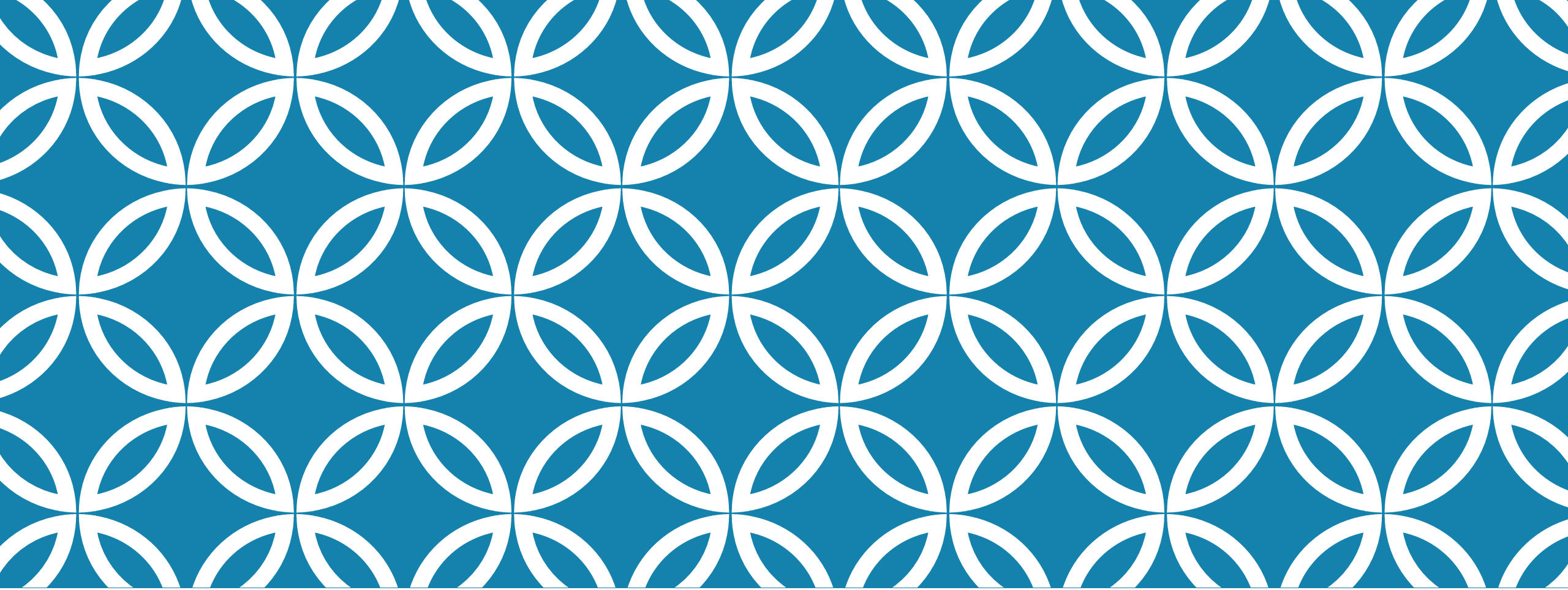
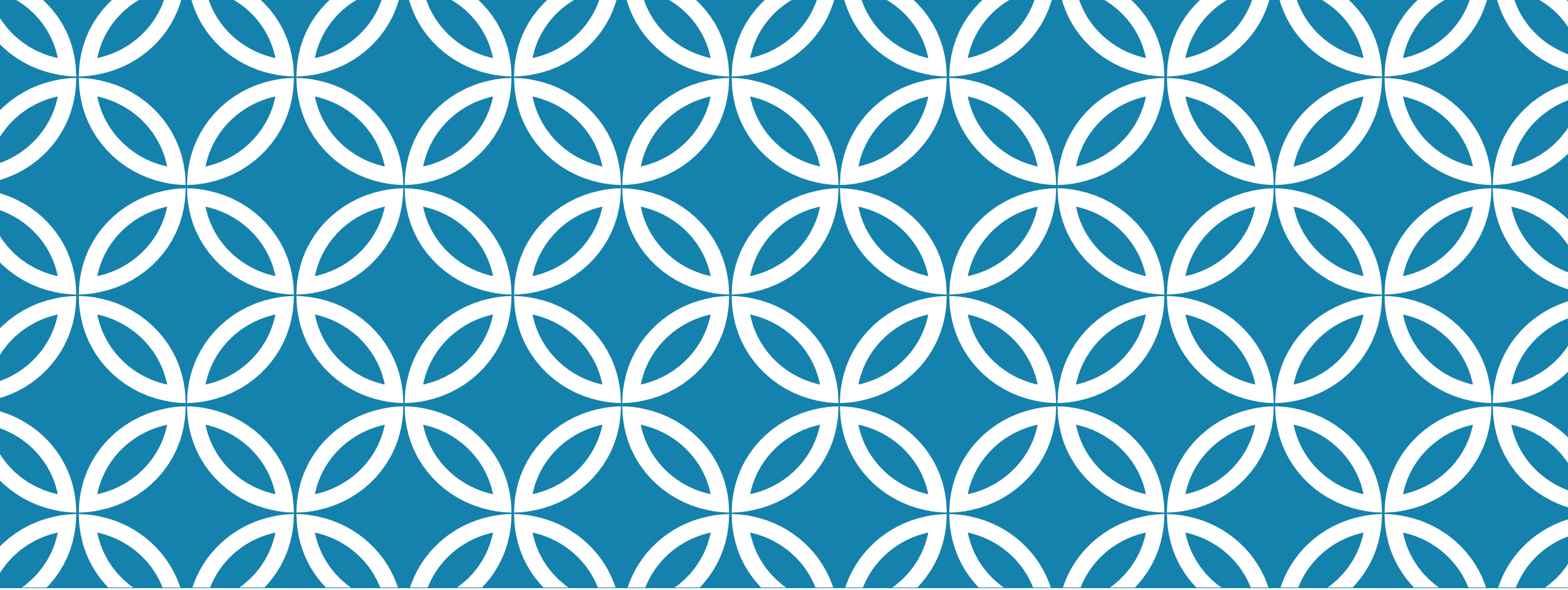


