Initiation à l'informatique et à l'algorithmique

1. Codage binaire

Conversion de la base 2 à la base 10

Le nombre en base 2 est converti en base 10 en utilisant la somme des puissances de 2.

Exemple :
Nombre binaire : 100101
En base 10 : $1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 37$

Conversion de la base 10 à la base 2

On divise successivement le nombre par 2, en notant les restes jusqu'à obtenir un quotient nul. Les restes, lus de bas en haut, forment le nombre binaire.

Exemple :
Nombre décimal : 37
En base 2 : 100101

Conversion entre bases 2, 8 et 16

- **Base 2 à Base 8 :** Regrouper les bits par paquets de 3 (en partant de la droite) et convertir chaque paquet en son équivalent décimal.
- **Base 2 à Base 16 :** Regrouper les bits par paquets de 4 et convertir chaque paquet en son équivalent hexadécimal.
- **Base 8 ou 16 à Base 2 :** Convertir chaque chiffre octal (ou hexadécimal) en un groupe de 3 (ou 4) bits.

Conversion de la base 10 à la base 8 ou 16

- **Base 8 :** Diviser successivement par 8, puis lire les restes du bas vers le haut.
- **Base 16 :** Diviser successivement par 16 et utiliser les chiffres A, B, C, D, E, F pour les restes de 10 à 15.

Exemple :
Nombre décimal : 93
En base 16 : $5D$

2. Les variables

Une variable est une zone de mémoire réservée pour stocker des valeurs. Chaque variable a un type et un nom.

Type	Signification	Taille (en octets)	Plage de valeurs
char	Caractère UTF-16	2 octets	0 à 65535 (Unicode)
byte	Entier signé de 8 bits	1 octet	-128 à 127
short	Entier signé de 16 bits	2 octets	-32 768 à 32 767
int	Entier signé de 32 bits	4 octets	-2,1E9 à 2,1E9
long	Entier signé de 64 bits	8 octets	-9,2E18 à 9,2E18
float	Nombre à virgule flottante de 32 bits	4 octets	±1.4E-45 à ±3.4E38
double	Nombre à virgule flottante de 64 bits	8 octets	±4.9E-324 à ±1.8E308
boolean	Valeur booléenne (vrai/faux)	1 bit (en pratique 1 octet)	true / false

3. Les Opérateurs

Catégorie	Opérateur	Description
Concaténation	+	Combine des chaînes de caractères : "Hello" + " World" → "Hello World"
Logiques	&& || !	ET logique : vrai si les deux conditions sont vraies OU logique : vrai si au moins une condition est vraie NON logique : inverse la valeur de vérité
Relationnels	== != > < >= <=	Égalité : vrai si les valeurs sont égales Différence : vrai si les valeurs sont différentes Supérieur à : vrai si la première valeur est supérieure Inférieur à : vrai si la première valeur est inférieure Supérieur ou égal à Inférieur ou égal à
Arithmétiques	+ - * / %	Addition Soustraction Multiplication Division Modulo : reste de la division

4. Boucles

Les boucles permettent de répéter une séquence d'instructions un certain nombre de fois ou jusqu'à ce qu'une condition soit remplie.

```
1 // Exemple de boucle for: utile pour repeter un nombre fixe de fois
2 System.out.println("Boucle for :");
3 for (int i = 0; i < 5; i++) {
4     // Le code a executer dans chaque iteration
5 }
6 System.out.println("Iteration for : " + i);
7
```

```java
// Exemple de boucle for imbriquee : utile pour travailler avec des structures multidimensionnelles
System.out.println("\nBoucle for imbriquee :");
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 2; j++) {
        // Le code a executer dans chaque iteration de la boucle imbriquee
    }
    System.out.println("i : " + i + " j : " + j);
}
```

5. Les conditions

Les conditions permettent d'exécuter une portion de code seulement si une condition donnée est remplie (`if`, `else`, `switch`).

```java
// Exemple d'utilisation de if, else if, et else
int age = 20;
// Verifie si l'age est superieur ou egal a 18
if (age >= 18) {
    System.out.println("Vous etes adulte.");
}
// Si age est entre 13 et 17, on considere l'utilisateur comme adolescent
else if (age >= 13) {
    System.out.println("Vous etes adolescent.");
}
// Si aucune condition precedente n'est remplie, on affiche "enfant"
else {
    System.out.println("Vous etes enfant.");
}

// Exemple de boucle while
int compteur = 0;
// Tant que compteur est inferieur a 5, la boucle continue
while (compteur < 5) {
    System.out.println("Compteur (while) : " + compteur);
    compteur++; // Incremente compteur de 1 a chaque iteration
}

// Reinitialise le compteur pour la boucle do-while
compteur = 0;
// Exemple de boucle do-while
do {
    System.out.println("Compteur (do-while) : " + compteur);
    compteur++; // Incremente compteur de 1 a chaque iteration
} while (compteur < 5); // Verifie la condition apres l'execution du bloc
```

