

GRAU D'ENGINYERIA INFORMÀTICA

PROGRAMACIÓ II

Problema 2:

Aspectes bàsics de la Programació Orientada a Objectes (en JAVA)

Objectius

- **Dels Problemes:** Realitzar exercicis pràctics per aprofundir la teoria i ajudar-vos en la part pràctica de l'assignatura
- **Del Problemes 2:** Conèixer aspectes bàsics i importants del paradigma d'orientació a objectes i la seva programació en JAVA

Continguts

- **Creació de classes i objectes**
- Pas de paràmetres primitius i de referència
- Mètodes de classe i d'objecte
- Inicialització de variables
- Encapsulació

Creació de classes

Estat i comportament

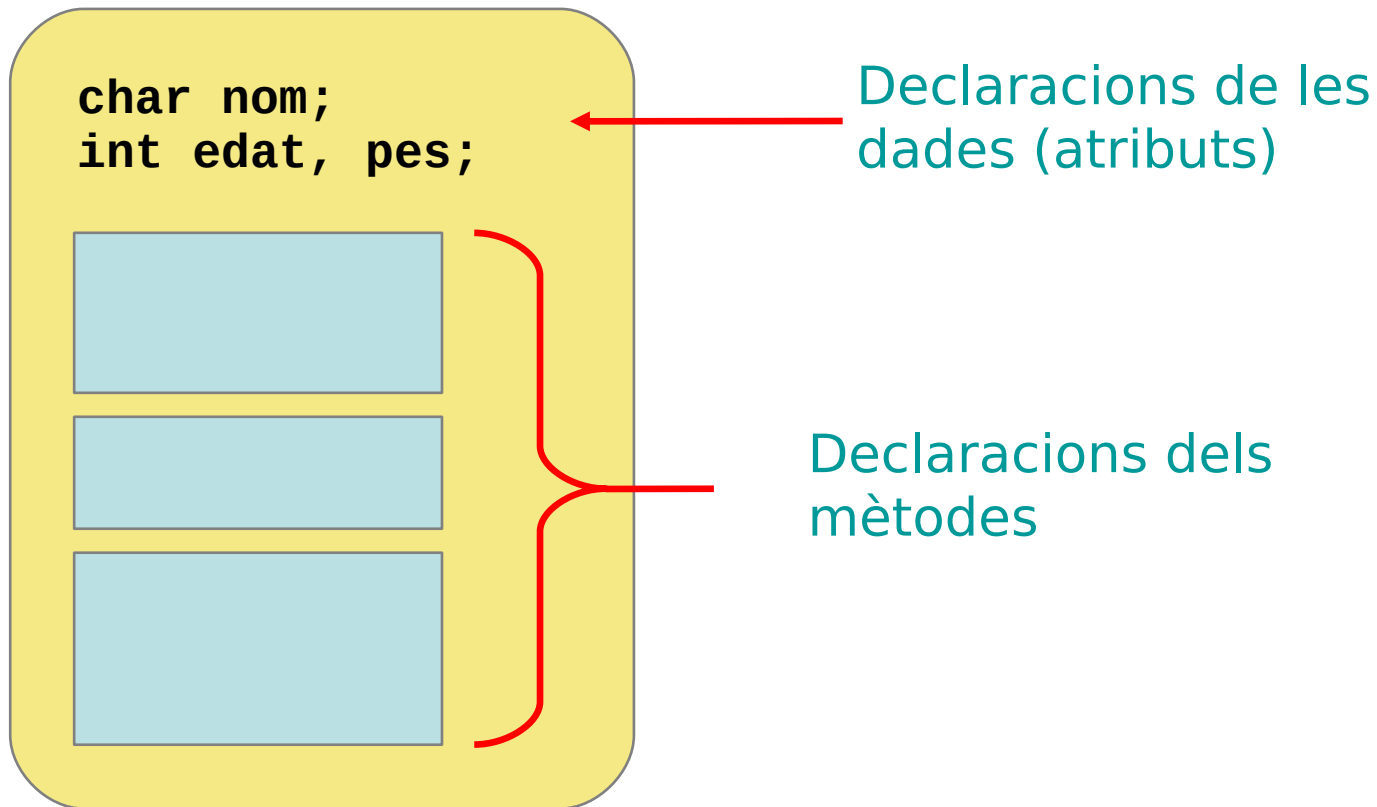
- Un objecte té estat i comportament
- Per exemple, la classe Persona
 - El seu estat està definit per Nom, Edat i Pes.
 - El seu comportament principal és CanviaEdat, ConsultaNom i ConsultaEdat