FORMATION ALGORITHMIQUE

Jordan ABID



DÉFINITION

QU'EST-CE QU'UN ALGORITHME ?

Comment un GPS trouve-t-il l'itinéraire qui nous fera esquiver les bouchons ? Comment Google nous présente-t-il toujours la page que nous cherchons ?
Grâce à l'algorithmique ! C'est un des outils les plus puissants que nous avons pour résoudre les problèmes qui se mettent en travers de notre route. C'est en effet la base de tout programme informatique.
Quelque soit le langage utilisé, l'algorithme sera le même, il sera seulement traduit dans le langage concerné.
Il nous est ainsi possible d'écrire un Algorithme en français (Pseudo-Code), puis de le transcrire en JAVA, C, PHP….

QU'EST-CE QU'UN PROGRAMME ?

Un programme est un ensemble d'instructions exécutables par un ordinateur et qui permet à ce dernier de répondre à un souci que nous nous posons.

il prend généralement un ensemble de données en entrée, exécute des instructions puis retourne des données en sortie.

Les actions effectuées par un programme sont des *instructions*. Une instruction peut être une opération de base, une exécution conditionnelle, une itération, un affichage ou une saisie.

Opérations de base : addition, soustraction, multiplication, division...

Exemple : "tire" + "-" + "bouchon" => "tire-bouchon"

Exécution conditionnelle : si (condition), alors (fais ça), sinon (fais ça).

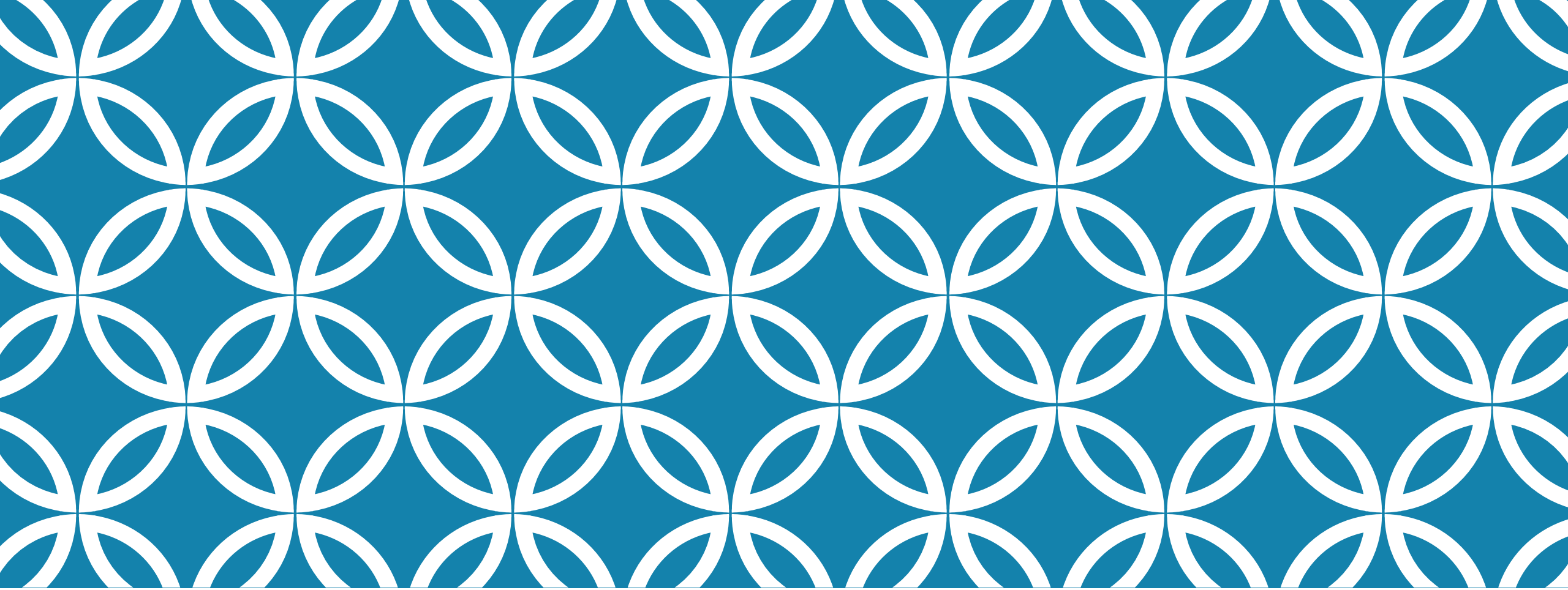
Exemple : si je suis connectée, affiche "Salut Jordan !"

