Programació I - Tema 2 - Entitats i expressions, sentències elementals

UNIVERSITAT DE BARCELONA

Grau en Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica
Curs 18-19

Programació I - T2- Entitats i expressions, sentències elementals

1. 2.1. Introducció
2. 2.2. Entitats (variables i constants) i expressions
3. 2.3. Sentències elementals

1 2.1. Introducció

2 2.2. Entitats (variables i constants) i expressions

3 2.3. Sentències elementals

2.1 Introducció

Recordem el concepte de programa.
Introduïm el concepte d'estructura de dades.

- **Programa:** Descripció NO AMBIGUA de les accions que cal realitzar per tal de donar la solució CORRECTA a un problema en un temps FINIT.
  Estructura de Dades: Descripció de les ENTITATS utilitzades en el transcurs d'un programa per EMMAGATZEMAR-HI les dades.
- Recordeu aquesta imatge? Les dades s'emmagatzemen a la memòria.

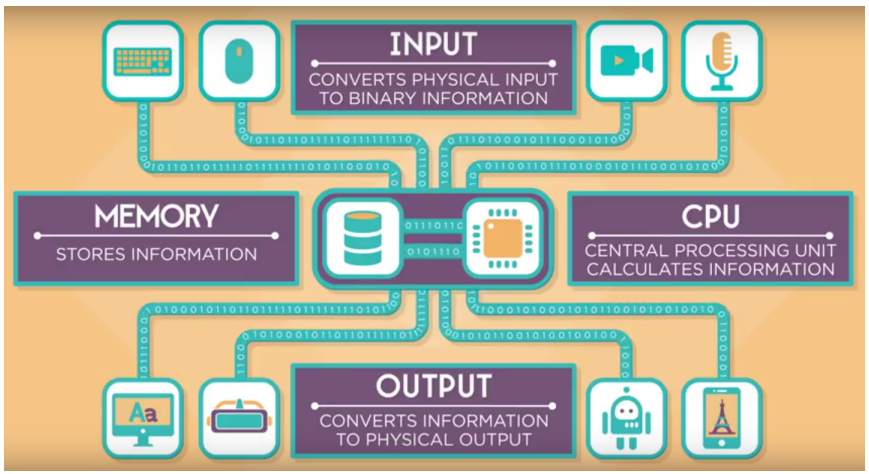
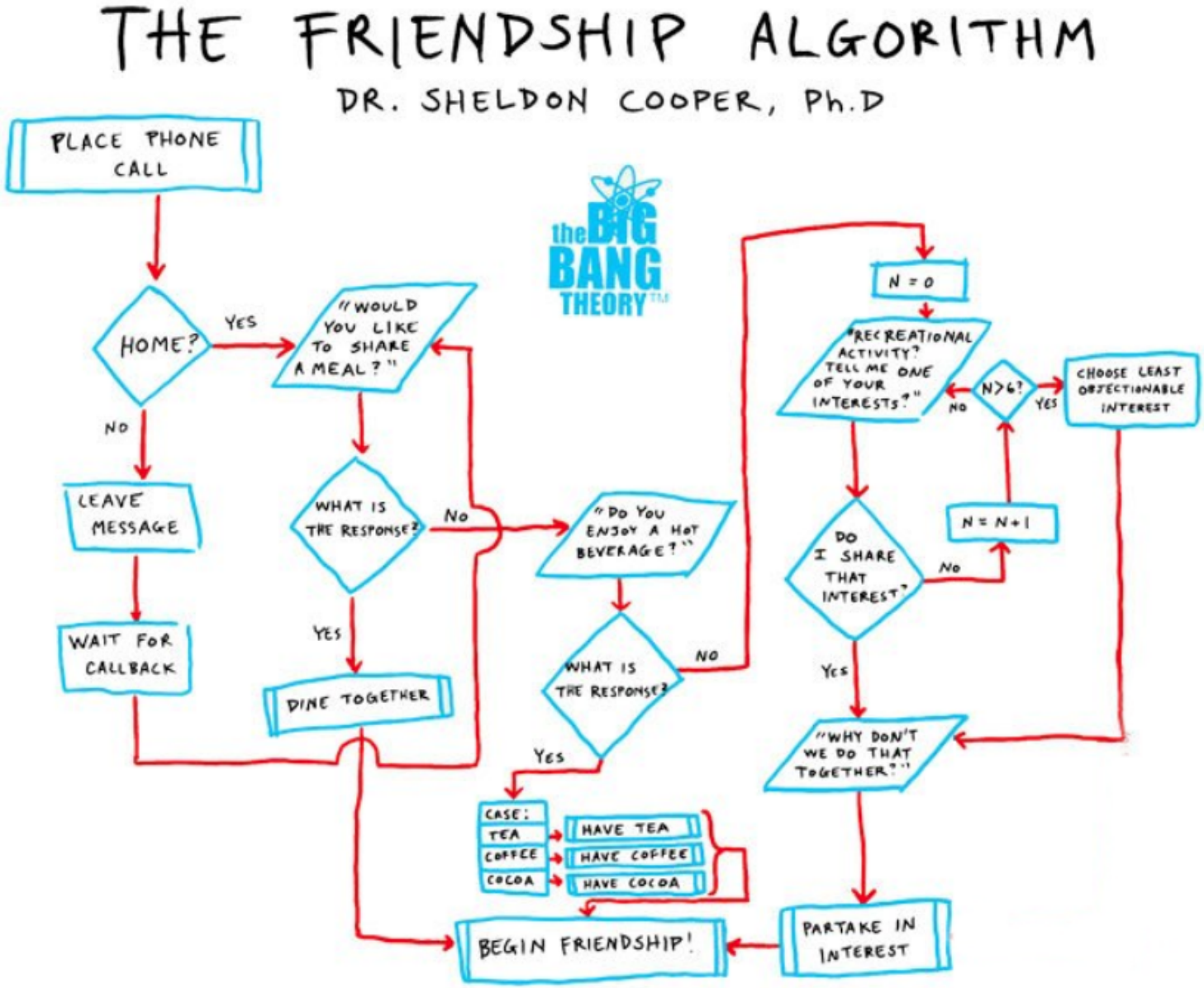


