



---

# R1.01

## Initiation au développement

---

# Les ordinateurs

---



- Peuvent être guidés (programmés) en plusieurs langues (langages) ;
- Ne mettent fondamentalement en œuvre qu'un nombre très restreint d'instructions :
  - La mémorisation / le rappel d'une valeur,
  - Les opérations,
  - Le saut à la prochaine instruction à exécuter, dans une liste prédéfinie et ordonnée,
  - Le choix du saut à une instruction ou une autre selon le résultat d'un test logique (vrai / faux)

# Les programmes d'ordinateurs

---

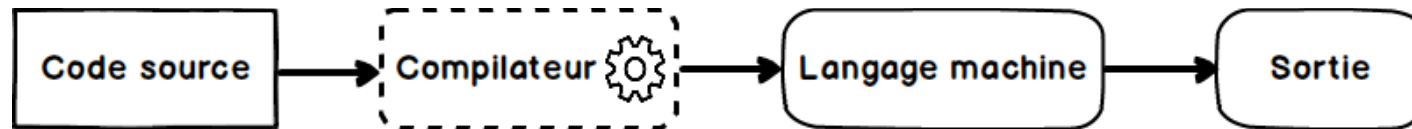


- Suites ordonnées d'instructions dans un langage compréhensible par l'ordinateur.
- Le langage : **java**, C, C++, python, CAML, PHP, javascript, ...
- Chaque langage **possède ses propres syntaxe, grammaire et spécificités.**
- Les programmes sont **compilés ou interprétés**, selon le langage.

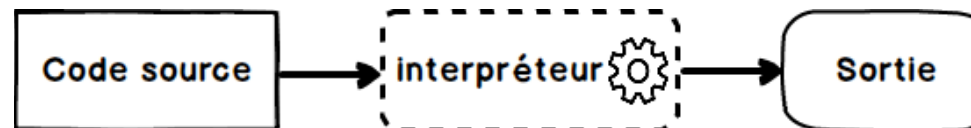
# Compilateur et l'interpréteur

Le **compilateur** et l'**interpréteur** sont deux manières différentes pour exécuter un programme écrit dans un langage de programmation.

- Un compilateur prend tout le programme et le convertit en code objet qui est généralement stocké dans un fichier.
  - Le code objet est également référencé en tant que code binaire et peut être exécuté directement par la machine après la liaison.



- Un interpréteur exécute directement des instructions écrites sans les convertir en un code objet ou un code machine.



# Table de comparaison

---



Interpréteur	Compilateur
Convertit le programme en prenant une seule ligne à la fois.	Analyse l'ensemble du programme et le traduit dans son ensemble en code machine.
L'analyse du code source prend moins de temps, mais le temps d'exécution global est plus lent.	L'analyse du code source prend beaucoup de temps, mais le temps d'exécution global est comparativement plus rapide.
Aucun code d'objet intermédiaire n'est généré, la mémoire est donc efficace.	Génère du code d'objet intermédiaire qui nécessite en outre une liaison, nécessite donc davantage de mémoire.
Continue de traduire le programme jusqu'à ce que la première erreur soit rencontrée. Par conséquent, le débogage est facile.	Il génère le message d'erreur uniquement après avoir analysé l'ensemble du programme. Par conséquent, le débogage est relativement difficile.
Exemple d'interpréteur: Python, Ruby.	Exemple de compilateur: C, C++.

# Qu'est ce qu'un algorithme?

---



- Quand un problème informatique se présente==> On est souvent amené à **créer un programme pour le résoudre.**
- Pour créer ce programme là, il faut utiliser un **langage de programmation**
- La logique de la programmation est tout simplement l'algorithme.
  - Un algorithme est **indépendant de tout langage de programmation,**
  - et en plus, il n'est pas destiné à être exécuté sur l'ordinateur comme c'est le cas d'un programme.
  - **Un algorithme est traduit en programme quand il est jugé correcte et efficace.**