Itération : répéter une instruction un nombre déterminé de fois.

Exemple : affiche chaque Eleve de la Classe.

Affichage/Saisie : Afficher un message à l'utilisateur, récupérer une donnée saisie par l'utilisateur.

Exemple : Afficher «Saisir votre age», faire saisir l'age.



INTRODUCTION AU LANGAGE JAVA

DÉCLARATION D'UN ALGORITHME

En Pseudo Code

```
Début
        Traitement
Fin
```

En JAVA

```
public static void main(String[] args) {
        Traitement
}
```

LES VARIABLES

Une variable est comme un tiroir dans un meuble de rangement.
Une variable a un nom unique qui nous permet de l'identifier et un contenu.

Dans la plupart des langages informatique, il faut également spécifier le type du contenu : Caractères, Entier…

Pseudo Code

```
Variables : nom Caractères, age Entier
Début
        age<=25
Fin
```

JAVA

```
public static void main(String[] args)
{
        String nom;
        int age;
        age=25;
}
```

LES VARIABLES

Type	Taille en octets	Représentation
boolean	Non spécifié	true /false
char	2	Caractère unicode
String	Non spécifié	Chaine de Caractères
short	2	$[-2^{15} ; 2^{15}-1]$
int	4	$[-2^{31} ; 2^{31}-1]$
long	8	$[-2^{63} ; 2^{63}-1]$
float	4	$[-3,4*10^{38} ; 3,4*10^{38}]$
double	8	$[-1,7*10^{308} ; 1,7*10^{308}]$

AFFICHAGE

Ici, on veut afficher à l'écran « Saisir votre Age »
On affiche ensuite « Vous avez 30 ans ! » à l'aide d'une variable

Pseudo Code

```
Variables : age Entier
Début
        age=30
        Afficher « Saisir votre Age »
        Afficher « Vous avez »+age+« ans !»
Fin
```

JAVA

```
public static void main(String[] args)
{
    int age=30;
    System.out.println("Saisir votre age");
    System.out.println("Vous avez "+age+ " ans ! ");
}
```

SAISIE

Pour permettre à l'utilisateur de saisir des données au clavier, il y a trois étapes :

- *Déclarer une variable de type Scanner (une seule fois)*
- *Déclarer le type de donnée que l'utilisateur doit saisir*
- *Affecter cette donnée dans une variable du même type*

Pseudo Code

```
Variables : age Entier

Début

        Afficher « Saisir votre Age »

        Saisir age

        Afficher « Vous avez  »+age+« ans !»

Fin
```

JAVA

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

        int age;

        Scanner sc = new Scanner(System.in);

        System.out.println("Saisir votre age");

        age =  sc.nextInt();

        System.out.println("Vous avez "+age+ " ans ! ");