Image from 'How computers work', code.org

https://www.youtube.com/watch?v=DKGZlaPlVLY&list=PLzdnOPI1iJNcsRwJhvksEo1tJqjIqWbN-&index=5

2.1 Introducció



1 2.1. Introducció

2 2.2. Entitats (variables i constants) i expressions
- 2.2.1. Entitats
- 2.2.2. Declaració d'entitats
- 2.2.3. On trobem les entitats a un programa Java?
- 2.2.4. Expressions

3 2.3. Sentències elementals

Punt actual

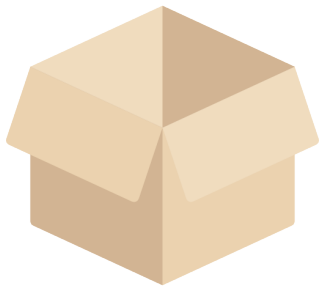
1. 2.1. Introducció

2. 2.2. Entitats (variables i constants) i expressions
   - 2.2.1. Entitats
   - 2.2.2. Declaració d'entitats
   - 2.2.3. On trobem les entitats a un programa Java?
   - 2.2.4. Expressions

3. 2.3. Sentències elementals

Concepte d'entitats

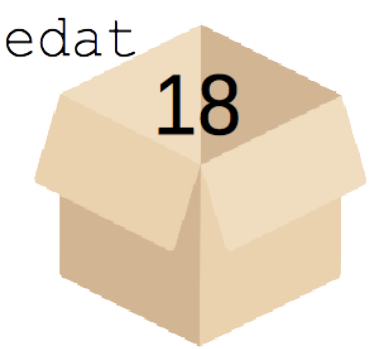
- Per a implememtentar l'algorisme 'Friendship', necessitem una manera d'emmagatzemar 'coses'. Una entitat es una 'caixa' que emmagatzema 'coses'.



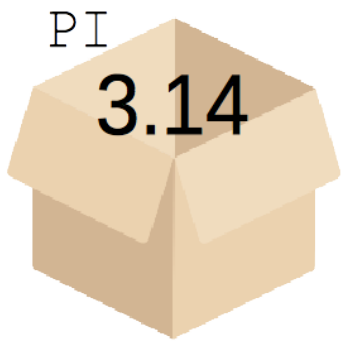
- Dit d'altre manera: les **entitats** són elements que permeten representar les dades (valors) amb les quals treballarà un programa.

