Laboratoire 8

DURÉE PRÉVUE : 8H00

OBJECTIFS

Au terme de ce laboratoire, l'étudiant sera capable de :

- → déclarer un tableau à deux dimensions
- → manipuler un tableau à deux dimensions quel que soit le type de ses éléments (*int*, *String*, ...)
- → transmettre un tableau à deux dimensions à une fonction

EXERCICE 1 : MOTIFS

Ce premier exercice a pour but de vous familiariser avec les tableaux à deux dimensions.

PRÉPARATION

1. Dans le package **util**, ajoutez une classe nommée **Tableau2D**
2. Dans la classe **Tableau2D**, déclarez une fonction nommée **afficher** qui permet d'afficher ligne par ligne les valeurs d'un tableau de caractères à deux dimensions :

   ```
   public static void afficher(char[][] t)
   ```

   Par exemple, l'affichage du tableau `[[a, b, c], [d, e, f]]` donne :

   ```
   a b c
   d e f
   ```

PROGRAMME PRINCIPAL

1. Ajoutez un package nommé **labo8** au répertoire **src** du projet **prb**
2. Dans le package **labo8**, ajoutez une classe nommée **Motifs** contenant une fonction **main**
3. Dans la classe **Motifs**, déclarez :

   → une fonction nommée **creerMotifCarre** qui permet d'obtenir un tableau de caractères à deux dimensions contenant des astérisques formant un carré :

   ```
   private static char[][] creerMotifCarre(int taille)
   ```

   Le paramètre `taille` indique le nombre de lignes et de colonnes du tableau à créer

   Exécutez cette fonction à partir de la fonction **main**, puis affichez le tableau obtenu. Si la taille est 7, l'affichage donne :

   ```
   * * * * * * *
   * * * * * * *
   * * * * * * *
   * * * * * * *
   * * * * * * *
   * * * * * * *
   * * * * * * *
   ```

→ une fonction nommée **creerMotifTriangles** qui permet d'obtenir un tableau de caractères à deux dimensions contenant des astérisques et des points formant deux triangles :

```
private static char[][] creerMotifTriangles(int taille)
```

Le paramètre `taille` indique le nombre de lignes et de colonnes du tableau à créer

Exécutez cette fonction à partir de la fonction **main**, puis affichez le tableau obtenu. Si la taille est 7, l'affichage donne :

```
* * * * * * *
. * * * * * *
. . * * * * *
. . . * * * *
. . . . * * *
. . . . . * *
. . . . . . *
```

→ une fonction nommée **creerMotifX** qui permet d'obtenir un tableau de caractères à deux dimensions contenant des astérisques et des points formant un X :

```
private static char[][] creerMotifX(int taille)
```

Le paramètre `taille` indique le nombre de lignes et de colonnes du tableau à créer

Exécutez cette fonction à partir de la fonction **main**, puis affichez le tableau obtenu. Si la taille est 7, l'affichage donne :

```
* . . . . . *
. * . . . * .
. . * . * . .
. . . * . . .
. . * . * . .
. * . . . * .
* . . . . . *
```

4. **[Facultatif - Difficulté élevée]** Dans la classe **Motifs**, déclarez :

→ une fonction nommée **creerMotifPapillon** qui permet d'obtenir un tableau de caractères à deux dimensions contenant des astérisques et des points formant un papillon :

```
private static char[][] creerMotifPapillon(int taille)
```

Le paramètre `taille` indique le nombre de lignes et de colonnes du tableau à créer

Exécutez cette fonction à partir de la fonction **main**, puis affichez le tableau obtenu. Si la taille est 7, l'affichage donne :

```
* . . . . . *
* * . . . * *
* * * . * * *
* * * * * * *
* * * . * * *
* * . . . * *
* . . . . . *
```

→ une fonction nommée **creerMotifLosange** qui permet d'obtenir un tableau de caractères à deux dimensions contenant des astérisques et des points formant un losange :

```
private static char[][] creerMotifLosange(int taille)
```

Le paramètre `taille` indique le nombre de lignes et de colonnes du tableau à créer

Exécutez cette fonction à partir de la fonction **main**, puis affichez le tableau obtenu. Si la taille est 7, l'affichage donne :

```
. . . * . . .
. . * * * . .
. * * * * * .
* * * * * * *
. * * * * * .
. . * * * . .
. . . * . . .
```

EXERCICE 2 : COMPÉTITION DE FOOTBALL

Les organisateurs d'une compétition de football souhaitent un programme capable de générer le planning des rencontres. La compétition est organisée en un tour (les équipes ne se rencontrent qu'une fois lors de la compétition) et est subdivisée en plusieurs journées (chaque équipe ne joue qu'une rencontre par journée).