}
```

LES PROCÉDURES ET FONCTIONS

En informatique, on appel procédures ou fonctions une suite de traitements, en général récurrents, que l'on place dans un bloc pouvant être appelé et exécuté dans tout notre code.

Elles peuvent recevoir de 0 à N variables (paramètres) pour effectuer leurs traitements.
Une fonction retournera un résultat pouvant être manipulé par la suite dans le code alors qu'une procédure ne retournera rien.

Il faudra leur donner un nom, un type de retour ainsi que le type de ses paramètres.

Attention : Les variables envoyés ne subissent aucun changement, les fonctions utilisent une copie des valeurs

LES PROCÉDURES ET FONCTIONS

Lorsque l'on souhaite faire plusieurs saisies, on passe obligatoirement par les étapes suivantes :

- Déclaration des scanners
- Affichage d'un message pour demander une saisie
- Affectation de la saisie dans une variable

Grâce aux fonctions, on peut simplifier ce traitement.

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

        Scanner sclnt=new Scanner(System.in);

        Scanner scString=new Scanner(System.in);

        System.out.println("Saisir votre age");

         int age= scString.nextInt();

        System.out.println("Saisir votre nom");

        String nom= scString.nextLine();

        System.out.println("Saisir votre nom");

        String nom= scString.nextLine();

}
```

```
public static int saisieInt(String message)
{
    Scanner sc=new Scanner(System.in);
    System.out.println(message);
    return sc.nextInt();
}

public static String saisieString(String message)
{
    Scanner sc=new Scanner(System.in);
    System.out.println(message);
    return sc.nextLine();
}

public static void main(String[] args) {

        int age= saisieInt("Saisir votre age");

        String nom= saisieString("Saisir votre nom");

        String prenom= saisieString("Saisir votre prenom");

}
```

CALCULETTE – ETAPE 1

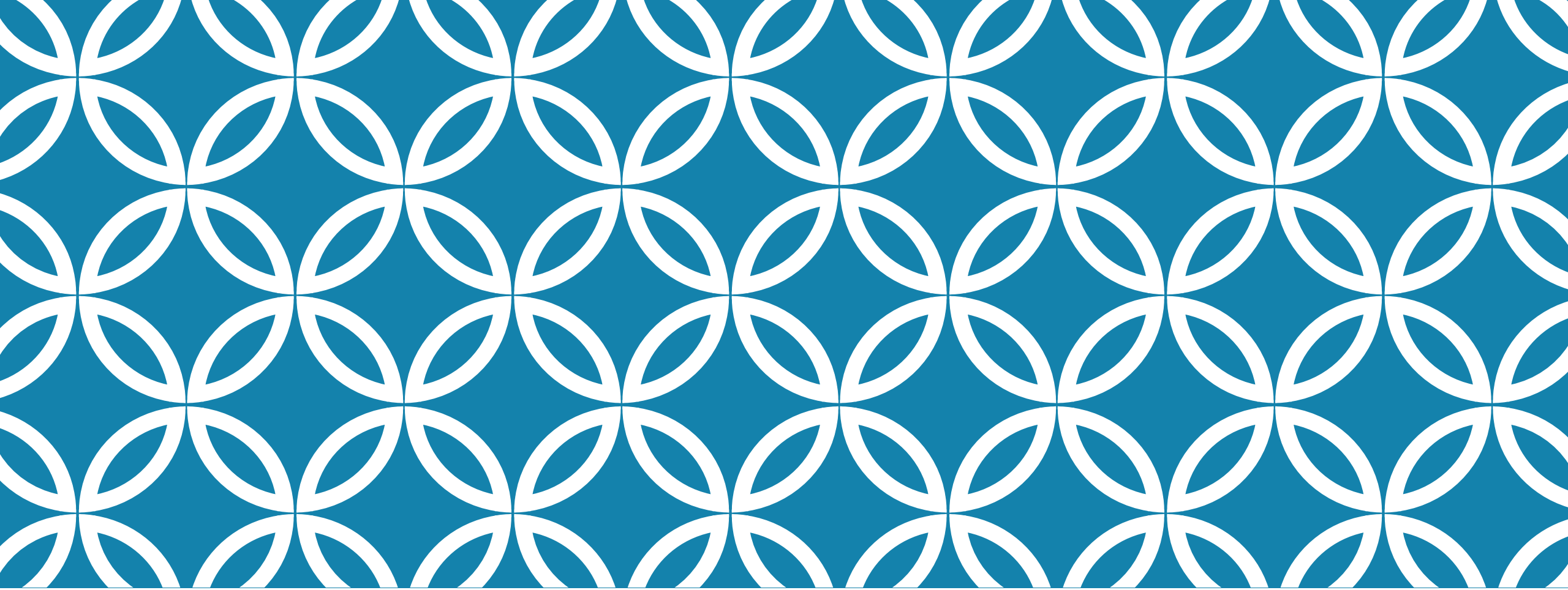
Écrire un programmes permettant d'effectuer des calculs de bases (+,-,/,*)

Le traitement du programme s'effectue dans le main à l'aide de 4 fonctions :

- addition
- soustraction
- multiplication
- division

Faire saisir deux nombres puis exécuter toutes les fonctions à la suite

Faire les exercices des parties I et II



LES BLOCS CONDITIONNELS

LES CONDITIONS - IF

Il nous est possible d'effectuer des test dans notre code. En fonction du résultat de ce test, le programme adapte son traitement.