Persona
nom edat pes
canviEdat consultaNom consultaEdat

Creació de classes

Estructura

Classe Persona



Creació de classes

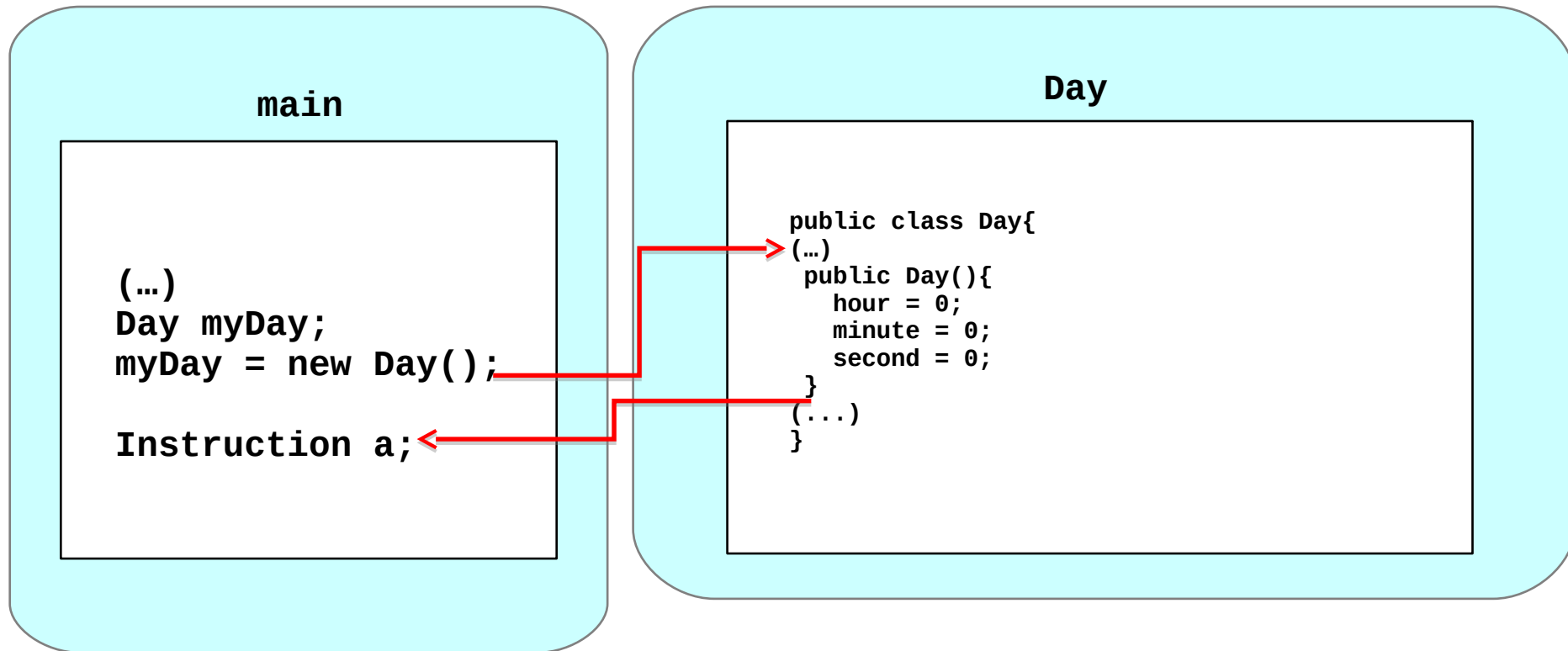
Construcció d'objectes

```
public class Day {  
    //attributes  
    (...)  
    //constructor without parameters  
    public Day() {  
        (...)  
    }  
    //constructor with parameters  
    public Day (int h, int m, int s){  
        (...)  
    }  
    (...)  
}  
  
public class Ex1{  
    public static void main (String [] args){  
        Day myDay = new Day();  
        Day anotherDay = new Day (12,30,0);  
    }  
    (...)  
}
```

- Construcció d'objectes: amb instrucció **new**
- Instrucció **new** invoca a un constructor de la classe
- **Constructor** = mètode amb mateix nom que la classe
- **Sobrecàrrega** de constructors: mateix nom, diferents arguments
- Un constructor pot cridar a un altre constructor (amb **this**)