# Les algorithmes

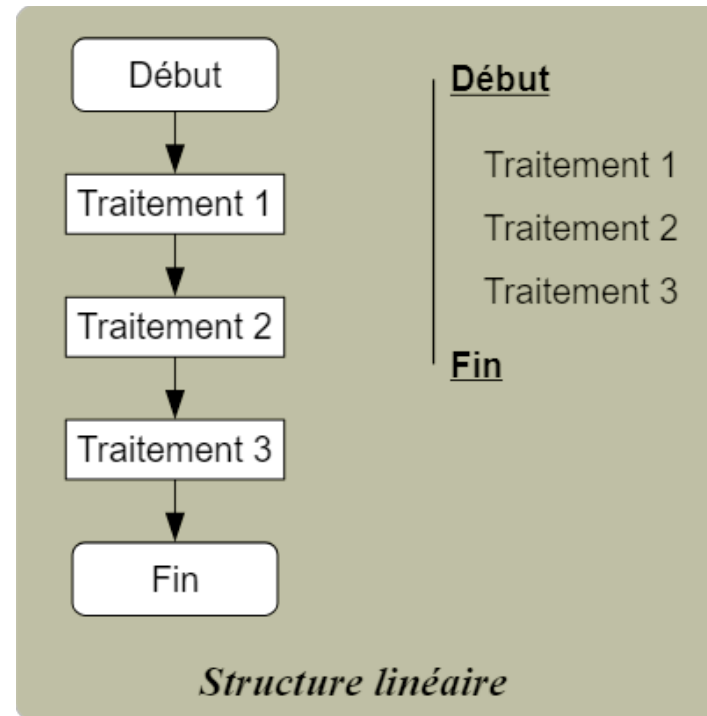
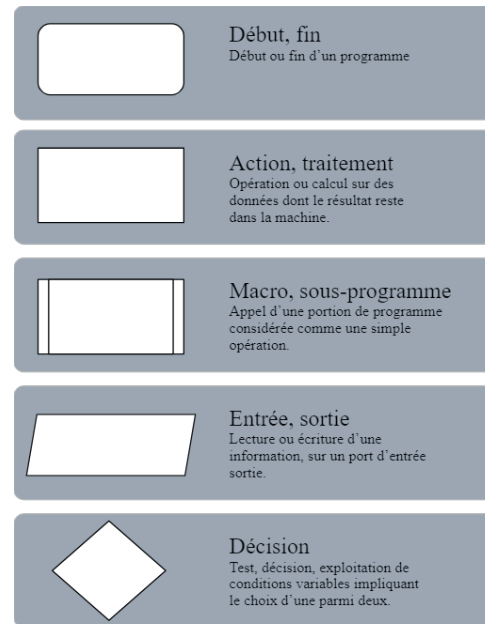
---



- Décomposer le problème **à résoudre en *sous-problèmes* plus simples.**
- Décrire comment résoudre les problèmes simples.
- Expliquer à un humain **comment** faire (Algorithme: la structure logique de la solution) ≠ Décrire à un ordinateur **quoi** faire (Programme: expression de la solution dans un langage informatique).

# Ecrire un algorithme sous forme d'organigramme ou pseudo-code

- Deux méthodes sont largement utilisées pour écrire un algorithme:
  - L'organigramme
  - Le pseudo-code



La description du processus doit être **sans ambiguïté** et **compréhensible** par celui qui devra le mettre en œuvre.



# Pseudo-code

---

- Découvrir comment écrire un algorithme en pseudo-code:
  - Les variables et leurs types,
  - La lecture et l'écriture,
  - Les opérateurs,
  - Les structures conditionnelles et itératives,
  - Les tableaux
  - Les sous-programmes (fonctions et procédures).



# L'écriture des algorithmes

---



- Beaucoup de façons de faire, pas de norme.
- Le **pseudo-code**:
  - n'est pas standardisé et peut varier d'un programmeur à l'autre.
  - est idéalement indépendant de tout langage de programmation.
  - est aussi idéalement indépendant de la machine qui exécutera le programme, réalisant (implémentant) l'algorithme.
- Nous écrirons nos algorithmes dans un pseudo-code, en français, dont nous présenterons les règles au fur et à mesure des chapitres.

# Exemple

## Pseudo-code

Algorithme **Alter**

Variables:

x entier

y entier

**Début**

x := 6

y := 12

**Si** x < y **alors**

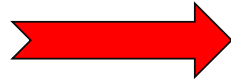
    x := y

**Sinon**

    y := x

**fin si**

**Fin**



## Program

**Alter.java**

```
class Alter{
    public static void main(String args[]){
        int x ;
        int y ;
        char z ;

        x = 6 ;
        y = 12 ;
        z = 'b' ;
        if (x < y)
            x = y ;
        else
            y = x ;
    }
}
```



# Problèmes & énoncés

---



- L'énoncé d'un problème donne des informations sur le résultat attendu.
  - **Exemples:** Calculer la surface d'une pièce d'habitation.
- L'énoncé peut être en langage naturel, imprécis, incomplet, et ne donne généralement pas la façon d'obtenir le résultat.
- Certains énoncé, quoiqu'élémentaires dans leur formulation, peuvent être difficiles à résoudre.

# Les variables

---



- Une **variable** permet d'associer un **nom** à une valeur.
  - Une variable est analogue à une boîte étiquetée à son nom, dans laquelle on range une valeur.
- Avant d'utiliser une variable, il faut la **déclarer**:
  - Mettre un nom sur l'étiquette
  - et dire quel **type** de valeur elle va contenir (nombre entier ou réel, caractère, etc.).

Variables:  
i2 entier  
x réel  
initiale caractère  
messageBienvenue chaine



- Le **type** de la variable est important. il détermine:
  - Les opérations que l'on peut effectuer.
  - La taille de la boîte.