6. Les chaînes de caractères (String)

Une chaîne de caractères est une suite de caractères encadrée par des guillemets doubles. Exemple :

```java
// Creation d'un objet Scanner pour lire la saisie clavier
Scanner scanner = new Scanner(System.in);
// Demande a l'utilisateur de saisir par clavier
String S = scanner.nextLine();
// Declaration et initialisation d'une chaine de caracteres
String originalString = "Bonjour, monde!";

// 1. Utilisation de length() pour obtenir la longueur de la chaine
```

```java
int length = originalString.length();
System.out.println("Longueur de la chaine: " + length); // Affiche la longueur de la chaine

// 2. Utilisation de charAt() pour obtenir un caractere a un index donne
char firstChar = originalString.charAt(0); // Le premier caractere
char lastChar = originalString.charAt(length - 1); // Le dernier caractere
System.out.println("Premier caractere: " + firstChar); // Affiche 'B'
System.out.println("Dernier caractere: " + lastChar); // Affiche '!'

// 3. Utilisation de substring() pour obtenir une sous-chaine
String subString = originalString.substring(0, 7); // Obtient "Bonjour"
System.out.println("Sous-chaine: " + subString);

// 4. Utilisation de indexOf() pour trouver l'index d'un caractere
int indexOf0 = originalString.indexOf('o'); // Trouve le premier 'o'
System.out.println("Index du premier 'o': " + indexOf0); // Affiche 1

// 5. Utilisation de equals() pour comparer des chaines
String anotherString = "Bonjour, monde!";
boolean areEqual = originalString.equals(anotherString); // Compare les deux chaines
System.out.println("Les chaines sont egales: " + areEqual); // Affiche true
```

7. Les Tableaux

La déclaration d'un tableau prend cette forme :
<type>[] <nomDuTableau> = <valeur1>, <valeur2>, ..., <valeurN>;

```java
public class ArrayExample {
    public static void main(String[] args) {
        // 1. Declaration et initialisation d'un tableau unidimensionnel
        int[] oneDimensionalArray = {1, 2, 3, 4, 5}; // Initialisation correcte
        // Afficher les elements du tableau unidimensionnel
        System.out.println("Tableau unidimensionnel :");
        for (int i = 0; i < oneDimensionalArray.length; i++) {
            System.out.print(oneDimensionalArray[i] + " "); // Affiche chaque element
        }
        System.out.println(); // Nouvelle ligne apres l'affichage du tableau

        // 2. Declaration et initialisation d'un tableau bidimensionnel (matrice)
        int[][] twoDimensionalArray = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
        // Afficher les elements du tableau bidimensionnel
        System.out.println("\nTableau bidimensionnel :");
        for (int i = 0; i < twoDimensionalArray.length; i++) {
            for (int j = 0; j < twoDimensionalArray[i].length; j++) {
                System.out.print(twoDimensionalArray[i][j] + " "); // Affiche chaque element
            }
            System.out.println(); // Nouvelle ligne apres chaque ligne de la matrice
        }

        // 3. Modification d'un element dans le tableau unidimensionnel
        oneDimensionalArray[2] = 10; // Modifie l'element a l'index 2 (le troisieme element)
        System.out.println("\nTableau unidimensionnel apres modification :");
```

```
        for (int num : oneDimensionalArray) {
            System.out.print(num + " "); // Affiche les elements du tableau modifie
        }

        // 4. Calcul de la somme des elements dans le tableau bidimensionnel
        int sum = 0;
        for (int i = 0; i < twoDimensionalArray.length; i++) {
            for (int j = 0; j < twoDimensionalArray[i].length; j++) {
                sum += twoDimensionalArray[i][j]; // Ajoute chaque element a la somme
            }
        }
        System.out.println("\n\nSomme des elements du tableau bidimensionnel: " + sum);

        // 5. Recherche d'un element dans le tableau unidimensionnel
        int searchElement = 10; // element e rechercher
        boolean found = false;
        for (int i = 0; i < oneDimensionalArray.length; i++) { // Recherche dans le tableau
            if (oneDimensionalArray[i] == searchElement) { // Si trouve
                found = true;
                System.out.println("element " + searchElement + " trouve a l'index " + i);
                break; // Sort de la boucle si l'element est trouve
            }
        }
        if (!found) {
            System.out.println("element " + searchElement + " non trouve dans le tableau.");
        }
    }
}
```