Par exemple, si quatre équipes A, B, C et D participent à la compétition, le planning des journées est le suivant :

Journée 1	Journée 2	Journée 3
A - C	A - B	A - D
D - B	C - D	B - C

LOGIQUE DE RÉSOLUTION

Considérons six équipes nommées A, B, C, D, E et F.

Ces noms sont placés dans un tableau :

A	B	C	D	E	F

Pour obtenir les rencontres de chaque journée de la compétition, il faut répéter les étapes suivantes :

1. Décaler tous les éléments du tableau d'une position vers la droite
2. Permuter les deux premiers éléments
3. Parcourir les éléments positionnés symétriquement dans le tableau pour afficher les rencontres de la journée (l'équipe en 0 rencontre l'équipe en 5, l'équipe en 1 rencontre l'équipe en 4, l'équipe en 2 rencontre l'équipe en 3)

Voici une illustration de ces étapes pour les deux premières journées :

1. Décalage vers la droite

   Avant :

A	B	C	D	E	F

   Les éléments en vert doivent être décalés d'une position vers la droite. Le dernier élément (ici, F) sera placé en début de tableau.

   Pour opérer ce décalage, le plus simple est de sauvegarder le dernier élément dans une variable, puis de copier chaque élément à la position suivante en parcourant le tableau **de la droite vers la gauche**, et enfin de placer l'élément sauvegardé à la première position.

   Après :

F	A	B	C	D	E

2. Permutation des deux premiers éléments

A	F	B	C	D	E

3. Affichage des rencontres de la journée

A	F	B	C	D	E

Les rencontres de la 1ère journée opposent A à E, F à D et B à C.

4. Décalage vers la droite

Avant :

A	F	B	C	D	E

Après :

E	A	F	B	C	D

5. Permutation des deux premiers éléments

A	E	F	B	C	D

6. Affichage des rencontres de la journée

A	E	F	B	C	D

Les rencontres de la 2e journée opposent A à D, E à C et F à B.

7. …

PRÉPARATION

1. Dans la classe **TableauChaines** du package **util**, déclarez une fonction nommée **contient** qui permet de déterminer si un tableau de chaînes de caractères contient une chaîne donnée :

```
public static boolean contient(String[] t, String chaine)
```

2. Dans le package **labo8**, ajoutez une classe nommée **CompetitionSportive**
3. Dans la classe **CompetitionSportive**, déclarez :

→ une fonction nommée **encoderNomsEquipes** qui permet d'obtenir un tableau contenant les noms des équipes saisis par l'utilisateur :

```
static String[] encoderNomsEquipes(int nbEquipes)
```

Le paramètre `nbEquipes` indique le nombre de noms d'équipes que l'utilisateur doit encoder

Répétez l'acquisition d'un nom d'équipe si celui-ci est une chaîne de caractères vide ou s'il a déjà été encodé. Utilisez la fonction *TableauChaines.contient*

→ une fonction nommée **decalerADroite** qui permet de décaler les éléments d'un tableau de chaînes de caractères d'une case vers la droite (le dernier élément est placé en début de tableau) :

```
static void decalerADroite(String[] nomsEquipes)
```

→ une fonction nommée **journeeSuivante** qui permet d'obtenir un tableau à deux dimensions contenant les rencontres de la prochaine journée de la compétition :

```
static String[][] journeeSuivante(String[] nomsEquipes)
```

Cette fonction effectue une itération des 3 étapes de la logique expliquée précédemment. Pour ce faire, utilisez les fonctions *decalerADroite* et *TableauChaines.permuter*

Le nombre de lignes du nouveau tableau équivaut aux nombres de rencontres qui ont lieu dans une journée. Par exemple, avec 6 équipes, il y a 3 rencontres par journée

Chaque ligne du nouveau tableau contient les noms des deux équipes qui se rencontrent. Par exemple, avec six équipes nommées A, B, C, D, E et F :

	0	1
0	"A"	"E"
1	"F"	"D"
2	"B"	"C"

→ une fonction nommée **rencontreToString** qui permet d'obtenir une chaîne de caractères représentant une rencontre :

```
static String rencontreToString(String[] rencontre)
```

Le paramètre `rencontre` est la référence d'un tableau contenant les noms des deux équipes qui se rencontrent

Les noms des équipes sont séparés par un tiret et un saut de ligne est placé en fin de chaîne. Par exemple,

```
Aische - RRC Mormont
```

→ une fonction nommée **journeeToString** qui permet d'obtenir une chaîne de caractères représentant toutes les rencontres d'une journée de la compétition :

```
static String journeeToString(String[][] journee)
```

Le paramètre `journee` est la référence d'un tableau contenant les rencontres de la journée. Utilisez ici la fonction *rencontreToString*