# Types fondamentaux en java



## Numériques

Type	Description	Octets	Valeur mini	Valeur maxi
byte	petit entier	1	$-128 = -2^7$	+127
short	petit entier	2	$-32768 = -2^{15}$	+32767
int	entier	4	$-2147483648 = -2^{31}$	+2147483647
long	grand entier	8	$-9223372036854775808 = -2^{63}$	+9223372036854775807
float	réel simple précision	4	$-3.40282347 \cdot 10^{38} / 1.40239846 \cdot 10^{-45}$	$3.40282347 \cdot 10^{38} / -1.40239846 \cdot 10^{-45}$
double	réel double précision	8	$-1.79769313486231570 \cdot 10^{308}$ $4.94065645841246544 \cdot 10^{-324}$	$1.79769313486231570 \cdot 10^{308}$ $-4.94065645841246544 \cdot 10^{-324}$

## Non numériques

Type	Description	Octets	Valeur mini	Valeur maxi
boolean	booléen	1	0	1
char	caractère	2	\u0000	\uFFFF



---

<i>type</i>	<i>set of values</i>	<i>common operators</i>	<i>sample literal values</i>
int	integers	+ - * / %	99 12 2147483647
double	floating-point numbers	+ - * /	3.14 2.5 6.022e23
boolean	boolean values	&&    !	true false
char	characters		'A' '1' '%' '\n'
String	sequences of characters	+	"AB" "Hello" "2.5"

---

# Conventions de nommage

---



- Il est **PRIMORDIAL** de soigner la lisibilité des algorithmes et du code.
- L'adoption de règles de nommage des variable est un élément **ESSENTIEL** de la lisibilité.
- Les conventions applicables sont :
  - Former des noms explicites commençant par une **minuscule**, comme **prixUnitaire**.
  - Déclarer chaque variable sur une ligne et en commenter le rôle si nécessaire.
  - Initialiser si possible chaque variable lors de sa déclaration.
  - Mettre au pluriel le nom des tableaux (cella sera rappelé dans le cours *tableaux*).





---

## Question 1

---



On veut déclarer une variable représentant l'âge du capitaine.  
Qu'écrira-t-on de préférence en java ?

- A. `int age ;`
- B. `int ageCapitaine ;`
- C. `int n ;`
- D. `int AgeCapitaine ;`

# Les constantes

---

Une **constante** permet d'associer un **nom** à une valeur **qui ne changera pas au cours de l'algorithme.**

Constantes:

```
I_MAX = 2  
X_REF = 1.5  
FIRST_CHAR = 'b'  
TITRE = "myAlgo "
```

- Une constante *doit recevoir une valeur dès sa déclaration.*
- La valeur fournie pour **initialiser** la constante **détermine également son type.**

## Conventions de nommage

Les constantes doivent respecter les conventions suivantes :

- Être écrites en **majuscules**
- Séparer les mots par **un sous-tiret**, comme par exemple **TAUX\_TVA**.



---

## Question 2

---



En java, quel est le plus petit type d'entier permettant de représenter la valeur  $2^{18}$  ?

- A. byte
- B. short
- C. int
- D. long

# HelloWorld Exemple en java



Hello, World.

text file named HelloWorld.java

```
public class HelloWorld {  
    public static void main(String[] args)  
    {  
        // Prints "Hello, World" in the terminal window.  
        System.out.print("Hello, World");  
    }  
}
```

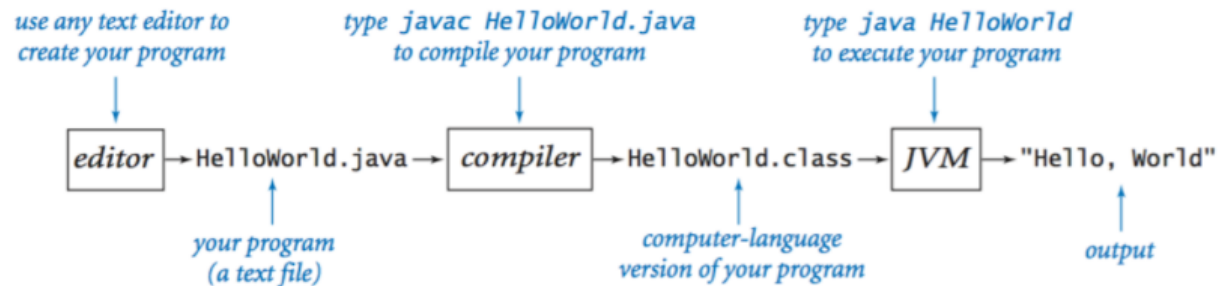
name

main() method

statements

body

## Editing, compiling, and executing.



# Lecture & écriture de données



- **Écrire** : Instruction permettant d'afficher la valeur de données rangées en mémoire.
- **Lire** : Instruction permettant de **ranger** des données fournies par l'utilisateur en mémoire, dans des variables, par exemple saisies au clavier.
  - Cela se fait simplement en écrivant le nom de la variable.

