

GRAU D'ENGINYERIA INFORMÀTICA

PROGRAMACIÓ II

Bloc 2:

Programació Orientada a Objectes (3)

Laura Igual

Departament de Matemàtiques i Informàtica

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

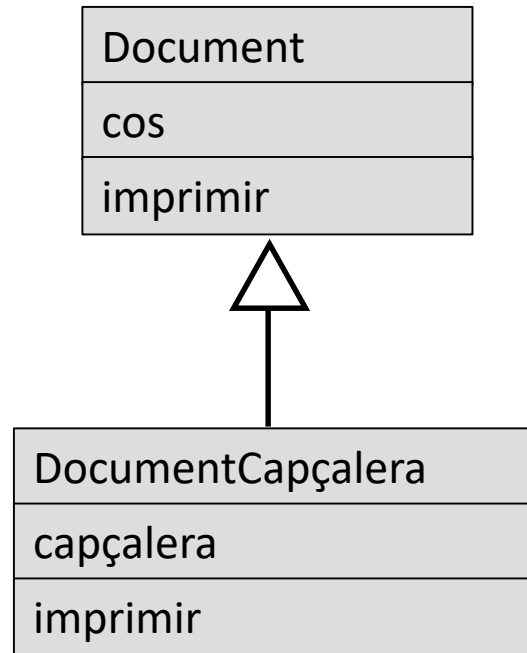


UNIVERSITAT DE
BARCELONA

MES EXEMPLES

Exemple

- Herència amb sobreescritura de mètodes:

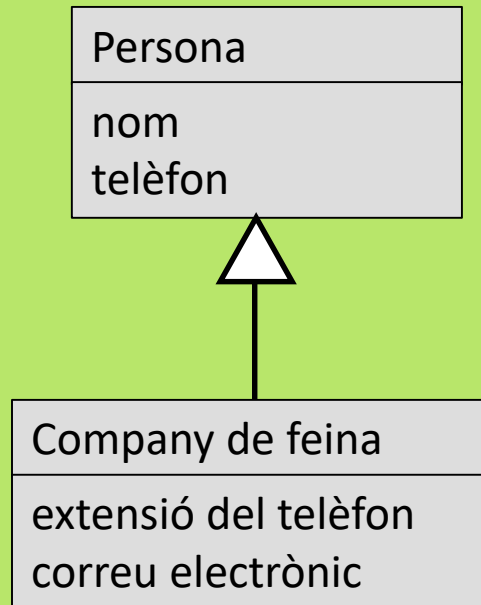


Entenem **DocumentCapçalera** com un tipus específic de document que té a més d'un cos de document una capçalera.

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

Exercici

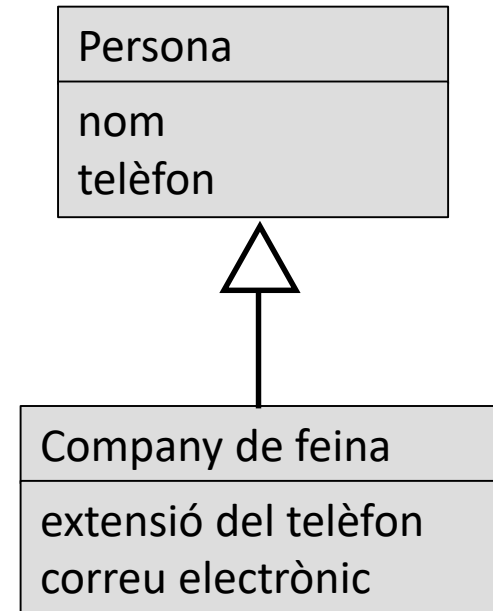
- Implementar el següent disseny:



Solució:

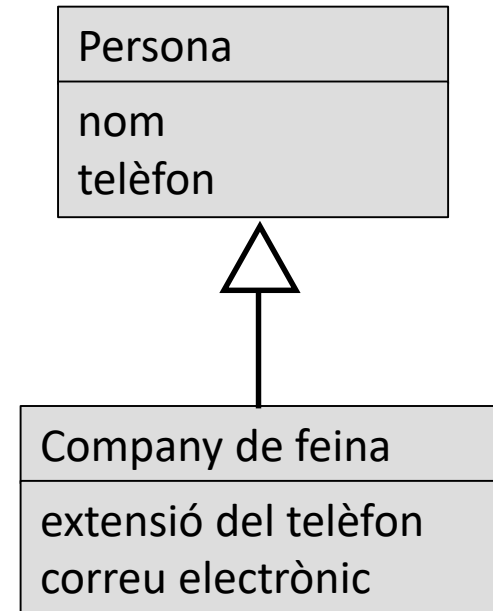
```
public class Persona {  
    private String nom;  
    private String telefon;  
    // constructor  
    public Persona (String pNom, String pTelefon) {  
        nom = pNom;  
        telefon = pTelefon;  
    }  
    // Getters i setters  
    public String getNom() {  
        return nom;  
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

Persona.java



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="931111111";  
        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="931111111";  
        String telefonActual;  
        Company nouCompany = new Company(nom, telefon);  
        System.out.println(nouCompany.getNom());  
        telefonActual = "93222222";  
        nouCompany.setTelefon(telefonActual);  
    }  
}
```

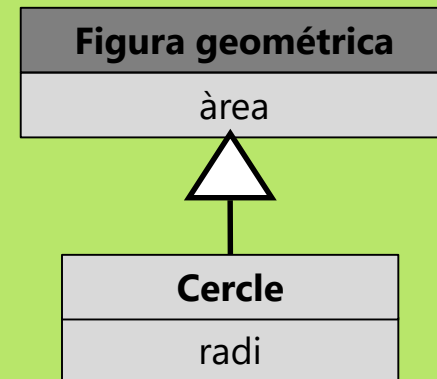
No. Encara que la classe
Company no té els
mètodes getNom i
setTelefon implementats,
la superclasse Persona si
que els té.