Pseudo Code

```
Variables : age=25 Entier
Début
        Si age >=18 alors
                Afficher   « Majeur »
        Fin Si
        Si age <18 alors
                Afficher   « Mineur »
        Fin Si
Fin
```

JAVA

```
public static void main(String[] args) {
        int age=25;
        if(age>=18){
                system.out.println("Majeur");
        }
        if(age<18){
                system.out.println( "Mineur");
        }
}
```

LES CONDITIONS - ELSE

Dans notre exemple, le programme n'a que deux options.

Soit l'individu est majeur, soit il est mineur par défaut.

Dans ce cas, il est préférable d'utiliser les instructions IF/ELSE plutôt que deux IF

Pseudo Code

```
Variables : age=25 Entier
Début
        Si age >=18 alors
                Afficher   « Majeur »
        Sinon
                Afficher   « Mineur »
        Fin Si
Fin
```

JAVA

```
public static void main(String[] args) {
        int age=25;
        if(age>=18)
                {system.out("Majeur");}
        else
                {system.out( "Mineur");}
}
```

LES CONDITIONS – ELSE IF

Il nous est possible d'effectuer plusieurs conditions avant celle par défaut grâce à l'instruction ELSE IF

Pseudo Code

```
Variables : age=25 Entier
Début
        Si age =18 alors
                Afficher   « Nouveau majeur !»
        Sinon Si age>18 alors
                Afficher «Majeur»
        Sinon
                Afficher «Mineur»
        Fin Si
Fin
```

JAVA

```
public static void main(String[] args) {
        int age=25;
        if (age ==18 )
                { system.out("Nouveau majeur !")}
        else if(age>18)
                {system.out("Majeur");}
        else
                {system.out( "Mineur");}
}
```

LES OPÉRATEURS

Opérateur	Description
+ - * / ()	Opérateurs classiques, la notion de priorité se fait avec des parenthèses
< ou >	Permet de vérifier si une valeur est inférieur/supérieur/
<= ou >=	Permet de vérifier si une valeur est inférieur ou égale / supérieur ou égale
!= ou == (Attention, un seul = affecte une valeur !)	Permet de vérifier si une valeur est Différent de / Egale à
++	Incrémente de 1 la variable. (Age=17; Age++; => Age vaut 18)
--	décrémente de 1 la variable. (Age=17; Age--; => Age vaut 16)
&&	**et** logique
||	**ou** logique
!	**Non** logique
%	Modulo, retourne le reste d'une division (10%3 retourne 1)

Avec une saisie au clavier

```
If(nom== "Jordan")
{…}

If(nom.equals( "Jordan"))
{…}
```

LES CONDITIONS (SWITCH/TERNAIRE)

La condition if... else que l'on vient de voir est le type de condition le plus souvent utilisé.

En fait, il n'y a pas 36 façons de faire une condition en C ou en Java.

Le if... else permet de gérer tous les cas.

Toutefois, il peut s'avérer quelque peu… répétitif. Nous allons donc découvrir les conditions ternaires ainsi que les switch

Dans le cas ou les tests se font uniquement sur des valeurs fixes, l'utilisation du switch est recommandé.

Dans le cas ou l'on doit affecter à une variable un choix entre deux valeurs, l'utilisation d'une condition ternaire est recommandé.

LES CONDITIONS - SWITCH

On veut effectuer notre traitement sur les valeurs 10,15,18,30 et 90

```
if(age==10){System.out.println("Vous êtes enfant");}
else if(age==15){System.out.println("Vous êtes ado");}
else if(age==18){System.out.println("Vous êtes majeur");}
else if(age==30){System.out.println("Vous êtes adulte");}
else if(age==90){System.out.println("Vous êtes agé");}
else {System.out.println("Aucune catégorie");}
```

```
switch(age)
{
 case 10 :  System.out.println ("Vous êtes enfant");break;
 case 15 :  System.out.println ("Vous êtes ado");break;
 case 18 :  System.out.println ("Vous êtes majeur");break;
 case 30 :  System.out.println ("Vous êtes adulte");break;
 case 90 :  System.out.println ("Vous êtes agé");break;
default :  System.out.println ("Aucune catégorie");break;
}
```

Vous devez mettre une instruction `break;` obligatoirement à la fin de chaque cas. Si vous ne le faites pas, alors l'ordinateur ira lire les instructions en dessous censées être réservées aux autres cas ! L'instruction `break;` commande en fait à l'ordinateur de « sortir » des accolades.