## Algorithme bonjour

### Variables:

prenom chaine

### Debut


**Écrire**("Entrez votre prénom :")

**Lire**(prenom)

**Écrire**("Bonjour " + prenom + " je suis heureux de te connaître.")

### Fin

## Bonjour.java



```
import java.util.Scanner; // Import the Scanner class
class Bonjour{
    public static void main(String args[]){
        Scanner myObj = new Scanner(System.in); // Create a Scanner object
        String prenom;
        System.out.print("Entrez votre prénom :");
        prenom=myObj.nextLine();
        System.out.println("Bonjour " + prenom + " je suis heureux de te connaître.");
    }
}
```

# Lecture & écriture de données

## Affectation

- Instruction permettant de **modifier** la valeur associée à une variable.
- En pseudocode, on la représente par le symbole **:=**

## Exemple:

### Algorithme age

#### Constantes:

ANNEE\_COURANTE = 2016

#### Variables:

annee entier

age entier

#### Debut

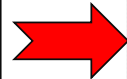
Ecrire("Entrez votre année de naissance :")

Lire(annee)

age := ANNEE\_COURANTE - annee

Ecrire("Vous avez donc " + age + " an(s).")

#### Fin



### Age.java

```
import java.util.Scanner; // Import the Scanner class
class Age{
    public static void main(String args[]){
        final int ANNEE_COURANTE = 2016;
        int annee;
        int age;

        Scanner myObj = new Scanner(System.in);

        System.out.println("Entrez votre année de naissance :") ;
        annee= myObj.nextInt();
        age= ANNEE_COURANTE - annee ;
        System.out.println("Vous avez donc " + age + " an(s).") ;
    }
}
```

En Java: Les constantes se déclarent avec le qualificatif **final** en début de ligne

# Séquences



Réaliser l'algorithme suivant sur feuille de brouillon à l'aide d'un tableau comportant autant de colonnes que de variables et autant de lignes qu'il y a d'instructions.

```
1 Algorithme Calcul
2 Constantes:
3   SEUIL = 13.5
4 Variables:
5   valA reel
6   valB reel
7   compteur entier
8 Debut
9   valA := 0.5
10  valB := valA
11  valA := valA*(10.5+SEUIL)
12  compteur := 1
13  compteur := compteur + 10
14  Ecrire(compteur)
15 Fin
```

n°ligne	valA	valB	compteur
9			
10			
11			
12			
13			

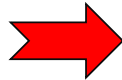
# Séquences

Dans une séquence, l'ordre dans lequel les instructions sont effectuées est important, même si plusieurs ordres sont parfois admissibles.

**Idée:** Echange de deux variables A et B

## Algorithme Echange

```
Variables
    valA reel
    valB reel
Debut
Ecrire("Entrez les valeurs pour A et B :")
Lire(valA)
Lire(valB)
valB := valA
valA := valB
Ecrire("valA")
Ecrire("valB")
Fin
```



## Echange.java

```
import java.util.Scanner;
class Echange{
    static Scanner myObj = new Scanner(System.in);

    public static void main(String args[]){

        double valA=0;
        double valB=0;

        System.out.println("Entrez les valeurs pour A :");
        valA= myObj.nextDouble();
        System.out.println("Entrez les valeurs pour B :");
        valB= myObj.nextDouble();
        System.out.println("valA="+valA+" , valB="+valB);
        valB = valA;
        valA = valB;
        System.out.println("valA="+valA+" , valB="+valB);

    }
}
```



**Question :** Quelles valeurs sont affichées pour A et B ?



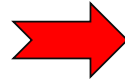
# Solution



## Algorithme Echange2

```
Variables
    valA reel
    valB reel
    temp reel

Debut
Ecrire("Entrez les valeurs pour A et B :")
Lire(valA)
Lire(valB)
temp := valB
valB := valA
valA := temp
Ecrire(valA)
Ecrire(valB)
Fin
```



## Echange2.java

```
import java.util.Scanner;
class Echange2{
    static Scanner myObj = new Scanner(System.in);

    public static void main(String args[]){

        double valA=0;
        double valB=0;
        double temp=0;
        System.out.println("Entrez les valeurs pour A :");
        valA= myObj.nextDouble();
        System.out.println("Entrez les valeurs pour B :");
        valB= myObj.nextDouble();
        System.out.println("valA="+valA+", valB="+valB);
        temp=valB;
        valB = valA;
        valA = temp;
        System.out.println("valA="+valA+", valB="+valB);

    }
}
```

## Question 3

Soit l'algorithme suivant

```
Programme Question3
Variables
    niveauBas réel
    niveauActuel réel
Début
    niveauBas = 2
    niveauActuel = 3
    niveauBas = niveauActuel
    niveauActuel = niveauBas
    écrire("niveau bas = "+niveauBas+", niveauActuel = "+niveauActuel)
Fin
```



Quel est la sortie produite à l'écran ?

