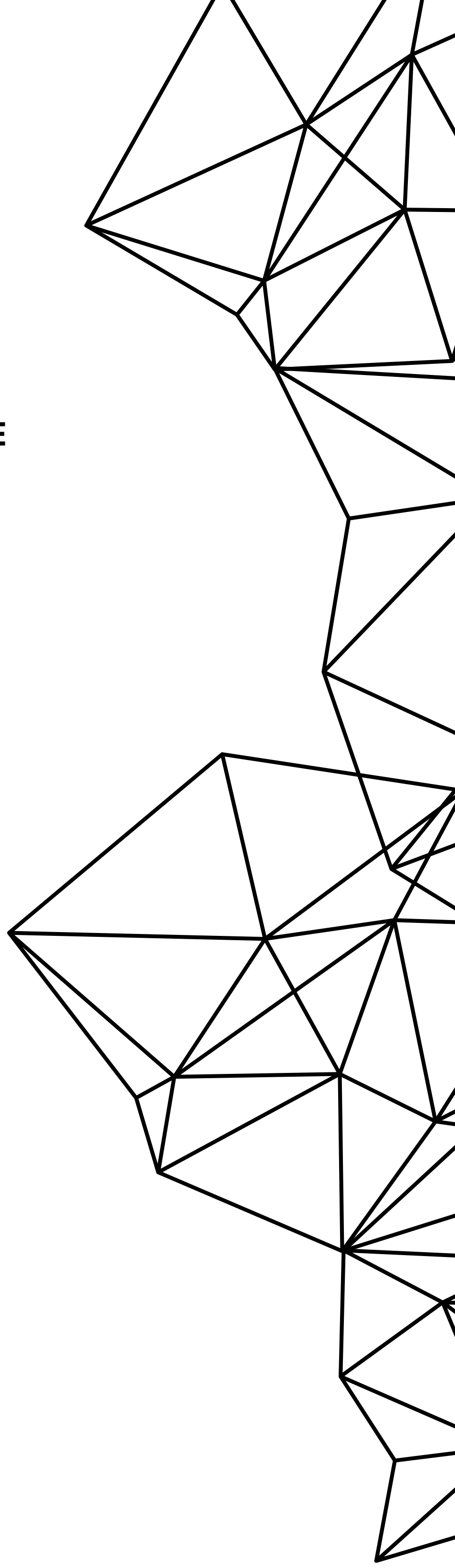
DM ALGO

PETER - THOMAS - AMIROUCHE

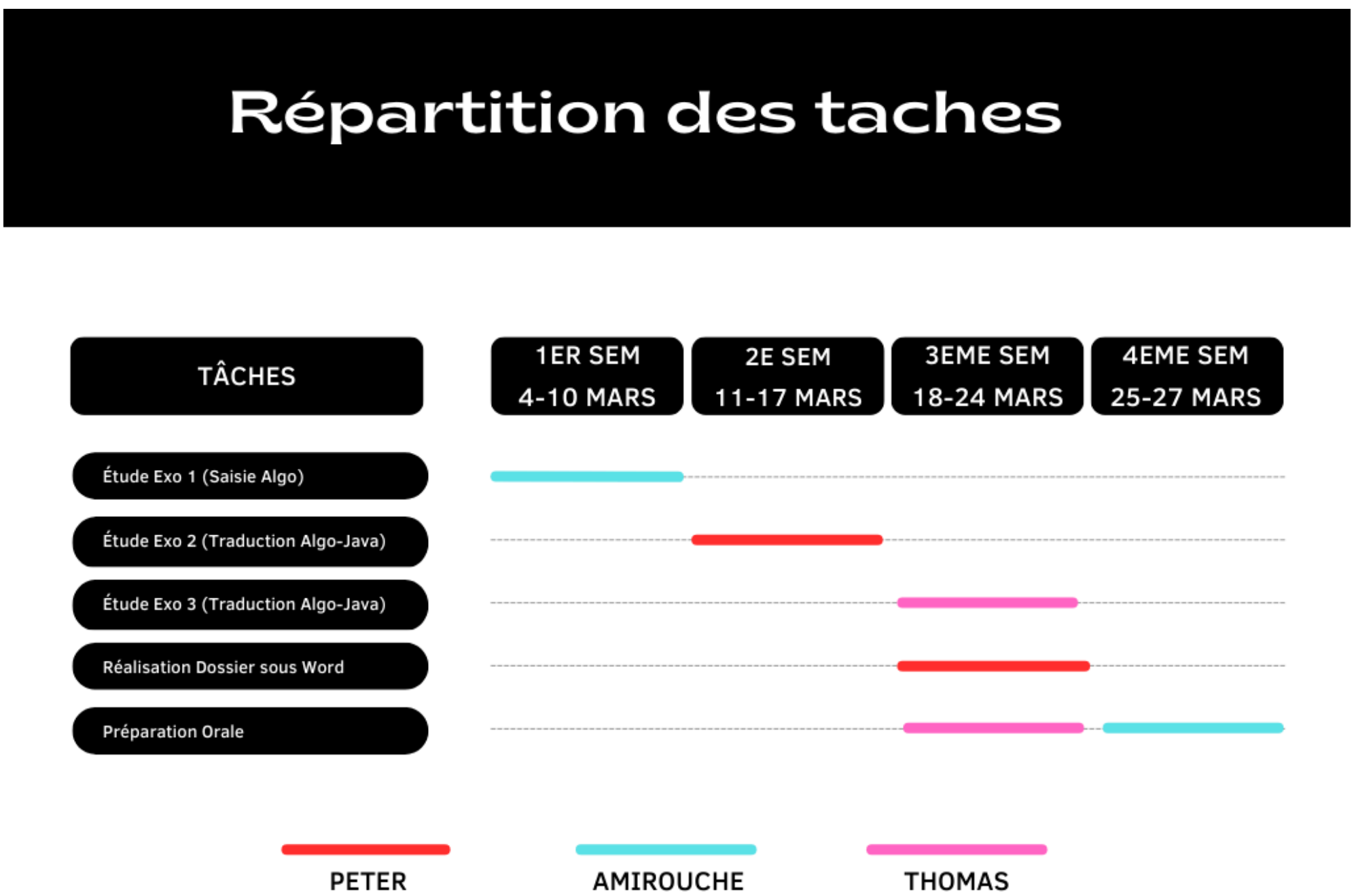


OBJECTIFS A ATTEINDRE:

-METTRE EN MARCHE LE PROGRAMME JAVA CORRESPONDANT À LA CONSIGNE DU JEU

-RESPECTEZ LES RÈGLES MIS EN PLACE SUR LE JEU

REPARTITION DES TACHES:



OUTILS UTILISÉS:

ECLIPSE (JAVA)

VISUAL STUDIO CODE (HTML, JAVASCRIPT,CSS)

FICHE DE COURS (LECONS CHAPITRE)

ALGORITHME MODULE 1 :

```
Algorithme "Distance"

Variable numérique : distance[], villes
|   |   |   |   |   |Rep de type chaîne de caractères

Faire :
    (*Déclaration de la matrice distances*)
    distances = [
        [0, 996, 723, 890, 1286, 305, 564, 245, 1026, 884],
        [0, 0, 750, 104, 286, 711, 576, 747, 505, 543],
        [0, 0, 0, 668, 979, 593, 224, 515, 524, 905],
        [0, 0, 0, 0, 316, 607, 472, 645, 434, 467],
        [0, 0, 0, 0, 0, 890, 769, 938, 750, 400],
        [0, 0, 0, 0, 0, 0, 386, 106, 832, 559],
        [0, 0, 0, 0, 0, 0, 0, 348, 447, 681],
        [0, 0, 0, 0, 0, 0, 0, 0, 799, 665],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 901],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ]

    (*Déclaration du tableau villes*)
    villes = ["Brest", "Grenoble", "Lille", "Lyon", "Marseille", "Nantes", "Paris", "Rennes", "Strasbourg", "Toulouse"]