Característiques de les entitats

- **Nom**: identificador de l'entitat.
- **Valor**: dada que representa.
  - Important! Veurem que aquesta dada pot variar durant l'execució del programa (variable) o és el mateix durant tota la execució del programa (constant)
- **Tipus**: conjunt de valors i operacions que es poden realitzar.



Nom: *edat*
Valor: *18*
Tipus: *int*



Nom: *PI*
Valor: *3.14*
Tipus: *double*

Característiques de les entitats: Nom

- Comença per una lletra, un subratllat (_) o un símbol $.
- Els següents caràcters poden ser lletres o dígits.
- Es distingeixen minúscules i majúscules.
- No es poden usar blancs entremig.
- No hi ha longitud màxima.
- No poden ser paraules clau de Java (keywords) o reservades.
  - `https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html`

Exemples:

```
identificador
nomUsuari
MAX_LONGITUD
_variableSistema
```

Característiques de les entitats: Valor

- La dada que conté l'entitat pot ser modificable o no al llarg del programa:
  - **Constants:** El valor no pot variar durant el programa.
    - Totes les lletres del seu identificador es posen en majúscules.
    - Si està formada per diferents paraules, es separen pel símbol del subratllat (_)
    - Exemples: IVA_LUXE, IVA_NORMAL
  - **Variables:** El valor pot ser modificat durant el programa.
    - Per convencions, la primera lletra ha de ser una minúscula.
    - El nom d'una variable ha de ser curt però entenedor.
    - S'eviten d'una sola lletre, excepte per a variables temporals (i, j, k, m, n s'usen de tipus enter; c, d, e per a tipus caràcters).
    - Si està formada per diferents paraules, les paraules internes comencen en majúscules (mixcase)
    - Exemples: suma, nomAlumne

Característiques de les entitats: Tipus

Tipus valor o primitius / Tipus referència

- El tipus determina:
  - l'espai de memòria ocupat.
  - el conjunt de valors que es poden representar (Rang).
  - la codificació interna utilitzada.
  - els literals o valors que contindrà l'entitat.
  - les operacions que es poden efectuar (Operadors).
- Variants:
  - **tipus valor o primitiu** (bàsics): byte, short, int, long, float, double, char, boolean.
  - **tipus referència**: objectes, ex. String, Gos, Scanner, ...
  - Fixeu-vos com els tipus primitius s'escriuen en minúscula i els tipus referència en majúscula!
  - Java és un llenguatge "statically-typed", vol dir què s'ha de declarar els tipus de dades de les vostres entitats abans d'utilitzar-les.

Característiques de les entitats: Tipus

- Tipus valor: valor atòmic (o únic)

Tipus	Ocupació (en bytes)	Codificació	Rang
byte	1	compl. a 2	$-128 \ldots 127$
short	2	compl. a 2	$-32768 \ldots 32767$
int	4	compl. a 2	$-214748364 \ldots 214748363$
long	8	compl. a 2	$-9223372036854775808 \ldots$ 9223372036854775807
float	4	IEEE 745	
double	8	IEEE 745	
char	2	Unicode UTF-16	'\u0000' (o 0) ... (o 65535) '\uffff' (o 65535)
boolean	1		false, true

- Tipus referència:
  - **String**: cadenes de caràcters.
  - Qualsevol altre classe d'objectes definida per l'API de Java (ex. Scanner) o creada per nosaltres mateixos (ex. Gos), etc.

Característiques de les entitats: Tipus

Operadors

- Tipus valor: valor atòmic (o únic)

Tipus	Valors (o Literals)	Operadors
byte	5	
short	24563	Unari: $-$
int	234234	Binaris: $,+,-,*,/,\%$
long	40358	
float	5.0f	
double	5.0d	
char	'c'	
boolean	true	!, &&, ||

`https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html`

- Tipus referència:
  - **String**: cadenes de caràcters. Mètodes: s.charAt(), s.length()...
    `http://docs.oracle.com/javase/8/docs/api/java/lang/String.html`.