- A. niveau bas = "niveauBas", niveauActuel = "niveauActuel"
- B. niveau bas = 3, niveauActuel = 3
- C. niveau bas = 2, niveauActuel = 2
- D. niveau bas = 3, niveauActuel = 2

## Opérations sur les variables en java: Cas général

---

- Les opérateurs arithmétiques standards n'effectuent leur opération que si les opérandes **sont tous du même type**.
- Le résultat est alors aussi du même type ; par exemple :
  - `int = int + int`
  - `double = double + double`
- Des opérandes de types différents sont parfois admis,
  - si l'opérande de plus *petit type* peut être *promu* au rang du type le plus grand.

Cast automatique	<pre>double y = 2 + 3.0 ; System.out.println("y =" + y) ;</pre>
Cast explicite	<pre>double z = (double)2/3 ; System.out.println("z =" + z) ; double w = 2.0/3 ; System.out.println("w =" + w) ;</pre>

## Question 4

---



Quel sera l'affichage produit par les instructions java suivantes :



```
final int NB_ENFANTS = 4 ;  
int poidsGâteau = 2 ;  
double poidsPart ;  
poidsPart = poidsGâteau / NB_ENFANTS ;  
System.out.println("Poids d'une part de gâteau : " + poidsPart + " kg.");
```

- A. Poids d'une part de gâteau : 2 kg.
- B. Poids d'une part de gâteau : 0.5 kg.
- C. Poids d'une part de gâteau : 2/4 kg.
- D. Poids d'une part de gâteau : 0 kg.

## Opérations sur les variables en java: Cas particuliers

---

- L'opérateur d'affectation **=** s'applique à tous les types, de droite à gauche.
- Les opérateurs **+** et **+=** permettent également la concaténation de chaînes de caractères.
- La concaténation via **+** et **+=** permet de *mélanger* chaînes de caractères et types fondamentaux.
- Les variables de types fondamentaux sont *transtypées* en chaînes avant d'être concaténées.
- **Penser à prioriser les opérations arithmétiques à l'aide de parenthèses (...) pour éviter des comportements non souhaités.**

### Exemples

```
int x=2 ;  
int y=3 ;  
System.out.println("x+y= " +x+y  ) ; // écrira "x+y= 23"  
System.out.println("x+y= " +(x+y)) ; // écrira "x+y= 5"
```

## Math library.

public class Math

double abs(double a)	absolute value of a
double max(double a, double b)	maximum of a and b
double min(double a, double b)	minimum of a and b
double sin(double theta)	sine of theta
double cos(double theta)	cosine of theta
double tan(double theta)	tangent of theta
double toRadians(double degrees)	convert angle from degrees to radians
double toDegrees(double radians)	convert angle from radians to degrees
double exp(double a)	exponential ( $e^a$ )
double log(double a)	natural log ( $\log_e a$ , or $\ln a$ )
double pow(double a, double b)	raise a to the bth power ( $a^b$ )
long round(double a)	round a to the nearest integer
double random()	random number in [0, 1)
double sqrt(double a)	square root of a
double E	value of e (constant)
double PI	value of $\pi$ (constant)



## Java library calls.

method call	library	return type	value
Integer.parseInt("123")	Integer	int	123
Double.parseDouble("1.5")	Double	double	1.5
Math.sqrt(5.0*5.0 - 4.0*4.0)	Math	double	3.0
Math.log(Math.E)	Math	double	1.0
Math.random()	Math	double	random in [0, 1)
Math.round(3.14159)	Math	long	3
Math.max(1.0, 9.0)	Math	double	9.0

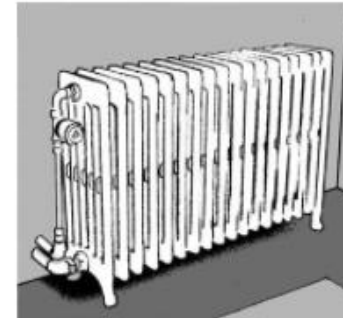
## Exercice n°1

---



Un chauffagiste doit déterminer, pour chaque pièce, le nombre de radiateurs à installer. Chacun des radiateurs choisis est capable de chauffer un volume donné  $V_u$ .

- 1) Écrire un algorithme qui permette au chauffagiste d'entrer:
  - Les dimensions de la pièce  $L, l, h$
  - La valeur  $V_u$  des radiateurs à installeret qui affiche en retour le nombre de radiateurs nécessaires



- 2) Traduire l'algorithme dans un programme Java complet.

# Solution: Algorithme Radiateurs



## Idée :

- Demander les dimensions de la pièce à l'utilisateur.
- Puis calculer le volume de la pièce et le diviser par la capacité de chauffage d'un radiateur ( $Vu$ ).
- En prendre ensuite la partie entière supérieure pour obtenir le nombre de radiateurs nécessaires.