LES CONDITIONS - TERNAIRE

Imaginons qu'on veuille affecter à une variable testMajeur la valeur «MINEUR» si l'age est <18, dans le cas contraire on affecte « MAJEUR».

```
If(age<18)
        { testMajeur="MINEUR";}
else
        { testMajeur=" MAJEUR";}
```

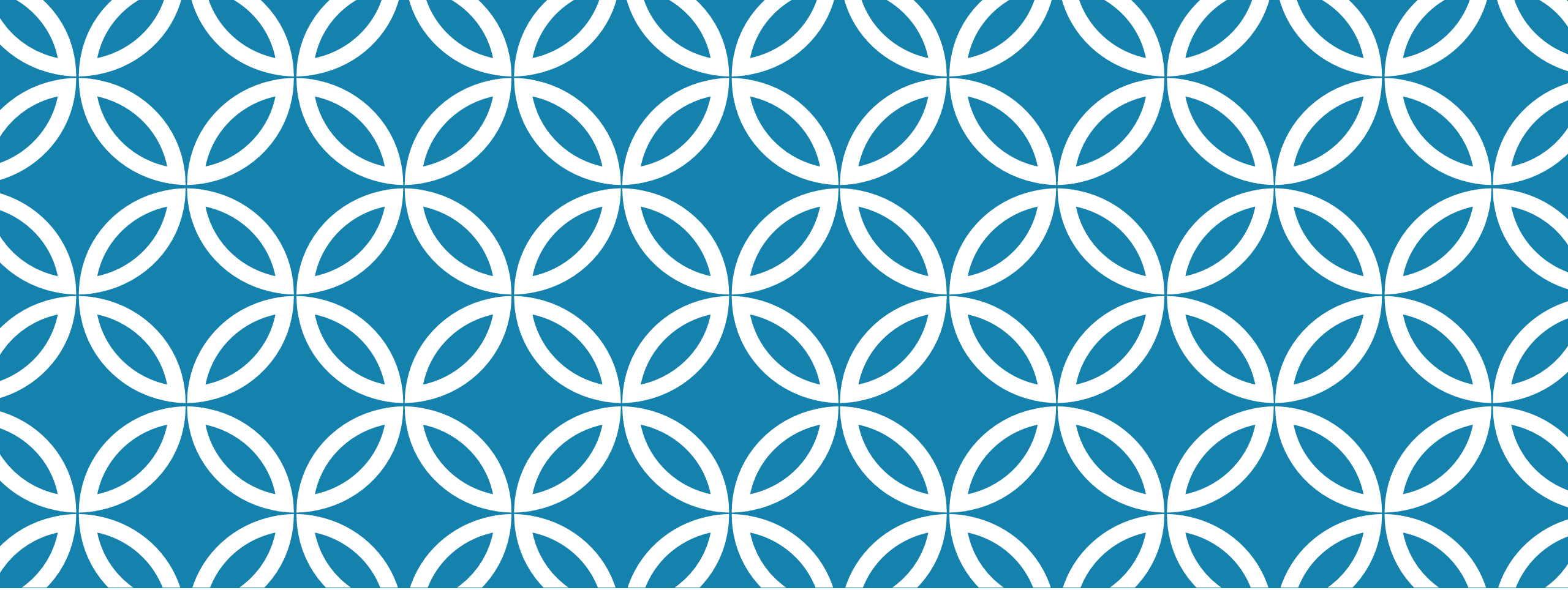
```
testMajeur=(age<18)? "Mineur" : "Majeur";
```

CALCULETTE – ETAPE 2

Reprendre le programme calculette et l'améliorer en faisant également saisir l'opérateur

Le programme ne déclenche qu'une seule fonction

Faire les exercices des parties III et IV



LES BOUCLES

LES BOUCLES - WHILE

Nous avons rencontré un problème avec les conditions lorsque l'âge saisi était négatif. On pourrait demander à l'utilisateur de le saisir de nouveau, mais il faudrait autant de conditions que de possibilités qu'a l'utilisateur de se tromper…

Il nous est donc possible d'effectuer un traitement autant de fois que nécessaire tant que la condition de boucle est vraie avec l'instruction while

Attention : Si la valeur du test reste la même, on risque de créer une boucle infinie

Pseudo Code

```
Tant que age<0
        Afficher  «Saisir Age»
        Saisir age
Fin Tant que
```

JAVA

```
while(age<0){
        system.out("Saisir Age");
        age=sc.nextInt();
}
```

LES BOUCLES – DO WHILE

Il existe une seconde forme de l'instruction WHILE, le DO WHILE.