Exercici de repàs

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

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

Els dos mètodes han de comparar el radi dels dos objectes **Cercle** i retornar l'objecte que tingui el radi més gran.



```
public class Cercle extends Figura {
```

```
    static int numCercles = 0;
```

```
    public static final double PI=3.14159265358979323846;
```

```
    public double radi;
```

```
    // constructors
```

```
    public Cercle(double radi) {
```

```
        this.radi =radi;
```

```
        numCercles++;}
```

```
    public Cercle(double radi) { this(radi); }
```

```
    public Cercle(Cercle c) { this(c.radi); }
```

```
    public Cercle() { this(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;
```

```
    }
```

```
    // mètodo d'objecte per a comparar cercles
```

```
    public Cercle elMajor(Cercle c) {
```

```
        if (this.radi>=c.radi)
```

```
            return this;
```

```
        else return c;
```

```
    }
```

```
    // mètodo 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
```

```
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 radi) {
        this.radi=radi;
        // actualitza la quantitat d'objectes d'aquesta classe
        sumarCercle();
    }
    public Cercle(double radi) { this(radi); }
    public Cercle(Cercle c) { this(c.radi); }
    public Cercle() { this(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

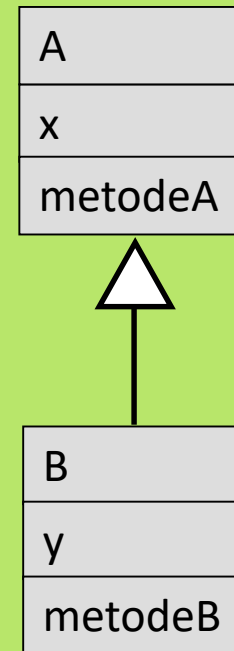
```

Una altra possible implementació de la classe Cercle.

Exercici

- Donat el següent codi de la classe A i la classe B (que hereta de la classe A) i el diagrama il·lustrant la relació entre les classes

```
public class A{  
    protected int x;  
    public void metodeA() {  
        ....  
    }  
}  
  
public class B extends A{  
    private int y;  
    public void metodeB() {  
        ....  
    }  
}
```



Exercici

Preguntes sobre el codi de la transparència següent.

1. Especifica si hi ha alguna **conversió** de tipus **implícita** en el codi i en cas afirmatiu en quines línies.

2. Si afegim un nou mètode a la classe A anomenat imprimir que imprimeix el missatge “Missatge d’A”, però no el sobreescrui a la classe B, que passa quan fem una crida d’aquesta forma:

```
var2.imprimir();
```

3. Indica com has de sobreescrui el mètode imprimir a la classe B de manera que quan fas la crida

```
var2.imprimir();
```

La sortida sigui: “Missatge de B”

4. Ara, indica com has de sobreescrui el mètode imprimir a la classe B de manera que quan fas la crida

```
var2.imprimir();
```

La sortida sigui: “Missatge d’A”

“Missatge de B”

Exercici

```
0 public static void main(String[] args) {  
1     A var1 = new B();  
2     B var2 = new B();  
3     A var3;  
4     var3 = var2;  
5     int j = var2.x;  
6     int i = var1.x;  
7     int k = var1.y;  
8     var1.metodeA();  
9     var1.metodeB();  
10    var2.metodeA();  
11    var2.metodeB();  
12 }
```

Exercici: solució

1. Especifica si hi ha alguna **conversió** de tipus **implícita** en el codi anterior i en cas afirmatiu en quines línies.

Solució:

Sí, a la línia 1 i 4

2. Si afegim un nou mètode a la classe A anomenat imprimir que imprimeix el missatge “Missatge d’A”, però no el sobreescriu a la classe B, que passa quan fem una crida d’aquesta forma:

```
var2.imprimir();
```

Solució:

Apareixerà el missatge: “Missatge d’A”

Exercici: solució

3. Indica com has de sobreescrivre el mètode imprimir a la classe B de manera que quan fas la crida

```
var2.imprimir();
```

La sortida sigui: "Missatge de B"

Solució: En la classe B sobreescrivim:

```
public void imprimir(){  
    System.out.println("Missatge de B");  
}
```

4. Ara, indica com has de sobreescrivre el mètode imprimir a la classe B de manera que quan fas la crida

```
var2.imprimir();
```

La sortida sigui: "Missatge d'A"

"Missatge de B"

Solució:

```
public void imprimir(){  
    super.imprimir();  
    System.out.println("Missatge de B");  
}
```