## Variables:

$L$	réel	Longueur de la pièce
$l$	réel	Largeur de la pièce
$h$	réel	hauteur de la pièce
$Vu$	réel	capacité de chauffage d'un radiateur (m3)
volume	réel	volume de la pièce
nombre	entier	nombre de radiateurs

## Debut

**Ecrire**("Entrer la longueur de la pièce:")

**Lire**( $L$ )

**Ecrire**("Entrer la largeur de la pièce:")

**Lire**( $l$ )

**Ecrire**("Entrez la hauteur de la pièce:")

**Lire**( $h$ )

**Ecrire**("Entrer la capacité de chauffage d'un radiateur (m3):")

**Lire**( $Vu$ )

$Volume := L \times l \times h$

$Nombre := \lceil volume / Vu \rceil$

**Ecrire**(" Nombre de radiateurs nécessaires pour chauffer une pièce est : "+  $Nombre$ )

## Fin



# Radiateurs.java



```
import java.util.*;
class Radiateurs {
    static Scanner input = new Scanner(System.in);

    public static void main(String [] args) {
        double Longueur;
        double largeur ;
        double hauteur;
        double capacite;
        double volume;
        int nombre;

        System.out.println("Entrez la longueur de la pièce:");
        Longueur = input.nextDouble();

        System.out.println("Entrez la largeur de la pièce:");
        largeur = input.nextDouble();

        System.out.println("Entrez la hauteur de la pièce:");
        hauteur = input.nextDouble();

        System.out.println("Entrez la capacité de chauffage d'un radiateur (m3):");
        capacite = input.nextDouble();

        volume = Longueur * largeur * hauteur;
        nombre = (int)Math.ceil(volume / capacite );
        System.out. println ("Le nombre de radiateurs nécessaires est: " + nombre);
    }
}
```

## Exercice n°2

---

Un producteur de céréales pour petit déjeuner souhaite exporter ses produits vers le royaume uni. Il souhaiterait disposer d'un programme qui lui permette d'entrer:

- le poids unitaire des paquets de céréales exigé par ses clients anglo-saxons, exprimé en *onces* (oz).

et qui lui affiche en retour:

- le poids unitaire exprimé en *kg*
- le nombre de ces paquets qu'il pourra conditionner avec une tonne de céréales.



On sait que une tonne vaut 35 273,96 onces →  $1000\ 000 / 35\ 273,96$  → 1 once vaut 28,34g

# Algorithme: céréales

---



## Idée :

- Demander le poids unitaire des paquets de céréales à l'utilisateur.
- Puis calculer le poids unitaire exprimé en kg
- Ensuite calculer le nombre de ces paquets qu'il pourra conditionner avec une tonne de céréales.

## Constantes:

CONV\_ONCE\_GRAMME= 28,34

## Variables:

poidsPaquetOnces	reel
poidsPaquetGrammes	reel
kilos	reel
grammes	reel
nbPaquetsTonne	entier

