

GRAU D'ENGINYERIA INFORMÀTICA
Curs acadèmic 2018-19

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**
 - Estat i comportament
 - Estructura d'una classe: atributs i mètodes
 - Construcció d'objectes i flux
 - Destructor
 - Exercicis
- Pas de paràmetres primitius i de referència
- Mètodes de classe i d'objecte
- Inicialització de variables
- Encapsulació

Creació de classes

Estat i comportament

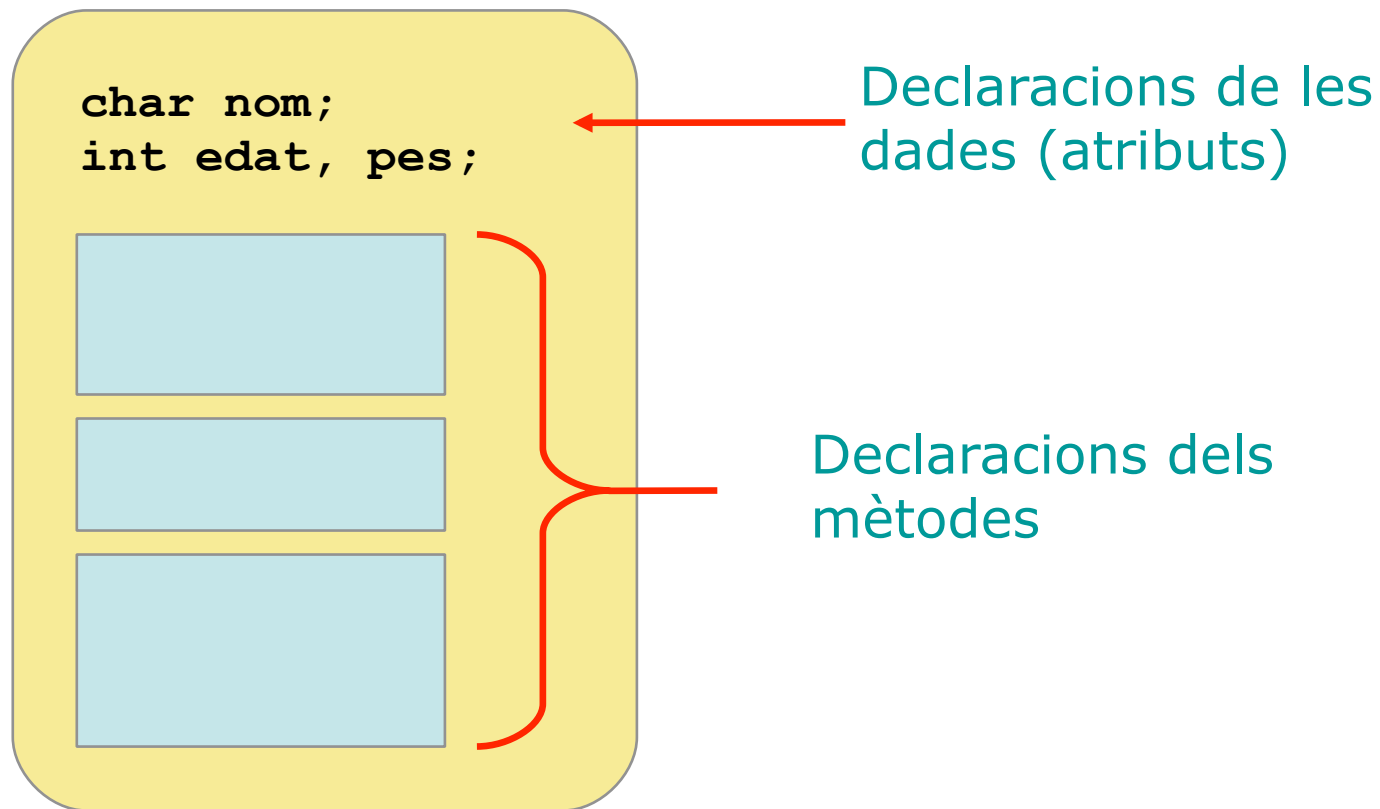
- Un objecte té estat i comportament
- Per exemple, la classe Persona
 - El seu estat està definit per Nom, Edat i Pes.
 - El seu comportament principal és CanviaEdat, ConsultaNom i ConsultaEdat