Característiques de les entitats: Tipus

Operadors

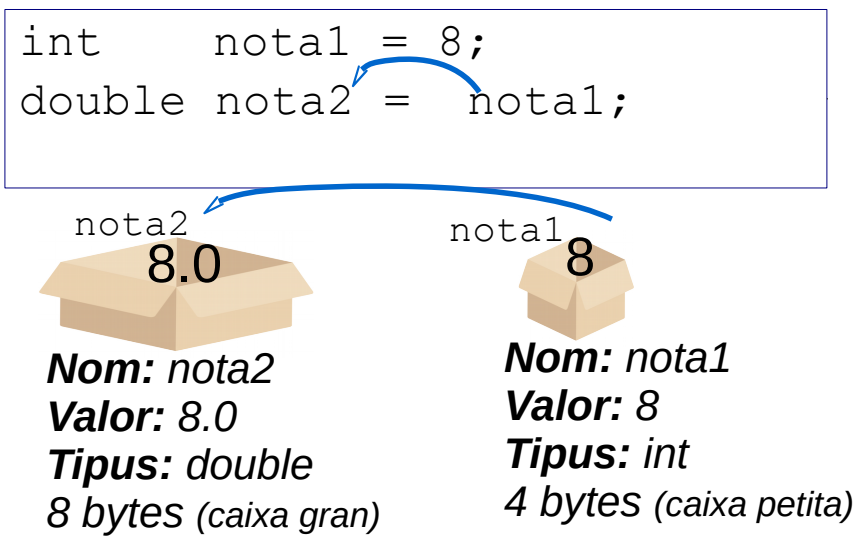
- **unaris**:
  - numèrics: -, ++, −−
  - lògics: !
- **binaris**:
  - numèrics: +, -, *, /, %
  - lògics: &&, ||
  - relacionals: **==**, <, >, <=, >=, !=
  - assignació: +=, -=, *=, %=, /=, ...
  - ...

Hi han més operadors dels citats en aquesta pàgina!
NOTA: Al laboratori veureu amb més detall la prioritat i l'associativitat d'operadors.

Característiques de les entitats: Tipus

Conversions entre tipus

- Valors del mateix tipus poden operar entre ells (**tipificació forta**).
- Valors de diferent tipus:
  - 1. Tot valor d'un tipus amb ocupació de memòria major es convert automàticament a un tipus d'ocupació menor (**cast IMPLÍCIT**).
    - Ex. `nota1` és un double que es convert automàticament a int quan s'assigna a `nota2`.

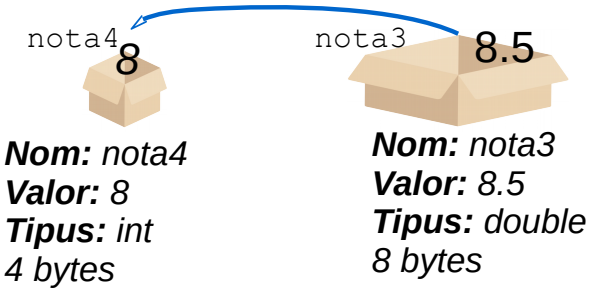


Característiques de les entitats: Tipus

Conversions entre tipus

- Valors de diferent tipus:
  - 2. Tot valor d'un tipus amb ocupació de memòria més gran cal convertir-lo de forma EXPLÍCITA a un tipus d'ocupació menor. És responsabilitat del programador(**cast EXPLÍCIT**).

```
double nota3 = 8.5;
int    nota4 = (int)nota3;
```



Fem un cast explícit 'manualment' **(tipus_a_convertir),** sinò ho fem és un error semàntic!

- Com puc ficar a una caixa petita coses que vénen d'una gran?
- Fem un cast explícit manualment (tipus_a_convertir), sinò ho fem és un error semàntic

Característiques de les entitats: Tipus

Conversions entre tipus

- Exemple de (**cast IMPLÍCIT**, automàticament).
  - `short a = 3; int b = 5;`
    `int i = a * b;`
    L'expressió `a * b` és semànticament correcta (n'hi ha concordança de tipus). El short (2 bytes) es converteix amb un cast IMPLÍCIT a int (4 bytes).
- Exemple de (**cast EXPLÍCIT**).
  - `short a = 3; int b = 5;`
    `short s = a * (short) b;`
    L'expressió `a * (short) b` és semànticament correcta perquè el programador ha fet la conversió de `b` a short de forma EXPLÍCITA, **(short)**.
- Més detalls: `http://docs.oracle.com/javase/specs/jls/se7/html/jls-5.html`

Punt actual

1. 2.1. Introducció

2. 2.2. Entitats (variables i constants) i expressions
   - 2.2.1. Entitats
   - 2.2.2. Declaració d'entitats
   - 2.2.3. On trobem les entitats a un programa Java?
   - 2.2.4. Expressions

3. 2.3. Sentències elementals

2.2.2. Declaració d'entitats

Declaració de variables

- **Sintaxi:**

```
tipusVariable nomVariable [=valorInicial];
```

- Exemples:
  - **tipus valor o primitiu**
    - `char a;`
    - `int b, c;`
    - `boolean d = false;`
  - **tipus referència**
    - `String t, u;`
    - `String v = new String("Hola");`
    - `String s;`
      `s = new String();`
    - `Circle circle;`
      `circle = new Circle(2.0);`
- Una variable és **visible** només en el bloc on està definida.
- Important! Recordeu que la paraula reservada `new` fa una crida al constructor d'una classe.