## Debut

**Ecrire**("Entrer le poids unitaire des paquets de céréales en onces:")

**Lire**(poidsPaquetOnces)

**poidsPaquetGrammes** := poidsPaquetOnces  $\times$  convOnceGramme

**kilos** :=  $\lfloor \text{poidsPaquetGrammes} / 1000 \rfloor$

**grammes** := poidsPaquetGrammes - kilos  $\times$  1000

**Ecrire**("Un paquet pèse " + kilos + " kilos " + grammes + " grammes")

**nbPaquetsTonne** :=  $\lfloor 1000000 / \text{poidsPaquetGrammes} \rfloor$

**Ecrire**( "Le nombre de paquets obtenus a partir d'une tonne est: " + nbPaquetsTonne )

## Fin

---

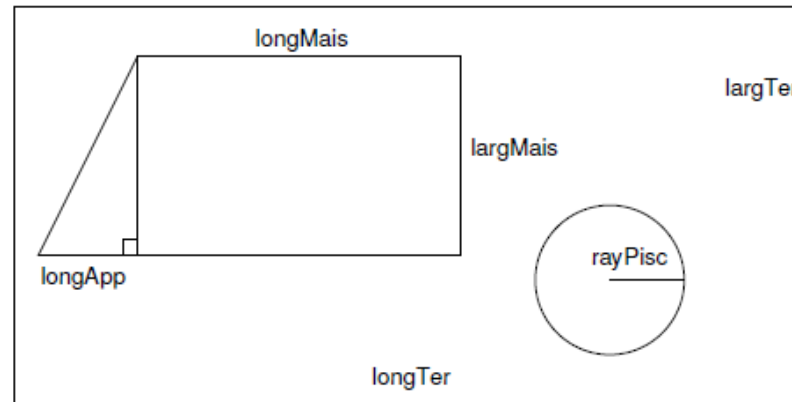
# Cereales.java



```
import java.util.*;
class Cereales {
    static Scanner input = new Scanner(System.in);
    static final double convTonneOnces= 35273.96;
    static final double convOnceGramme= 28.34;
    public static void main(String [] args) {
        double poidsPaquetOnces;
        double poidsPaquetGrammes;
        double kilos;
        double grammes;
        int nbPaquetsTonne;
        System.out.println("Entrer le poids unitaire des paquets de céréales en onces:") ;
        poidsPaquetOnces=input.nextDouble() ;
        poidsPaquetGrammes= poidsPaquetOnces*convOnceGramme;
        kilos=(int) Math.floor(poidsPaquetGrammes/1000);
        grammes= (int) Math.floor(poidsPaquetGrammes- (kilos*1000));
        System.out.println("Un paquet pèse " + kilos + " kilos " + grammes + " grammes");
        nbPaquetsTonne =(int) Math.floor(1000000/poidsPaquetGrammes);
        System.out.println("Le nombre de paquets obtenus a partir d'une tonne est: " + nbPaquetsTonne);
    }
}
```

## Exercice n°3

- 1) Écrire l'algorithme qui calcule la surface du gazon et le temps de tonte du gazon pour un terrain rectangulaire sur lequel sont bâtis une maison rectangulaire et un appentis triangulaire, et qui comporte une piscine circulaire. On sait que la tonte du gazon prend  $x$  secondes au  $m^2$  ( $x$  est donné), et on donne les dimensions indiquées sur la figure 2.



- 2) Traduire l'algorithme dans un programme Java complet.

# Algorithme Gazon

---



## Idée

Calculer la surface de gazon en retranchant de la superficie du terrain, la superficie occupée par les différents éléments.

Calculer ensuite le temps total de tonte en multipliant la surface de gazon par le temps de tonte unitaire.

## Constantes

$\pi$  réel = 3,141 592 653 59 constante mathématique

## Variables

<i>longTer</i> (réel)	//longueur du terrain (m)
<i>largTer</i> (réel)	//largeur du terrain (m)
<i>longMais</i> (réel)	//longueur de la maison (m)
<i>largMais</i> (réel)	//largeur de la maison (m)
<i>longApp</i> (réel)	//longueur de l'appentis (m)
<i>rayPisc</i> (réel)	//rayon de la piscine (m)
<i>x</i> (réel)	//temps de tonte unitaire (s/m <sup>2</sup> )
<i>surfaceGazon</i> (réel)	//surface du Gazon (m <sup>2</sup> )
<i>tempsTonte</i> (réel)	//temps total de tonte (s)

## Debut

```
Ecrire("Entrer la valeur .....")
Lire(longTer, largTer, longMais, largMais, longApp, rayPisc)
Lire(x)
surfaceGazon:= longTer× largTer-((longMais × largMais)+(longApp × largMais)/2)+(PI × rayPisc × rayPisc)
tempsTonte:= x × surfaceGazon
Ecrire("Surface de gazon : " + surfaceGazon + " m^2")
Ecrire("Temps de tonte : " + tempsTonte)
```

## Fin



```
import java.util.*;

class Gazon {

    static final Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        /* Données de l'algorithme du gazon */
        double longTer;
        double largTer;
        double longMais;
        double largMais;
        double longApp;
        double rayPisc;
        double x;
        /* Résultats de l'algorithme du gazon */
        double surfaceGazon;
        double tempsTonte;

        /***** Entrée des données *****/

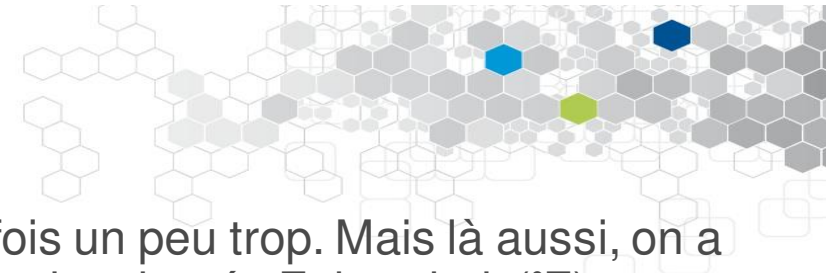
        System.out.println("Entrer la longueur et la largeur du terrain (m)");
        longTer = input.nextDouble();
        largTer = input.nextDouble();
        System.out.println("Entrer la longueur et la largeur de la maison (m)");
        longMais = input.nextDouble();
        largMais = input.nextDouble();
        System.out.println("Entrer la longueur de l'appentis (m)");
        longApp = input.nextDouble();
        System.out.println("Entrer le rayon de la piscine (m)");
        rayPisc = input.nextDouble();
        System.out.println("Entrer le temps de tonte unitaire (s/m^2)");
        x = input.nextDouble();

        /***** Algorithme du gazon *****/
        surfaceGazon = longTer * largTer - ((longMais * largMais) + ((longApp * largMais) / 2.0) + (Math.PI * rayPisc * rayPisc));
        tempsTonte = x * surfaceGazon ;

        /***** Affichage des résultats *****/
        System.out.println("Surface de gazon : " + surfaceGazon + " m^2");
        System.out.println("Temps de tonte : " + tempsTonte + " seconds");
    }
}
```

## Exercice n°4

---



Entre les français et les anglo-saxons, la température monte parfois un peu trop. Mais là aussi, on a du mal à s'entendre sur la mesure, entre les degrés Celsius (°C) et les degrés Fahrenheit (°F).