Creació de classes

Construcció d'objectes i flux



Creació de classes

Destructor

- Un destructor és un mètode que realitza les **tasques prèvies a l'eliminació** de l'objecte
- Una classe pot definir un mètode destructor quan, a més d'alliberar la memòria ocupada per l'objecte que s'elimina, **sigui necessari especificar l'execució d'alguna operació**

Creació de classes

Destructor

```
public class MiClase {  
    int i;  
    public MiClase() {  
        i = 10;  
    }  
    ...  
  
    public void finalize() {  
        // Fer alguna tasca per eliminar  
        System.out.println("Destructor de la classe  
MiClasse");  
    }  
  
}
```

Exercici 1

Activitat 1: Creeu la següent classe:

- Nom: Day
- Atributs:
 - int: hour
 - int: minute
 - int: second
- Mètodes:
 - void: setHour(int h)
 - int: getHour()
 - void: setMinute(int m)
 - int: getMinute()
 - void: setSecond(int s)
 - int: getSecond()

Exercici 1

Activitat 2: Segons la classe Day, i aquest main:

```
public class Ex1 {  
    public static void main(String[] args) {  
        // Bloc 1  
        Day myDay;  
        System.out.println(myDay.getHour());  
        //Bloc 2  
        Day myDay = null;  
        System.out.println(myDay.getHour());  
        //Bloc 3  
        Day myDay = new Day(11,30,0);  
        System.out.println(myDay.getHour());  
    }  
}
```

- Què fa la instrucció del bloc 1 `Day myDay` ?
- Perquè el Bloc 1 ens dona l'error "*myDay has not been initialised*"?
- El Bloc 2, compila, però s'executa correctament?

Day day = new Day();

The screenshot shows a Java IDE with a code editor and a variable watch window. The code editor displays the following code:

```
24 System.out.println(myDay.getHour());*/
25
26 Day myDay = new Day();
27 System.out.println(myDay.getHour());
28 }
29
30 }
31
```

The variable watch window is open, showing the following variables:

Name	Type	Value
<Enter new watch>		
Static		
args	String[]	#56(length=0)
myDay	Day	#59

Exercici 1

Activitat 3: Analitzeu el següent codi i responeu a les preguntes:

```
public class Ex1{  
    public static void main(String[] args) {  
        //Bloc 1  
        Day[] week = new Day[7];  
        week[0].setHour(12);  
        //Bloc 2  
        Day[] week = new Day[7];  
        week[0] = new Day(12,30,0);  
        System.out.println(week[0].getHour());  
    }  
}
```

- Què fa la instrucció `Day[] week = new Day[7]` ?
- Quin bloc és el correcte, i perquè?

Array d'objectes creat però vuit

```
30 //Error: java.lang.NullPointerException
    Day[] semana = new Day[7];
    semana[0].setHour(12);
33 }
34
35 }
36
```

ex1.Ex1 > main >

Output Variables Breakpoints

Name	Type	Value
semana	Day[]	#59(length=7)
[0]		null
[1]		null
[2]		null
[3]		null
[4]		null

Continguts

- Creació de classes i objectes
- **Pas de paràmetres primitius i de referència**
- Mètodes de classe i d'objecte
- Inicialització de variables
- Encapsulació

Pas de paràmetres

Pas per valor = còpia dels valors

Pas per referència = direccions / “punters”

Java passa els objectes per referència

Descarrega't el projecte Netbeans
ValorOReferencia.java del campus, executa'l i
analitza el programa

Pas de paràmetres

Paràmetres

Quan es crida a un mètode. Els paràmetres actuals es **copien** en els paràmetres formals

ch = obj.calc (25, count, "Hello");

arguments

Passa per valor
o passa per copia

char calc (int num1, int num2, String message)

paràmetres

```
{  
    int sum = num1 + num2;  
    char result = message.charAt (sum);  
    return result;  
}
```

Example


```
public class ExempleOperacionsChars {  
  
    char extractChar(int num1, int num2, String message){  
        int sum = num1 + num2;  
        char result = message.charAt(sum);  
        return result;  
    }  
  
    public static void main(String[] args){  
        ExempleOperacionsChars exemple= new ExempleOperacionsChars();  
        int count = 2;  
        char ch = exemple.extractChar(1, count, "Hello");  
        System.out.println(ch);  
    }  
}
```

← Torna el caràcter de la posició donada.