2.2.2. Declaració d'entitats

Declaració de constants

- **Declaració d'una constant dintre d'una classe:**

```
static final tipusConstant NOM_CONSTANT=valorConstant;
```

- Exemple: `static final float PI=3.14f;`
  El modificador `static` indica que no és necessari crear un objecte de la classe per a utilitzar la constant. El modificador `final` indica que el valor que emmagatzema MAX és invariable.

2.2.2. Declaració d'entitats

Declaració de constants

- **Declaració d'una constant dintre d'un mètode:**

```
final tipusConstant NOM_CONSTANT=valorConstant;
```

- Exemple: `final int MAX = 30;`
  El modificador `final` indica que el valor que emmagatzema MAX és invariable. La constant es pot utilitzar només a l'àmbit on està definida, es a dir, dintre del mètode.

2.2.2. Declaració d'entitats

Conjunt de constants

- El tipus `enum` serveix per a manipular un conjunt fix de constants.
- Es declara utilitzant la paraula clau **enum**
- **Sintaxi:**

```
enum nomtipusEnum {NOM_CONSTANT1, ...  ,NOM_CONSTANTn}
```

Exemple: `enum Day { SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY }`

- Veure exemples de codi amb enum al CV (`Calculatorv1.java`, `Calculatorv2.java`).

Punt actual

1. 2.1. Introducció

2. 2.2. Entitats (variables i constants) i expressions
   - 2.2.1. Entitats
   - 2.2.2. Declaració d'entitats
   - 2.2.3. On trobem les entitats a un programa Java?
   - 2.2.4. Expressions

3. 2.3. Sentències elementals

Variables locals i atributs

On trobem les entitats a un programa Java?

- 1) A la definició de qualsevol mètode (dintre de les claus { } del mètode). Diem que són **variables o constants locals** del mètode.
- 2) A la definició d'una classe (dintre les claus { } de la classe). Diem que són **atributs**. Veurem que poden ser atributs d'objecte i atributs de classe.

Variables o constants locals

On trobem les entitats a un programa Java?

- 1) A la definició de qualsevol mètode (dintre de les claus { } del mètode). Diem que són **variables o constants locals** del mètode.
  - Els paràmetres d'un mètode són variables locals del mètode.
  - Les variables locals només existeixen mentre el mètode s'està executant!

Variables o constants locals I

```
public class Exemple {

   public static void main (String [ ] args) {

     /* .... variables o constants locals ....*/

   }
 }

class AltreClasse{

  AltreClasse ( [<paràmetres>] ){

     /* .... variables o constants locals ....*/
  }
  <valor_retorn> metode1( [<paràmetres>] ){

    /* .... variables o constants locals ....*/
  }
}
```

Atributs

On trobem les entitats a un programa Java? (cont.):

- 2) A la definició d'una classe (dintre les claus $\{\ \}$ de la classe). Diem que són **atributs**.
  - En el cas de les variables, diem que són **atributs** i serveixen per a definir les característiques de la classe d'objectes que s'està definint.
  - En el cas de les constants, són valors fixos que s'utilitzen dintre de la classe. Diem que són constants definides dintre de la classe.
  - Veurem que es distingueix entre **atributs d'objecte** i **atributs de classe**.