    Afficher "Voici la liste des villes"
    Pour i allant de 0 à 9:
        Afficher "Ville " + (i + 1) + " : " + villes[i]

    Afficher "Entrez la ville de départ : "
    Saisir villeDepart
    Afficher "Entrez la ville d'arrivée : "
    Saisir villeArrivee

    Initialisation des indices de départ et d'arrivée à -1
    indiceDepart = -1
    indiceArrivee = -1

    Pour i allant de 0 à longueur(villes) - 1:
        Si villes[i] est égal à villeDepart (en ignorant la casse):
            indiceDepart = i
        Si villes[i] est égal à villeArrivee (en ignorant la casse):
            indiceArrivee = i

    Si indiceDepart n'est pas égal à -1 et indiceArrivee n'est pas égal à -1:
        Si indiceDepart < indiceArrivee:
            Afficher "La distance entre " + villeDepart + " et " + villeArrivee + " est de " + distances[indiceDepart][indiceArrivee] + " km"
        Sinon:
            Afficher "La distance entre " + villeDepart + " et " + villeArrivee + " est de " + distances[indiceArrivee][indiceDepart] + " km"
    Sinon:
        Afficher "Les villes saisies ne sont pas valides."

    Rep = Vrai
    Tant que Rep est vrai:
        Afficher "Voulez-vous continuer? (O/N) : "
        Saisir Rep
        Si Rep est égal à "O" ou "o":
            Rep = Vrai
        Sinon:
            Rep = Faux

Close
```