Exercici 2: Passo parametres

```
public class Llibre {  
    private String titol;  
    private String autor;  
  
    public Llibre(String titol){  
        this.titol = titol;  
        this.autor = "";  
    }  
    public Llibre(String titol, String autor){  
        this.titol = titol;  
        this.autor = autor;  
    }  
    public void setAutor(String autor){  
        this.autor = autor;  
    }  
    public String getAutor(){  
        return this.autor;  
    }  
}
```

Copia l'autor d'un altre llibre.



```
public void copiaAutor(Llibre llibre){  
    this.setAutor(llibre.getAutor());  
}  
public void copiaModificaAutor(Llibre llibre){  
    this.setAutor(llibre.getAutor());  
    llibre.setAutor("");  
}
```



Copia l'autor d'un altre llibre passat per paràmetre i modifica aquest.

Exercici 2

```
public class ExemplePassoParam {  
    public static void main(String[] args){  
        ExemplePassoParam exemple= new ExemplePassoParam();  
        exemple.passoParametre();  
    }  
    public void passoParametre(){  
        Llibre llibre1 = new Llibre("Europa", "Kath");  
        Llibre llibre2 = new Llibre("Europa", "Bob");  
        System.out.println("1.");  
        System.out.println("Autor del llibre 1: " + llibre1.getAutor());  
        System.out.println("Autor del llibre 2: " + llibre2.getAutor());  
        llibre1.copiaAutor(llibre2);  
        System.out.println("2.");  
        System.out.println("Autor del llibre 1: " + llibre1.getAutor());  
        System.out.println("Autor del llibre 2: " + llibre2.getAutor());  
        llibre1.setAutor("");  
        llibre2.setAutor("Peter");  
        System.out.println("3.");  
        System.out.println("Autor del llibre 1: " + llibre1.getAutor());  
        System.out.println("Autor del llibre 2: " + llibre2.getAutor());  
        llibre1.copiaModificaAutor(llibre2);  
        System.out.println("4.");  
        System.out.println("Autor del llibre 1: " + llibre1.getAutor());  
        System.out.println("Autor del llibre 2: " + llibre2.getAutor());  
    }  
}
```

Exercici 3: Passo parametres

```
public class Llibre {  
    private String titol;  
    private String autor;  
  
    public Llibre(String titol){  
        this.titol = titol;  
        this.autor = "";  
    }  
  
    public Llibre(String titol, String autor){  
        this.titol = titol;  
        this.autor = autor;  
    }  
  
    public void setAutor(String autor){  
        this.autor = autor;  
    }  
  
    public String getAutor(){  
        return this.autor;  
    }  
}
```

```
    public void copiaAutor(Llibre llibre){  
        autor = llibre.autor;  
    }  
  
    public void copiaModificaAutor(Llibre llibre){  
        autor = llibre.getAutor();  
        llibre.setAutor("");  
    }  
  
    public void canviaObjecte(Llibre llibre){  
        llibre = new Llibre("Europa", "Paul");  
    }  
}
```

Exercici 3

```
public class ExemplePassoParam2 {  
  
    public static void main(String[] args){  
        ExemplePassoParam2 exemple= new ExemplePassoParam2();  
        exemple.passoParametre();  
    }  
  
    public void passoParametre(){  
  
        Llibre llibre1 = new Llibre("Europa", "Bob");  
        System.out.println("1.");  
        System.out.println("Autor del llibre 1: " + llibre1.getAutor());  
        llibre1.canviaObjecte(llibre1);  
        System.out.println("2.");  
        System.out.println("Autor del llibre 1: " + llibre1.getAutor());  
    }  
}
```

Exercici 4

Activitat 1: Què val m abans i després de cridar al mètode *changeValuePrimitiveType*?

```
public class Ex{
    public static void main(String[] args) {
        int m = 5;
        System.out.println("Value of m: " + m);
        changeValuePrimitiveType(m);
        System.out.println("Value of m: " + m);
    }
    public static void changeValuePrimitiveType(int m){
        m = 10;
    }
}
```