Atributs

```
public class Exemple {

    public static void main (String [ ] args) {

    }
 }

class AltreClasse{

 /* .... atributs ..... */

   AltreClasse ( [<paràmetres>] ){

   }

   <valor_retorn> metode1( [<paràmetres>] ){

   }
}
```

De les entitats que trobes en aquest codi: quines són constants? quines són variables locals? quines són atributs?

```
import java.util.Scanner;
/* Classe ExempleArea
   Versio 1.0 7/7/2017
   Author Inma Rodriguez */
public class ExempleArea {
  public static void main (String[] args) {
    Scanner teclado = new Scanner(System.in);
    Circle circle0 = new Circle();
    Circle circle1 = new Circle(1.6);
    Circle circle2;
    double ra; //Podem anomenar-la radi enlloc de ra?

    System.out.println("Area del circle0 " + "amb radi "
                       + circle0.radi + " = " + circle0.calcularArea()
                           );

    System.out.println("Area del circle1 " + "amb radi "
                       + circle1.radi + " = " + circle1.calcularArea()
                           );

    System.out.print("Introdueix el radi del circle2: ");
    ra = teclado.nextDouble();
```

```
    circle2 = new Circle (ra);
    System.out.println("Area␣del␣circle2␣"+ "amb␣radi␣"
                       + circle2.radi + "␣=␣" + circle2.calcularArea()
                           );
  }
}
class Circle{
  double radi;
  //IMPORTANT! Els metodes amb el mateix nom que la classe
  //son els constructors de la classe.
  //Un constructor es crida al crear un objecte amb new.
  //Als constructors, mai s'indica tipus de retorn.
  Circle(){
  }

  Circle(double r){
    radi = r;
  }
  //<tipus_retorn> <nom_metode> (<parametres>)
  double calcularArea(){
    return Math.PI * radi * radi;
  }
}
```

Atributs d'objecte i atributs de classe

Definicions

- **Atributs d'objecte**: cada objecte (instància) de la classe té el seu valor de l'atribut. Cada objecte té el seu lloc de memòria (donat per la crida a **new**) on es guarden els valors dels seus atributs d'objecte!
  Es poden utilitzar mentre l'objecte existeix.
- **Atributs de classe**: són atributs comuns a tots els objectes d'una classe. Es defineixen amb la paraula clau **static**. S'emmagatzemen a una única posició 'fixa' de memòria (on 'tothom' accedeix). Es poden utiltizar encara que no existeixen objectes creats.

Atributs d'objecte I

Definició amb exemples

- **Atributs d'objecte**: cada objecte (instància) de la classe té el seu valor de l'atribut. Cada objecte té el seu lloc de memòria (donat per la crida a **new**) on es guarden els valors dels seus atributs d'objecte!
  Es poden utilitzar mentre l'objecte existeix.
  - Es defineixen, dintre de les claus de la classe, de la forma: `<tipus> nomAtributDObjecte;`
  - Notació per accedir (quan l'utilitzem!!): `nomObjecte.nomAtributDObjecte`

Atributs d'objecte II

Definició amb exemples

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

    }
 }

class AltreClasse{

  /* .... atributs ..... */

  AltreClasse ( [<paràmetres>] ){

  }

  <valor_retorn> metode1( [<paràmetres>] ){

  }


}
```

Atributs d'objecte III

Definició amb exemples

- Ex. `color` de pell és un atribut d'objecte de la classe `Gos`. Aquest atribut té el valor **blanc** per a `elMeuGos` i el valor **negre** per a `elTeuGos`.
- Ex. `marca` és un atribut d'objecte de la classe `Cotxe`. Aquest atribut té el valor **Seat** per a `elMeuCotxe`, i el valor **Ford** per a `elTeuCotxe`.
- Ex. `radi` és un atribut d'objecte de la classe `Circle`. (Veure classe `Circle` a dalt, Quin valor té aquest atribut per a cada objecte `Circle` que es defineix?)

Atributs de classe I

Definició amb exemples

- **Atributs de classe**: són atributs comuns a tots els objectes d'una classe. Es defineixen amb la paraula clau **static**. S'emmagatzemen a una única posició 'fixa' de memòria (on 'tothom' accedeix). Es poden utiltizar encara que no existeixen objectes creats.
  - Es defineixen, dintre de les claus de la classe, com qualsevol entitat però amb **static**:
    `static <tipus_dades> nomAtributDeClasse;`
  - Notació per accedir (quan l'utilitzem!!): `NomClasse.nomAtributDeClasse`
  - Ex. Tots els gossos es poden vacunar un nombre màxim de vegades (`numMaxVacunes`).
  - `numCircles` és un atribut de classe de la classe **Circle** perquè emmagatzema el nombre de circles creats fins al moment actual. Inicialment 0. NO s'emmagatzema aquest valor en cada objecte.

Atributs de classe I

```
import java.util.Scanner;
/* Classe ExempleAreaV0
 Versio 1.0 7/7/2017
 Author Inma Rodriguez */
public class ExempleAreaV0 {
  public static void main (String[] args) {
    Scanner teclado = new Scanner(System.in);
    Circle circle0 = new Circle();
    Circle circle1 = new Circle(1.6);
    Circle circle2;
    double ra; //Podem anomenar-la radi enlloc de ra?