8. Les fichiers texte

Java permet la lecture et l'écriture de fichiers texte en utilisant les classes `BufferedReader`, `BufferedWriter`, etc.

```
// Exemple de lecture et ecriture dans un fichier
import java.io.*;
import java.util.Scanner;

public class FileReadWriteExample {
    public static void main(String[] args) throws IOException {
        // Nom du fichier dans lequel ecrire et lire
        String fileName = "example.txt";

        // 1. Ecriture dans le fichier avec BufferedWriter
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write("Bonjour, ceci est un exemple de fichier texte.");
        writer.newLine(); // Insere une nouvelle ligne
        writer.write("Voici une autre ligne.");
        writer.newLine(); // Insere une autre ligne
        writer.write("Derniere ligne de cet exemple.");
        writer.close(); // Ferme le writer manuellement
        System.out.println("ecriture terminee dans le fichier : " + fileName);

        // 2. Lecture et recherche dans le fichier avec BufferedReader
        System.out.println("\n--- Lecture du fichier et recherche de contenu ---");

```

```
23          // Demande a l'utilisateur d'entrer une chaine a rechercher
24          Scanner scanner = new Scanner(System.in);
25          System.out.print("Entrez le texte a rechercher dans le fichier : ");
26          String searchString = scanner.nextLine();
27          scanner.close(); // Ferme le scanner manuellement
28
29          // Lecture du fichier ligne par ligne
30          BufferedReader reader = new BufferedReader(new FileReader(fileName));
31          String ligneLue;
32          boolean found = false;
33
34          while ((ligneLue = reader.readLine()) != null) {
35              if (ligneLue.contains(searchString)) {
36                  System.out.println("Chaine trouvee : " + ligneLue);
37                  found = true;
38              }
39          }
40          reader.close(); // Ferme le reader manuellement
41
42          if (!found) {
43              System.out.println("La chaine \"" + searchString + "\" n'a pas
                    ete trouvee dans le fichier.");
44          }
45      }
46  }
```

9. Maths en Java

Méthode	Description
Math.abs(x)	Retourne la valeur absolue
Math.sqrt(x)	Retourne la racine carrée
Math.max(x, y)	Retourne le plus grand des deux nombres
Math.round(x)	Arrondit x à l'entier le plus proche
Math.min(x, y)	Retourne le plus petit des deux nombres
Math.random()	Retourne un nombre aléatoire entre 0.0 et 1.0
Math.pow(x, y)	Retourne x élevé à la puissance y
Math.exp(x)	Retourne e^x

10. Actions et fonctions

En Java, on a les actions et les fonctions. Pour faire une tâche, parfois c'est très compliqué, donc on divise le travail en des petits sous-partis rendant le programme plus structuré. Et là, c'est plus facile à comprendre, à maintenir, et à déboguer

- **Actions (méthodes void)** : Elles sont définies avec le mot-clé void, indiquant qu'elles ne retournent pas de valeur.

```
1  // Classe principale
2  public class PairImpair {
3      public static void main(String[] args) {
4          verifierPairImpair(7);  // Appelle la methode avec le nombre 7
5          verifierPairImpair(10); // Appelle la methode avec le nombre 10
6      }
7
```

```
    // Methode void (action) qui affiche si un nombre est pair ou impair
    public static void verifierPairImpair(int nombre) {
        System.out.println(nombre + " est " + (nombre % 2 == 0 ? "pair" : "impair") + ".");
    }
}
```

- **Fonctions (méthodes avec retour)** : retournent une valeur d'un type spécifique (`int`, `double`, `String`, etc.)En plus la méthode doit inclure une instruction return pour fournir le résultat.

```
// Methode principale
public static void main(String[] args) {
    int nombre = 7; // Exemple de nombre a tester
    if (estPair(nombre)) {
        System.out.println(nombre + " est pair.");
    } else {
        System.out.println(nombre + " est impair.");
    }
}

// Fonction qui retourne true si le nombre est pair, sinon false
public static boolean estPair(int nombre) {
    return nombre % 2 == 0;
}
```