Exercici 4

Activitat 2: Quins valors s'imprimeixen en el primer i segon print?

```
public class Ex{
    public static void main(String[] args) {
        Day myDay = new Day (12, 30,0);
        (1) System.out.println("Day before the method: " +
                               myDay.getHour() + ":" +
                               myDay.getMinute() + ":" +
                               myDay.getSecond());

        changeValueReferenceType(myDay);

        (2) System.out.println("Day after the method: " +
                               myDay.getHour() + ":" +
                               myDay.getMinute() + ":" +
                               myDay.getSecond());
    }

    public static void changeValueReferenceType(Day oneDay){
        oneDay.setHour(24);
        oneDay.setMinute(24);
        oneDay.setSecond(24);
    }
}
```


Exercici 4

Activitat 3: I en aquest cas, que treballem amb anotherDay?

```
public class Ex{
    public static void main(String[] args) {
        Day myDay = new Day (12, 30,0);
        Day anotherDay = myDay;

        (1) System.out.println("Day before the method: " + myDay.getHour() +
            ":" + myDay.getMinute() + ":" + myDay.getSecond());

        changeValueReferenceType (myDay);

        (2) System.out.println("AotherDay after the method: " +
            anotherDay.getHour() + ":" + anotherDay.getMinute() + ":" +
            anotherDay.getSecond());
    }
    public static void changeValueReferenceType (Day oneDay) {
        oneDay.setHour (24);
        oneDay.setMinute (24);
        oneDay.setSecond (24);
    }
}
```

Nota: JAVA vs C++

- A JAVA, el pas d'objectes copia la referència
 - Això vol dir que si una funció rep com a paràmetre un objecte, i modifica els seus valors, fora de la funció, els canvis són visibles.
- C++ ens dóna la possibilitat de passar objectes per valor o per referència (&objecte)

... JAVA vs C++

- Si passem un objecte en C++ per valor a una funció: els canvis que fem a una funció no es veuen fóra

pseudo codi:

```
void main(){  
  Car myCar = new Car("Ferrari");  
  changeName(myCar);  
  print (myCar.getName());  
}  
void changeName (Car aCar){  
  aCar.setName("Renault");  
}
```

>> In C++, Ferrari

>> In JAVA, Renault (a on és el meu Ferrari?!)

Continguts

- Creació de classes i objectes
- Pas de paràmetres primitius i de referència
- **Mètodes de classe i d'objecte**
- Inicialització de variables
- Encapsulació

Mètodes de classe i d'objecte

- Un mètode d'objecte o d'instància és el que s'invoca sempre sobre una instància (objecte) d'una classe.
 - Per exemple `p1.getNombre()`; sent `p1` un objecte de tipus `Persona` és un mètode d'instància: per invocar necessitem una instància de persona.
- Un mètode de classe és aquell que pot ser invocat sense existir una instància.

Exercici 5

Activitat 1: Contesteu a les següent preguntes

```
public class Ex{  
    public String nameDay;  
    public static void main(String[] args) {  
        //Bloc 1  
        nameDay = "Monday";  
        System.out.println("Today is: " + nameDay);  
        //Bloc 2  
        Ex3 ex3 = new Ex3();  
        ex3.nameDay = "Monday";  
        System.out.println("Today is: " + ex3.nameDay);  
    }  
}
```

- Quin dels dos blocs pot accedir a l'atribut nameDay, i perquè?
- “Error: non-static variable cannot be referenced from a static context”: què significa?

Exercici 5

Activitat 2: Contesteu a la següent pregunta

```
public class Ex3{  
    public static String nameDay;  
    public static void main(String[] args) {  
        Ex3 ex3bis = new Ex3();  
        ex3bis.changeVariable();  
        System.out.println("Today is: " + ex3bis.nameDay);  
    }  
    public void changeVariable(){  
        nameDay = "Monday";  
    }  
}
```

- És correcte aquest codi? Si no ho és, perquè? Si sí, què imprimeix com a resultat?

Exercici 5: quants Daus es creen?

```
public class Dau{
    private final int MAX = 6;
    private int valorCara;
    private static int numDaus = 0;
    public Dau() {
        valorCara = 1;
        numDaus ++;
    }
    public int llansa() {
        valorCara = (int)(Math.random() * MAX) +1;
        return valorCara;
    }
    public void setValorCara(int valor) {
        valorCara = valor;
    }
    public int getValorCara() {
        return valorCara;
    }
    public static int getNumDausCreates(){
        return numDaus;
    }
    public static void main(String[] args) {
        for (int i=0;i<10;i++){
            Dau dau = new Dau();
            System.out.println(Dau.getNumDausCreates());
        }
    }
}
```