Persona
nom edat pes
canviEdat consultaNom consultaEdat

Creació de classes

Estructura

Classe Persona



Creació de classes

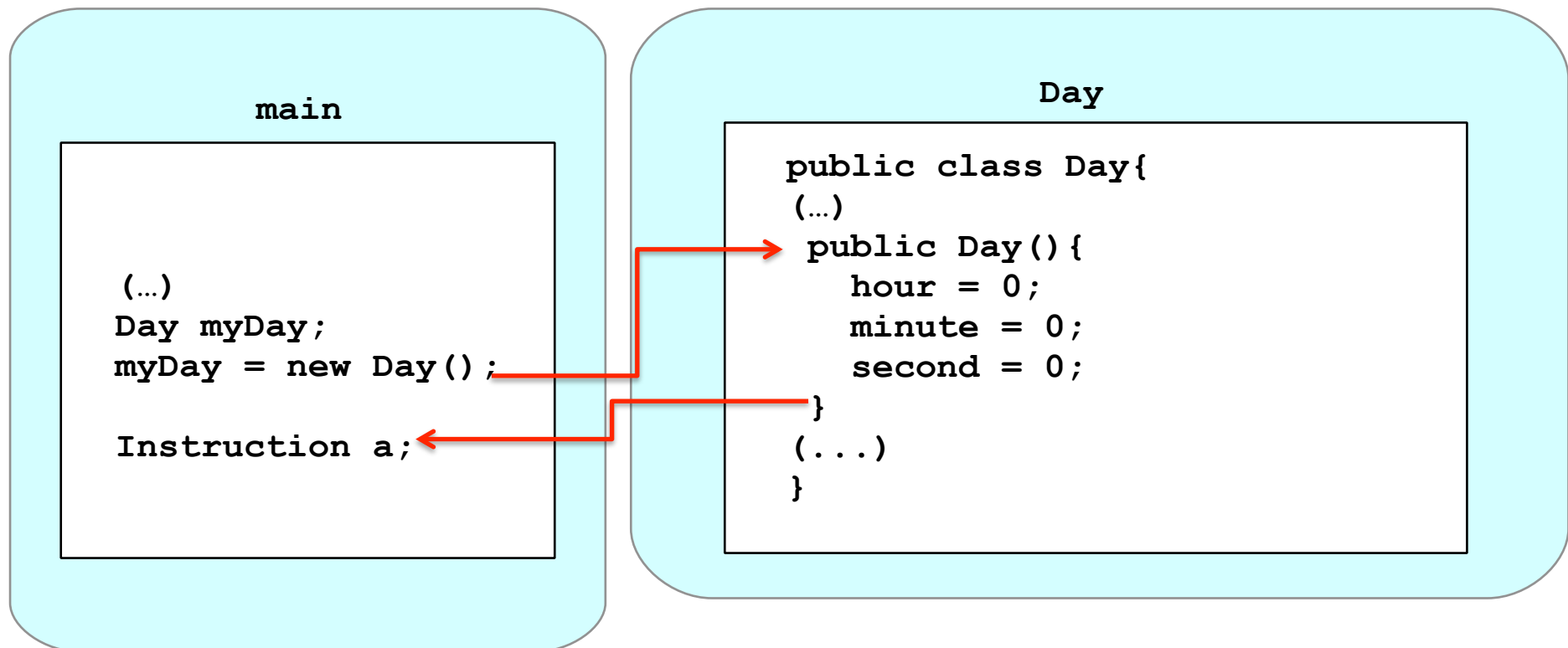
Construcció d'objectes

```
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 dóna l'error "*myDay has not been initialised*"?
- El Bloc 2, compila, però s'executa correctament?

Exercici 1 - Solució

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` ? **Declara una variable del tipus Day**
- Perquè el Bloc 1 ens dóna l'error "*myDay has not been initialised*"? **Perquè hem declarat una variable, però no l'hem inicialitzat. Què val myDay?**
- El Bloc 2, compila, però s'executa correctament? **No, ja que no tenim cap objecte Day creat**

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?

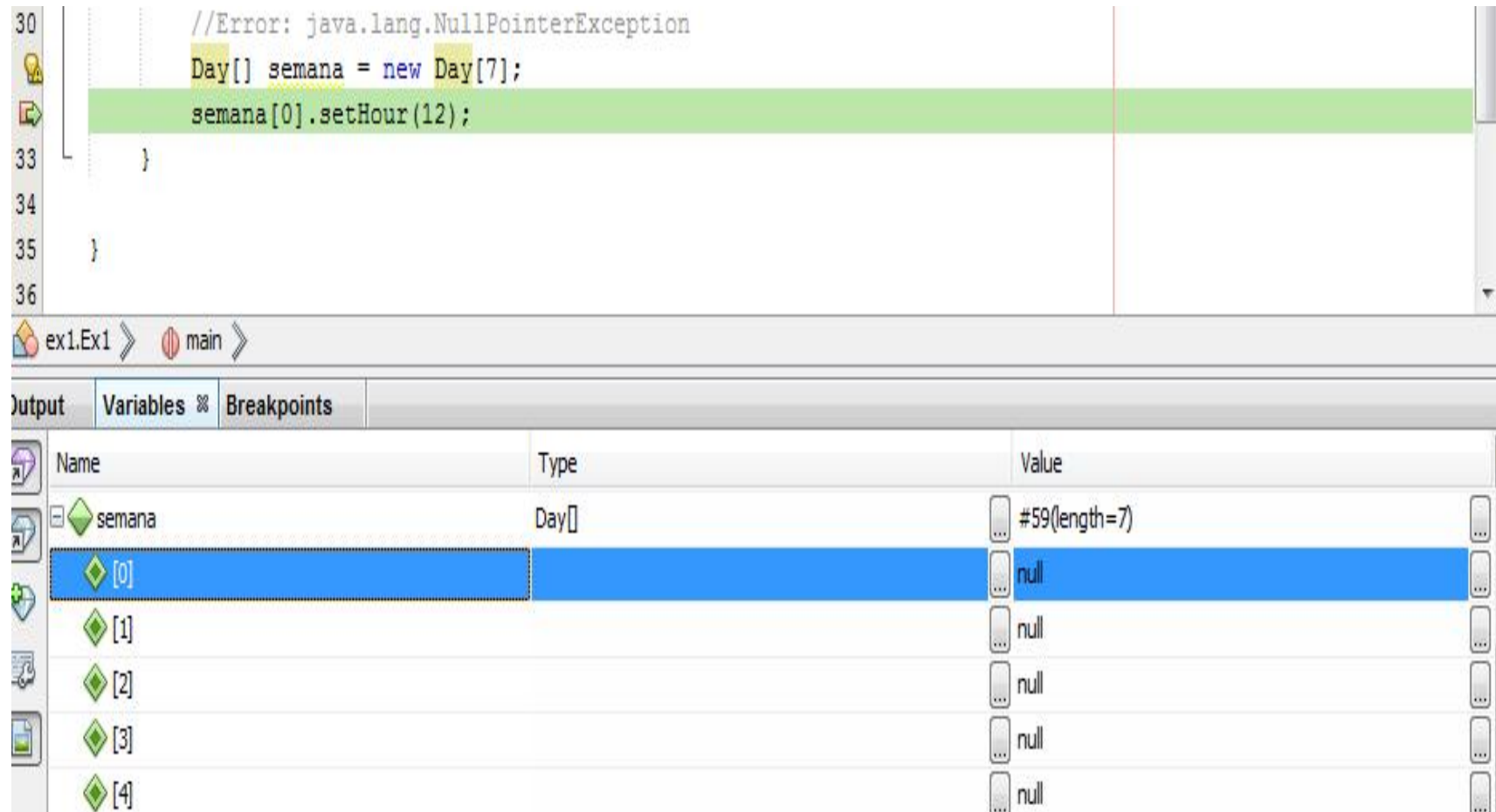
Exercici 1 - Solució

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]` ? Crea un array del tipus `Day` de 7 posicions.
- Quin bloc és el correcte, i perquè? El bloc correcte és el 2, perquè la posició 0 apunta a un objecte `Day` – la resta de posicions són null

Array d'objectes creat però vuit



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

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

The Variables window shows the state of the program:

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

The error message at the top of the code editor indicates a `java.lang.NullPointerException` at line 32, where `semana[0].setHour(12);` is executed. The Variables window shows that the array `semana` has a length of 7, and the first element `semana[0]` is `null`, which is the cause of the exception.

Continguts

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

Exercici 2

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