PROGRAMME JAVA MODULE 1 :

```java
package Monpackage;

import java.util.Scanner;

public class Distance {

    public static void main(String[] args) {
        String Rep = "";

        do {
            int i;

            int[][] distances = {
                {0, 996, 723, 890, 1286, 305, 564, 245, 1026, 884},
                {0, 0, 750, 104, 286, 711, 576, 747, 505, 543},
                {0, 0, 0, 668, 979, 593, 224, 515, 524, 905},
                {0, 0, 0, 0, 316, 607, 472, 645, 434, 467},
                {0, 0, 0, 0, 0, 890, 769, 938, 750, 400},
                {0, 0, 0, 0, 0, 0, 386, 106, 832, 559},
                {0, 0, 0, 0, 0, 0, 0, 348, 447, 681},
                {0, 0, 0, 0, 0, 0, 0, 0, 799, 665},
                {0, 0, 0, 0, 0, 0, 0, 0, 0, 901},
                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
            };

            String[] villes = {"Brest", "Grenoble", "Lille", "Lyon", "Marseille", "Nantes", "Paris", "Rennes", "Strasbourg", "Toulouse"};

            System.out.println("Voici la liste des villes");

            for (i = 0; i < 10; i++) {
                System.out.println("Ville " + (i + 1) + " : " + villes[i]);
            }

            System.out.println();
        Scanner scanner = new Scanner(System.in);

        System.out.print("Entrez la ville de départ : ");
        String villeDepart = scanner.nextLine();

        System.out.print("Entrez la ville d'arrivée : ");
        String villeArrivee = scanner.nextLine();

        int indiceDepart = -1;
        int indiceArrivee = -1;

        for (i = 0; i < villes.length; i++) {
            if (villes[i].equalsIgnoreCase(villeDepart)) {
                indiceDepart = i;
            }
            if (villes[i].equalsIgnoreCase(villeArrivee)) {
                indiceArrivee = i;
            }
        }

        if (indiceDepart≠ -1 && indiceArrivee≠ -1) {
            if (indiceDepart < indiceArrivee) {
                System.out.println("La distance entre " + villeDepart + " et " + villeArrivee + " est de " + distances[indiceDepart][indiceArrivee] + " km");
            } else {
                System.out.println("La distance entre " + villeDepart + " et " + villeArrivee + " est de " + distances[indiceArrivee][indiceDepart] + " km");
            }
        } else {
            System.out.println("Les villes saisies ne sont pas valides.");
        }

        System.out.print("Souhaitez-vous recommencez? (Oui/Non) : ");
        Rep = scanner.nextLine();

    } while (!Rep.equalsIgnoreCase("Non"));
  }
}
```

ALGORITHME MODULE 2 :

```
Algorithme "TriSelection"
Variable numériques : chiffre, indexmin, temp
|   |   |   |   |   : Tab[] entier



Début

(* Demander à l'utilisateur d'entrer les valeurs du tableau *)
Afficher "Entrez 5 chiffres supérieurs à 5 pour remplir le tableau"

Pour i allant de 0 à 4:
    Déclarer un booléen `valide` initialisé à faux
    Tant que `valide` est faux:
        Afficher "Chiffre " + (i + 1) + ": "
        `chiffre` = scanner.lireEntier()
        Si `chiffre` > 5:
             `tableau[i]` = `chiffre`
             `valide` = vrai
         Sinon:
             Afficher "Veuillez entrer un chiffre supérieur à 5."

(* Affichage du tableau initial *)
Afficher "Tableau initial"
Appeler afficherTableauAvecSeparateur(tableau, "|")

(* Tri par sélection *)
Pour i allant de 0 à longueur(tableau) - 2:
    `indexmin` = i
    Pour j allant de i + 1 à longueur(tableau) - 1:
        Si `tableau[j]` < `tableau[indexmin]`:
            `indexmin` = j

    `temp` = `tableau[indexmin]`
    `tableau[indexmin]` = `tableau[i]`
    `tableau[i]` = `temp`

    Afficher "Étape " + (i + 1) + " :"
    Appeler afficherTableauAvecSeparateur(tableau, "|")
    Afficher une ligne vide

Afficher "Tableau trié :"
Appeler afficherTableauAvecSeparateur(tableau, "|")

Fermer scanner
```