Sachant que la relation liant les valeurs  $F$  (en °F) et  $C$  (en °C) est donnée par la formule

$$F = \frac{9}{5}C + 32$$

- 1) Écrire un algorithme que lit une température exprimée en °C et affiche en retour la valeur équivalente en °F.
- 2) Traduire l'algorithme dans un programme Java complet.



# Solution

---



## Algorithme DegresCtoF

### Variables:

tempCelsius reel  
tempFahrenheit reel

### Debut

**Ecrire**("Algorithme de conversion deg. C vers deg. F")

**Ecrire**("Entrer une température en Celsius : ")

**Lire**(tempCelsius)

tempFahrenheit:=  $9.0/5 * \text{tempCelsius} + 32$

**Ecrire**("La température en Fahrenheit est : " +  
tempFahrenheit)

### Fin

```
// Programme de conversion de degrés Celsius vers Fahrenheit

import java.util.*;

class DegresCtoF {

    static final Scanner input = new Scanner(System.in);

    public static void main(String[] args) {

        double celsius;
        double fahr;

        System.out.println("Programme de conversion deg. C vers deg. F");
        System.out.println("Entrer une température en Celsius : ");
        celsius= input.nextDouble();

        fahr = 9.0/5 *celsius+ 32;

        System.out.println("La température en Fahrenheit est : " + fahr);

    }

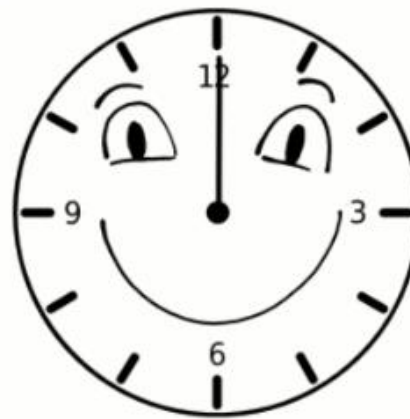
}
```

## Exercice n°5

---



- 1) Écrire, en utilisant les opérateurs **modulo** et **division**, un algorithme qui lit une durée exprimée en secondes et affiche en retour sa valeur, exprimée en heures, minutes et secondes,
- 2) Traduire l'algorithme dans un programme Java complet.



$$Duree = heures * 3600 + minutes * 60 + seconds$$



---

## Algorithme DureHeuresMinutesSeconds

### Variables:

duree entier  
heures entier  
minutes entier  
seconds entier

### Debut

**Ecrire**("Algorithme de conversion durée (seconds) vers heures, minutes et secondes")

**Ecrire**("Entrer une durée en seconds : ")

**Lire**(duree)

heures := duree/3600

minutes :=(duree modulo 3600)/60

seconds:=duree modulo 60

**Ecrire**("Duree: "+heures+ "h, "+minutes+" m " +seconds +" s " )

### Fin

---



```
import java.util.*;

class DureHeuresMinutesSeconds{

    static final Scanner input = new Scanner(System.in);

    public static void main(String args[]){
        int duree;
        int heures,minutes,seconds;

        System.out.println("Algorithme de conversion durée (seconds) vers heures, minutes et secondes");
        System.out.println("Entrer une durée exprimée en secondes:");
        duree= input.nextInt();

        heures = duree/3600;
        minutes =(duree % 3600)/60;
        seconds=duree % 60;
        System.out.println("Duree: "+heures+ " heures, "+minutes+" minutes, et " +seconds + " seconds  ");
        System.out.println("Duree: "+(heures*3600+minutes*60+seconds) );

    }
```

## Exercice n°6

---



Écrire un algorithme qui calcule le taux d'inflation de l'année passée.

→ Pour cela, l'algorithme demande les prix d'un même objet



- il y a un an,
- aujourd'hui.

On estime alors le taux d'inflation comme la différence des deux prix divisée par le prix de l'an passé.

Compléter l'algorithme pour qu'il imprime aussi le prix estimé de l'objet dans un an et dans deux ans en considérant que l'inflation demeure constante.



### Inflation Formula


$$\text{Rate of Inflation} = \frac{\text{CPI}_{x+1} - \text{CPI}_x}{\text{CPI}_x}$$


Traduire l'algorithme dans un programme Java complet.