Continguts

- Creació de classes i objectes
- Pas de paràmetres primitius i de referència
- Mètodes de classe i d'objecte
- **Inicialització de variables**
- Encapsulació

Inicialització de variables

- Les **variables d'instància** (o atributs) **sempre tenen un valor** per defecte encara que no li assignem explícitament cap valor o no cridem a un mètode setter.
- Exemples:
 - integers 0
 - floating points 0.0
 - booleans false
 - references null
- Les **variables locals** **no tenen** un valor per defecte

Exercici 6

Activitat 1: Contesteu a les següent preguntes

```
public class Ex4{  
    public static String a;  
    public static boolean b;  
    public static double[] c;  
    public static void main(String[] args) {  
        //Bloc 1  
        (1) System.out.println(a);  
        (2) System.out.println(b);  
        (3) System.out.println(c);  
        //Bloc 2  
        String d;  
        int e;  
        String[] f;  
        (4) System.out.println(d);  
    }  
}
```

- Què imprimeixen les línies (1), (2), i (3)? Quins valors tenen els atributs?
- Podeu executar la línia (4)? Quins valors tenen les variables locals?

Continguts

- Creació de classes i objectes
- Pas de paràmetres primitius i de referència
- Mètodes de classe i d'objecte
- Inicialització de variables
- **Encapsulació**

Encapsulació

Modificadors de visibilitat

- **public**

```
public void QualsevolPotAccedir(){}  

```

Qualsevol classe des de qualsevol lloc pot accedir a les variables i mètodes d'instància públics.

- **private**

```
private String NumeroDelCarnetDeldentidad;  

```

Les variables i mètodes d'instància privats només poden ser accedits dins de la classe. No són accessibles des de les subclasses ni des de altres classes.

Encapsulació

Modificadors de visibilitat

- **friendly** (també anomenades 'default')
`void MetodeDelMeuPaquet(){}
Per defecte, si no s'especifica el control d'accés, les variables i mètodes d'instància se declaren friendly (amigues). Són accessibles per tots els objectes dins del mateix paquet, però no per els externs al paquet.`
- **protected**
`protected void NomesSubClasses(){}
Molt semblant a l'accés friendly, amb la següent excepció: la classe on es declara i les subclasses de la mateixa poden accedir a les variables i mètodes d'instància protegits.`

Exercici 7

Activitat 1. Podeu imprimir el valor de l'atribut x de la classe A?

```
package unPaquet;

public class A {
    private int x;
    public A() {
        x=1;
    }
}

package unPaquet;

public class C {
    A a;
    public C(){
        a=new A();
    }
    public void metodeC(){
        System.out.println("el valor de a és:" +
            a.x);
    }
}
```

Exercici 7

Activitat 2. Suposem que voleu mantenir l'atribut de la classe A com a *private*. Quines modificacions faríeu a la classe A per tal de poder imprimir el valor del seu atribut privat a la classe C?

```
package unPaquet;    package unPaquet;

public class A {      public class C {
    private int x;      A a;
    public A() {        public C(){
        x=1;            a=new A();
    }                  }
}                      public void metodeC(){
                       System.out.println("el valor de a és:"
+   ?);
                       }
```


Exercici 7

Activitat 3. Podeu imprimir el valor de l'atribut x de la classe A des de la classe C si A i C estan programades de la següent manera?

```
package unPaquet;
```

```
public class A {  
    public int x;  
    public A() {  
        x=1;  
    }  
}
```

```
package unAltrePaquet;
```

```
import unPaquet.A;
```

```
public class C {  
    A a;  
    public C() {  
        a=new A();  
    }  
    public void metodeC() {  
        System.out.println("el valor de a és:"  
+ a.x);  
    }  
}
```

Exercici 7

Activitat 4. Podeu imprimir el valor de l'atribut x de la classe A des de la classe B si A i B estan programades de la següent manera?

I si x estigués declarada així: `int x;` ?

```
package unPaquet;    package unAltrePaquet;

public class A {      import unPaquet.A;
    protected int x;
    public A() {
        x=1;
    }
}

public class B extends A {
    public B() {
        System.out.println("Constructor de B");
    }

    public void metodeB() {
        System.out.println("el valor de x és:" +
            this.x);
    }
}
```

Exercici 8

Implementa la classe **Cercle** amb

- Un contador de cercles,
- Dos mètodes per comparar cercles que retornin l'objecte de radi més gran.
 - Un mètode d'objecte
 - Un mètode de classe