La différence entre ces deux instructions est l'intervention du test

Dans le cas du WHILE, le test est effectué avant le traitement

Dans le cas du DO WHILE, le programme exécute le traitement, puis verifie la condition

Pseudo Code

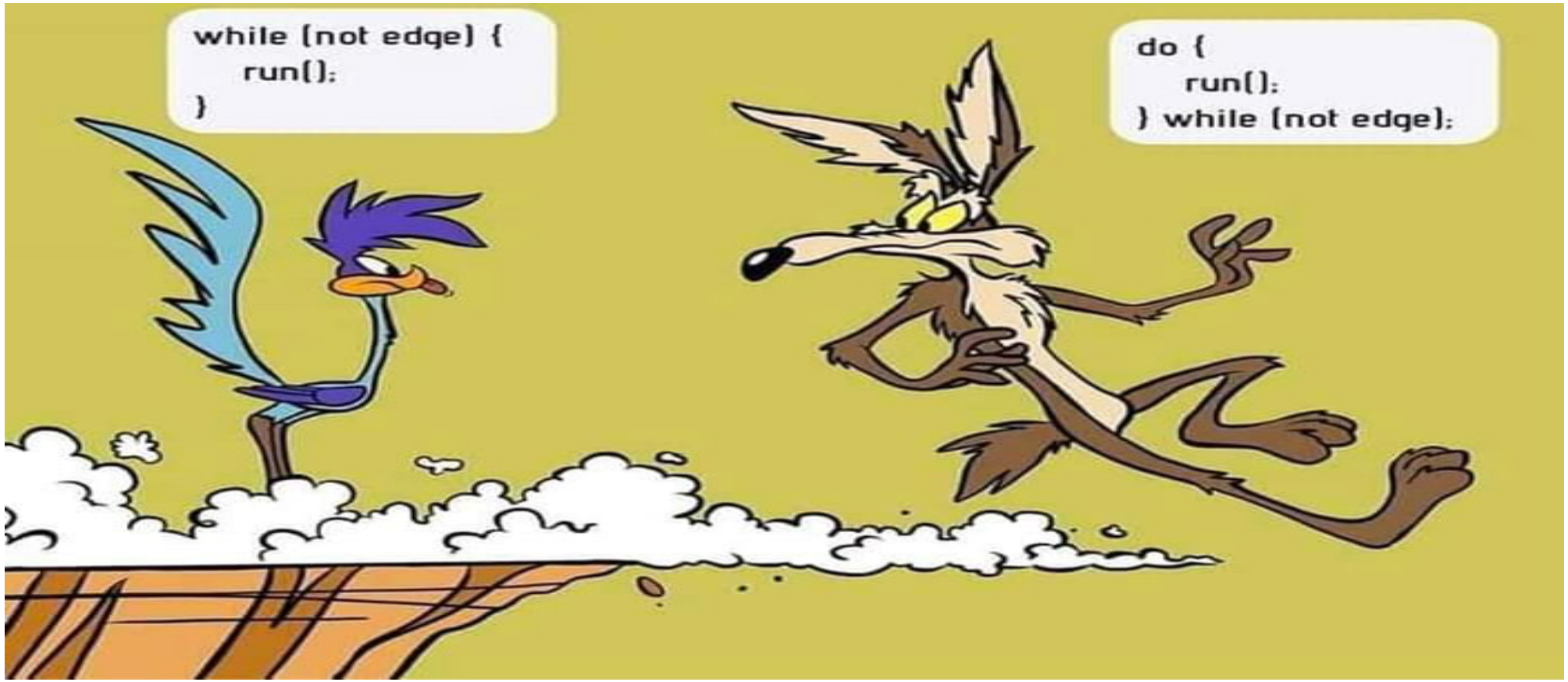
```
Faire
        Afficher  «Saisir Age»
        Saisir age
Tant que age<0
```

JAVA

```
do{
        system.out("Saisir Age");
        age=sc.nextInt();
} while(age<0);
```

LES BOUCLES – DO WHILE / WHILE

Un exemple d'utilisation des deux instructions



LES BOUCLES - FOR

Lorsque l'utilisateur connait à l'avance le nombre de tours de boucle nécessaire, on utilisera la boucle «for» qui est un cas particulier des boucles «while»

La boucle for doit avoir 3 paramètres.

Un indice de départ, un indice max et l'augmentation de l'indice à chaque tour

Voyons l'exemple suivant ou l'on voudrait afficher 10 fois « Bonjour » à l'écran

Pseudo Code

```
Pour i de 1 à 10
        Afficher   «Bonjour n°»+i
        i suivant
Fin Pour
```

JAVA

```
for(int i=1;i<=10;i++)
{
        system.out.println("Bonjour n°"+i);
}
```

