.y=y; this.radi =radi;
        numCercles++;}
    public Cercle(double radi) { this(0.0, 0.0, radi); }
    public Cercle(Cercle c) { this(c.x, c.y, c.radi); }
    public Cercle() { this(0.0, 0.0, 1.0); }

    // calcula l'area del cercle
    public double calculaArea() {
        area = PI * radi * radi;
        return area;
    }
    // calcula el valor del perímetre
    public double calculaPerimetre(){
        perimetre = 2 * PI * radi;
        return perimetre;
    }

    // mètode d'objecte per a comparar cercles
    public Cercle elMajor(Cercle c) {
        if (this.radi>=c.radi)
            return this;
        else return c;
    }

    // mètode de classe per a comparar cercles
    public static Cercle elMajor(Cercle c, Cercle d) {
        if (c.radi>=d.radi)
            return c;
        else return d;
    }

} // fi de la classe Cercle
```

TestCercles.java

```
public class TestCercles {
        public static void main(String[] args){
                Cercle cercleGran;
                System.out.println("número de cercles = " + Cercle.numCercles);
                Cercle cercle1 = new Cercle(1.5);
                System.out.println("número de cercles = " + Cercle.numCercles);
                Cercle cercle2 = new Cercle(2.5);
                System.out.println("número de cercles = " + Cercle.numCercles);
                Cercle cercle3 = new Cercle(3.5);
                System.out.println("número de cercles = " + Cercle.numCercles);

                cercleGran = Cercle.elMajor(cercle1, cercle2);
                System.out.println("El radi del cercle gran és = " + cercleGran.getRadi());

                cercleGran = cercle3.elMajor(cercle2);
                System.out.println("El radi del cercle gran és = " + cercleGran.getRadi());

        }
} // fi de la classe
```

Sortida per pantalla

```
número de cercles = 0
número de cercles = 1
número de cercles = 2
número de cercles = 3
El radi del cercle gran és = 2.5
El radi del cercle gran és = 3.5
```

```
public class Cercle extends Figura {
   // quantitat d'objectes d'aquesta classe que existeixen.
   static int numCercles = 0;
   // constant PI
   private static final double PI=3.14159265358979323846;
   // radi del cercle
   private double radi;

   // constructors
   public Cercle(double x, double y, double radi) {
         this.x=x; this.y=y; this.radi =radi;
         // actualitza la quantitat d'objectes d'aquesta classe
         sumarCercle();
   }
   public Cercle(double radi) { this(0.0, 0.0, radi); }
   public Cercle(Cercle c) { this(c.x, c.y, c.radi); }
   public Cercle() { this(0.0, 0.0, 1.0); }
    // calcula l'area del cercle
    public double calculaArea() {
         area = (double) (PI * radi * radi);
         return area;  }
      // calcula el valor del perímetre
      public double calculaPerimetre(){
         perimetre = (double) (2 * PI * radi);
         return perimetre;
      }
   // mètode d'objecte per a comparar cercles
   public Cercle elMajor(Cercle c) {
         if (this.radi>=c.radi) return this;
         else return c;    }
```

```
   // mètode de classe per a comparar cercles
   public static Cercle elMajor(Cercle c, Cercle d) {
   if (c.radi>=d.radi) return c; else return d;
   }
   public double getRadi(){
      return this.radi;
   }
   // destructor
   protected void finalize() {
      // actualitza la quantitat d'objectes d'aquesta classe que existeixen:
      restarCercle();
   }

   // mètode de classe que incrementa en un la quantitat d'objectes d'aquesta
classe que existeixen.
   private static void sumarCercle(){
         numCercles++;
   }

   // mètode de classe que decrementa la quantitat d'objectes creats dins
d'aquesta classe.
   private static void restarCercle(){
      numCercles--;
   }

} // fi de la classe Cercle
```

Una altra possible implementació de la classe Cercle.