Installation Java : *https://www.oracle.com/java/technologies/downloads/#jdk19-windows*

Installation Eclipse : *https://www.eclipse.org/downloads/packages/installer/*

Projet TP1 sous Eclipse :

Créez un nouveau **projet** (`TP1`) sous Eclipse et un **package** pour chaque exercice (`sort`, `babylone`, `dates`).

1. Random & Sort

Copiez et exécutez le programme `Java` ci-dessous, (classe `RandomSort`) qui **crée** un tableau d'entiers, `numbers`, de taille 10, le **remplie** d'une façon *aléatoire* et effectue le **trie** de ce tableau par ordre croissant.

a. La méthode `Arrays.sort(numbers)` trie le tableau par ordre croissant. Cette méthode utilise l'algorithme `Dual-Pivot Quicksort` pour trier le tableau, ce qui est efficace pour les grands tableaux et présente une complexité temporelle moyenne de `O(n log n)`.

b. La méthode `Arrays.toString(numbers)` pour imprimer le tableau. Voici une sortie de ce programme :

```
Original Array: [34, 12, 89, 56, 72, 8, 63, 27, 43, 91]
Sorted Array: [8, 12, 27, 34, 43, 56, 63, 72, 89, 91]
```

```java
import java.util.Arrays;
import java.util.Random;
public class RandomSort {
    public static void main(String[] args) {
        int[] numbers = new int[10];
        Random random = new Random();
        for (int i = 0; i < numbers.length; i++) {
            // generates a random number between 0 and 99
            numbers[i] = random.nextInt(100);
        }
        System.out.println("Original Array: " + Arrays.toString(numbers));
        Arrays.sort(numbers);
        System.out.println("Sorted Array: " + Arrays.toString(numbers));
    }
}
```

2. Algorithme de Babylone

Voici **l'algorithme de Babylone** pour calculer la **racine carrée** d'un nombre positif `n` :

1. Faire une estimation (**`guess`**) de la racine carrée (on peut choisir `n/2` comme estimation initiale).
2. Calculer **`r = n / guess.`**
3. Définir la nouvelle estimation : **`guess = (guess + r) / 2`**.
4. Revenir à l'étape 2 jusqu'à ce que les deux dernières valeurs de l'estimation soient à moins de **`1%`** l'une de l'autre.

a. **Compléter** le programme **Babylonian** qui implémente l'algorithme de Babylone. Ce programme demande juste à l'utilisateur de saisir un nombre positif. Utilisez notamment la méthode Math.abs(guess - r)pour calculer la valeur absolue. Voici une sortie de ce programme : `Square root of 12.2 is 3.49`

```
import java.util.Scanner;
public class Babylonian {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a positive number: ");
        double n = input.nextDouble();  // …… A compléter
```

b. Utilisez la classe **`DecimalFormat`** pour afficher le nombre et sa racine carrée sous un format : **`"#.##"`**.

```
        DecimalFormat df = new DecimalFormat("#.##");
        String formattedGuess = df.format(guess);
```

3. Formatage des dates

Exécutez le programme **`OperationDates`** suivant qui teste la mise en forme d'une date. Nous l'utiliseront dans la suite des TPs pour les dates des opérations bancaires. On utilise ici l'API « date/ heure » introduite dans `Java8`, notamment la méthode `DateTimeFormatter.ofPattern(pattern)` pour créer un formateur avec un motif spécifique, comme ici `"dd/MM/yyyy"`. La sortie de ce programme : `Formatted date: 20/01/2023`

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
public class OperationDates {
 public static void main(String[] args) {
     LocalDate today = LocalDate.now();
     //LocalDate today = LocalDate.of(2023, 1, 29);
     LocalDate nexMonth = today.plusMonths(1);
     DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
     System.out.println("today (formatted): " + today.format(formatter));
     System.out.println("nexMonth (formatted): " + nexMonth.format(formatter));
 }
}
```