PROGRAMME JAVA MODULE 2:

```java
import java.util.Scanner;

public class TriSelection {
    public static void main(String[] args) {
        int[] tableau = new int[5];
        Scanner scanner = new Scanner(System.in);

        // Demander à l'utilisateur d'entrer les valeurs du tableau
        System.out.println("Entrez 5 chiffres supérieurs à 5 pour remplir le tableau");

        for (int i = 0; i < 5; i++) {
            boolean valide = false;
            while (!valide) {
                System.out.print("Chiffre " + (i + 1) + ": ");
                int chiffre = scanner.nextInt();
                if (chiffre > 5) {
                    tableau[i] = chiffre;
                    valide = true;
                } else {
                    System.out.println("Veuillez entrer un chiffre supérieur à 5.");
                }
            }
        }

    // Affichage du tableau initial
    System.out.println("Tableau initial");
    afficherTableauAvecSeparateur(tableau, "|");

    // Tri par sélection
    for (int i = 0; i < tableau.length - 1; i++) {
        int indexmin = i;
        for (int j = i + 1; j < tableau.length; j++) {
            if (tableau[j] < tableau[indexmin]) {
                indexmin = j;
            }
        }
          int temp = tableau[indexmin];
          tableau[indexmin] = tableau[i];
          tableau[i] = temp;

          System.out.println("Étape " + (i + 1) + ":");
          afficherTableauAvecSeparateur(tableau, "|");
          System.out.println();
     }

     // Affichage du tableau trié
     System.out.println("Tableau trié:");
     afficherTableauAvecSeparateur(tableau, "|");

     scanner.close();
}

    public static void afficherTableauAvecSeparateur(int[] tableau, String separateur) {
        for (int i = 0; i < tableau.length; i++) {
            System.out.print(tableau[i]);
            if (i < tableau.length - 1) {
                System.out.print(separateur);
            }
        }
        System.out.println();
    }
}
```

ALGORITHME MODULE 3 :

```
(*Remplissage du tableau à trier*)
Pour I allant de 1 à N
 Afficher "Veuillez saisir le contenu de la case de rang", I
 Saisir Tab[I]
FPour

(*Sauvegarde de la valeur de N pour la réutiliser après le tri*)
Sauve ⟵ N

 (*Traitement*)
 TantQue N>0
         I⟵1

         Fin⟵0
         TantQue I<N

             Si Tab[I]>Tab[I+1]
             Alors Ech⟵Tab[I]
                   Tab[I]⟵Tab[I+1]
                   Tab[I+1]⟵Ech

             FSi

         FTantQue
         N⟵ N-1
FTantQue

(*Affichage du tableau trié*)
Pour I allant de 1 à Sauve
    Afficher Tab[I]
FPour
```

PROGRAMME JAVA MODULE 3 :

```
 // Initialisation du tableau avec la taille saisie
 tab = new int[n];

 // Remplissage du tableau à trier
 for (i = 0; i < n; i++) {
     System.out.print("Veuillez saisir le contenu de la case de rang " + i + " : ");
     tab[i] = scanner.nextInt();
 }

  // Sauvegarde de la valeur de n pour la réutiliser après le tri
  sauve = n;

  // Traitement
  while (n > 1) {
      fin = n;
      for (i = 0; i < fin - 1; i++) {
          if (tab[i] > tab[i + 1]) { // Condition modifiée pour trier du plus petit au plus grand
              ech = tab[i];
              tab[i] = tab[i + 1];
              tab[i + 1] = ech;
          }
         }
     }
     n = n - 1;
}
// Affichage du tableau trié
System.out.println("Tableau trié : ");
for (i = 0; i < sauve; i++) {
    System.out.print(tab[i] + " ");
}
// Fermeture du scanner
scanner.close();
```

DIFFICULTÉS RENCONTRÉS :

-RESPECT DU TABLEAU TRIÉ

-RESPECTEZ LES RÈGLES SPÉCIFIQUES MIS EN PLACE SUR LE JEU

-COORDINATION DES TROIS EXERCICES

CONCLUSION :

EN CONCLUSION, CE PROJET JAVA A ÉTÉ UNE OPPORTUNITÉ PASSIONNANTE D'EXPLORER LES CONCEPTS FONDAMENTAUX DE LA PROGRAMMATION ET DE LES APPLIQUER À LA CRÉATION D'UN TABLEAU TRIÉ DE NOMBRES. EN UTILISANT DES ALGORITHMES DE TRI EFFICACES, NOUS AVONS PU CONCEVOIR UNE SOLUTION ROBUSTE QUI ORGANISE LES CHIFFRES DANS L'ORDRE CROISSANT, FACILITANT AINSI LEUR MANIPULATION ET LEUR UTILISATION ULTÉRIEURE. EN PLUS DE DÉVELOPPER NOS COMPÉTENCES EN JAVA, CE PROJET NOUS A ÉGALEMENT PERMIS DE COMPRENDRE L'IMPORTANCE DES STRUCTURES DE DONNÉES EFFICACES DANS LE DÉVELOPPEMENT LOGICIEL. EN CONTINUANT À PERFECTIONNER NOS COMPÉTENCES EN PROGRAMMATION, NOUS SOMMES MIEUX ÉQUIPÉS POUR RELEVER LES DÉFIS FUTURS ET CRÉER DES SOLUTIONS INNOVANTES.