    System.out.println("Num de circles: " + Circle.numCircles);

    System.out.println("Area del circle0 " + "amb radi "
                       + circle0.radi + " = " + circle0.calcularArea()
                           );

    System.out.println("Area del circle1 " + "amb radi "
                       + circle1.radi + " = " + circle1.calcularArea()
                           );
```

Atributs de classe II

```
    System.out.print("Introdueix el radi del circle2: ");
    ra = teclado.nextDouble();
    circle2 = new Circle (ra);
    System.out.println("Area del circle2 "+ "amb radi "
                       + circle2.radi + " = " + circle2.calcularArea()
                         );
    System.out.println("Num de circles: " + Circle.numCircles);
    circle0.mostrarCircle();
    circle1.mostrarCircle();
    circle2.mostrarCircle();

  }
}
class Circle{
  double radi;
  int id;
  static int numCircles = 0;

  Circle(){
    radi = 0.0;
    id = ++numCircles;
  }
```

Atributs de classe III

```
  Circle(double r){
    id = ++numCircles;
    radi = r;
  }

  double calcularArea(){
    return Math.PI * radi * radi;
  }

  void mostrarCircle(){
    System.out.println("El meu id: " + id + " i el meu radi: " + radi);
  }

}
```

Punt actual

1. 2.1. Introducció
2. 2.2. Entitats (variables i constants) i expressions
   - 2.2.1. Entitats
   - 2.2.2. Declaració d'entitats
   - 2.2.3. On trobem les entitats a un programa Java?
   - 2.2.4. Expressions
3. 2.3. Sentències elementals

2.2.4. Expressions

- Una expressió és:
  - Una constant.
  - Una variable.
  - Una crida a una funció.
  - La combinació de qualsevol dels anteriors amb operadors aritmètics i lògics.
- **Si A és una expressió:**
  - (A) és una expressió.
  - operadorUnari A és una expressió.
  - A operadorBinari B és una expressió.
- IMPORTANT! Veure document de suport a l'anàlisi d'expressions a l'espai de laboratori del CV

1 2.1. Introducció

2 2.2. Entitats (variables i constants) i expressions

3 2.3. Sentències elementals

2.3. Sentències elementals (Statements)

- Sentències:
  - Comentaris: `//`, `/* ... */`
  - Assignacions: `<identificador> = <expressió>;`
  - Crides a mètodes:
    - Entrada: `import java.utils.Scanner; Scanner lectura; lectura = new Scanner(System.in); nom_variable = lectura.nextXXXXX();`
    - Sortida: `System.out.println(<cadena> + nom_variable);`
- Acaben sempre en `;`
- Formen part sempre d'un bloc (o conjunt) { `<sentències>` }

2.3. Sentències elementals

Assignació de variables

- **Assignació:**
  - acció elemental que permet de donar valor a una variable
  - **Sintaxi:**

```
nomVariable = expressió;
```

S'avalua l'expressió i el valor del resultat es posa com a contingut nou de la variable

El tipus de la part esquerra ha de coincidir amb el tipus de la part dreta.

2.3. Sentències elementals

Assignació de variables. Exemples

- Exemples:
  - tipus valor
    - `a = 1;`
    - `b = 2 + 3;`
    - `c = 4 + b;`
    - `d = (float) c;`
  - tipus referència (exemple particular String)
    - `s = new String("Hola");`
    - `t = "Ho"+ "la";`
    - `u = s + "!";`
    - `String t = u.substring(1, 2); // "o"`

2.3. Sentències elementals

Assignació de variables amb operadors aritmètics

- L'assignació:

```
nomVariable += expressió
```

és equivalent a:

```
nomVariable = nomVariable + expressió
```

- Hi han d'altres operadors que funcionen igual -=, *=, /=, %=
- Abreviacions d'assignacions:

```
nomVariable++;
```

és equivalent a:

```
nomVariable = nomVariable + 1
```

Exercici: Ara anem a fer un programa per a calcular la longitud d'una circunferencia? (ExempleLongitud.java) I

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









  }
}
class Circunferencia{









}
```