```
public class Ex2{
    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 2 - Solució

Activitat 1: Què val m abans i després de cridar al mètode *changeValuePrimitiveType*? 5

```
public class Ex2{
    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 2

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

```
public class Ex2{
    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 2 - Solució

Activitat 2: Quins valors s'imprimeixen en el primer i segon print? **12, 30, 0; 24, 24, 24**

```
public class Ex2{
    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 2

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

```
public class Ex2{
    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);
    }
}
```

Exercici 2 - Solució

Activitat 3: I en aquest cas, que treballem amb anotherDay?
El mateix resultat; tots dos són referències al mateix objecte

```
public class Ex2{
    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 sempre és per 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, és com passar en JAVA un tipus primitiu: 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

Continguts

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

Exercici 3

Activitat 1: Contesteu a les següent preguntes

```
public class Ex3{  
    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 3 - Solució

Activitat 1: Contesteu a les següent preguntes

```
public class Ex3{
    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? **El Bloc 2. Bloc 1 no pot perquè l'atribut no és static.**
- "Error: non-static variable cannot be referenced from a static context": què significa?
Estem accedint a una variable no estàtica des de un context (mètode) estàtic

Exercici 3

Activitat 2: Contesteu a la següent pregunta

```
public class Ex3{  
    public static String nameDay;  
    public static void main(String[] args) {  
        Ex3 ex3 = 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 3 - Solució

Activitat 2: Contesteu a la següent pregunta

```
public class Ex3{
    public static String nameDay;
    public static void main(String[] args) {
        Ex3 ex3 = 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? És correcte. Podem accedir a una variable estàtic des d'un context no estàtic. El resultat és Monday.

Exercici 3: 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());
        }
    }
}
```


Exercici 3: Què val numDaus al final del for? 10

```
public class Dau{
    private final int MAX = 6;
    private int valorCara;
    private static int numDaus = 0;
    public Dau() {
        valorCara = 1;
        numDaus ++;
    }
    public int llençar() {
        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

- 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 4

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?

Exercici 4 - Solució

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? **Null, false, 0.0**
- Podeu executar la línia (4)? **No**. Quins valors tenen les variables locals? **Cap**

Continguts

- Classes i objectes
- Pas de paràmetres primitius i de referència
- Mètodes de classe i d'objecte
- Inicialització de variables
- **Encapsulació**
 - Modificadors de visibilitat
 - Exercicis

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 accredits 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 5

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 5 - Solució

Activitat 1. Podeu imprimir el valor de l'atribut x de la classe A?
No. Tenim un error de compilació. La variable privada x no és visible fora de la seva classe

```
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:" +
                       a.x);
                       }
```

Exercici 5

Activitat 2. Supposeu 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 5 - Solució

Activitat 2. Supposeu 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;
```

```
public class A {  
    private int x;  
    public A() {  
        x=1;  
    }  
    public int getX(){  
        return  
        this.x;  
    }  
    public void  
    setx(int x){  
        this.x=x;  
    }  
}
```

```
package unPaquet;
```

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

Exercici 5

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 5 - Solució

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? **Sí. La variable és pública i és visible des de qualsevol classe**

```
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 5

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;
                     import unPaquet.A;

public class 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 5 - Solució

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? **Sí, x és visible per una subclasse de A**

I si x estigués declarada així: `int x;` ? **No, perquè és 'friendly'**

```
package unPaquet;    package unAltrePaquet;
                     import unPaquet.A;

public class 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);
    }
}
```