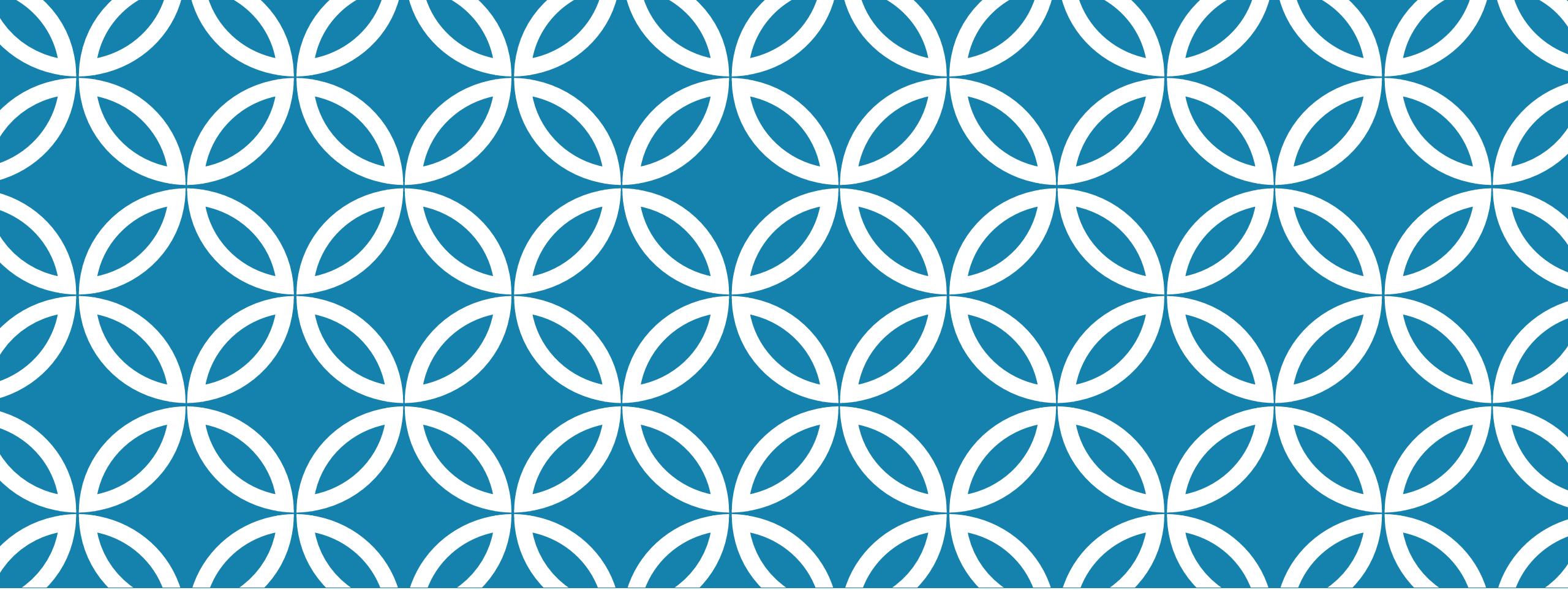
CALCULETTE – ETAPE 3

Reprendre le programme calculette et l'améliorer en proposant à l'utilisateur de continuer son calcul, effacer l'ensemble des calculs ou arrêter le programme

Pour continuer, l'utilisateur doit pouvoir saisir les valeurs suivantes :

- L'un des 4 opérateurs + - / *
- Un « C » pour effacer
- Un « S » pour quitter

Faire les exercices de la partie V



LES TABLEAUX

LES TABLEAUX

Imaginons que dans un programme, nous ayons besoin simultanément de 12 valeurs (par exemple, des notes pour calculer une moyenne). Evidemment, la seule solution dont nous disposons à l'heure actuelle consiste à déclarer douze variables, appelées par exemple Note1, Note2, Note3, etc.

C'est pourquoi la programmation nous permet de rassembler toutes ces variables en une seule, au sein de laquelle chaque valeur sera désignée par un numéro (index). En bon français, cela donnerait donc quelque chose du genre « la note numéro 1 », « la note numéro 2 », « la note numéro 8 ».

Cependant, la plupart des langages ont pour index initial 0.

La note à l'index 2 sera donc la 3e note du tableau

LES TABLEAUX

Pseudo Code

```
Tableau Note[12] en Numerique
Début
    Pour i<– 0 à 11
        Afficher « Saisir la note : », (i+1);
        Saisir Note[i]
    FinPour
Fin
```

JAVA

```
double Note[]=new double[12];
Scanner sc=new Scanner();

for(int i=0;i<=11;i++)
{
      System.out.println ("Saisir la note"+(i+1));
      Note[i]=sc.nextDouble();
}
```

On peut directement déclarer les valeurs d'un tableau :

int tableauEntier[] = {0,1,2,3,4,5,6,7,8,9};

Dans un tableau, la valeur d'un indice doit toujours :
être égale au moins à 0 (dans quelques rares langages, le premier élément d'un tableau porte l'indice 1). Note(6) est le septième élément du tableau Note !
être un nombre entier Quel que soit le langage, l'élément Note(3,1416) n'existe jamais.
être inférieure ou égale au nombre d'éléments du tableau (moins 1, si l'on commence la numérotation à zéro).

EXERCICES

Faire les exercices de la partie VI

Ecrire un programme permettant de trier dans l'ordre croissant un tableau déjà initialisé