Par exemple,

```
Aische - RRC Mormont
Tilleur - Richelle
FC Huy - RES Durbuy
```

→ une fonction nommée **getRencontre** qui permet de récupérer la rencontre que doit jouer une équipe lors d'une journée de la compétition :

```
static String[] getRencontre(String[][] journee, String nomEquipe)
```

Le paramètre `journee` est la référence d'un tableau contenant les rencontres de la journée. Le paramètre `nomEquipe` est le nom de l'équipe recherchée

La fonction retourne **`null`** si le nom de l'équipe ne correspond à aucun des noms trouvés

[Suggestion] Utilisez la fonction *TableauChaines.contient*

→ une fonction **choisirEquipe** qui permet à l'utilisateur de sélectionner une équipe parmi celles qui participent à la compétition :

```
static int choisirEquipe(String[] nomsEquipes)
```

Le paramètre `nomsEquipes` est la référence d'un tableau contenant les noms des équipes qui participent à la compétition

La fonction retourne l'indice de l'équipe sélectionnée

La sélection se fait au moyen d'un menu de la forme :

```
1. Aische
2. FC Huy
3. RES Durbuy
4. Richelle
5. RRC Mormont
6. Tilleur
Choix ?
```

TEST UNITAIRE

Avant d'utiliser les fonctions précédemment définies, appliquez des tests à trois d'entre elles : **decalerADroite**, **journeeSuivante** et **getRencontre**.

1. Dans le répertoire **test** du projet **prb**, ajoutez un package nommé **labo8**
2. Dans le package **labo8** du répertoire **test**, ajoutez une classe de test nommée **CompetitionSportiveTest**
3. Déclarez des fonctions de test permettant de valider le fonctionnement des trois fonctions précitées

[Remarque] La comparaison de tableaux à deux dimensions se fait de la même manière qu'avec les tableaux à une dimension, c'est-à-dire avec l'assertion *assertArrayEquals*

Par exemple,

```
@Test
publicvoid testJourneeSuivante4EquipesJ1() {
    String[][] expected = { { "A", "C" }, { "D", "B" } };
    String[] nomsEquipes = { "A", "B", "C", "D" };
    assertArrayEquals(expected, CompetitionSportive.journeeSuivante(nomsEquipes));
}
```

PROGRAMME PRINCIPAL

1. Créez une classe nommée **CompetitionDeFootball** dans le package **labo8**
2. Déclarez dans cette classe une fonction **main**
3. Dans cette fonction, réalisez le programme tel qu'il est montré dans l'exemple d'exécution ci-dessous. Pour ce faire, utilisez les fonctions précédemment déclarées dans la classe **CompetitionSportive**

   Lorsque l'utilisateur choisit l'option 1, assurez-vous que le nombre d'équipes est un nombre pair compris inclusivement entre 2 et 20

   Lorsque l'utilisateur choisit l'option 2 ou 3, assurez-vous que les noms d'équipes sont déjà encodés. Si ce n'est pas le cas, affichez le message *"Vous devez d'abord encoder les équipes !"*

   Si le choix fait par l'utilisateur ne correspond à aucune des options proposées dans le menu, affichez le message *"Choix incorrect !"*

EXEMPLE D'EXÉCUTION

Les données saisies par l'utilisateur sont représentées en vert.

```
Compétition de football
1. Encoder les équipes
2. Afficher le calendrier de la compétition
3. Afficher le calendrier d'une équipe
4. Quitter
Choix ? 1

Nombres d'équipes ? 6
Equipe 1 ? Aische
Equipe 2 ? FC Huy
Equipe 3 ? RES Durbuy
Equipe 4 ? Richelle
Equipe 5 ? RRC Mormont
Equipe 6 ? Tilleur

Compétition de football
1. Encoder les équipes
2. Afficher le calendrier de la compétition
3. Afficher le calendrier d'une équipe
4. Quitter
Choix ? 2

Journée 1
Aische - RRC Mormont
Tilleur - Richelle
FC Huy - RES Durbuy

Journée 2
Aische - Richelle
RRC Mormont - RES Durbuy
Tilleur - FC Huy

Journée 3
Aische - RES Durbuy
Richelle - FC Huy
RRC Mormont - Tilleur
```

```
Journée 4
Aische - FC Huy
RES Durbuy - Tilleur
Richelle - RRC Mormont

Journée 5
Aische - Tilleur
FC Huy - RRC Mormont
RES Durbuy - Richelle

Compétition de football
1. Encoder les équipes
2. Afficher le calendrier de la compétition
3. Afficher le calendrier d'une équipe
4. Quitter
Choix ? 3

1. Aische
2. FC Huy
3. RES Durbuy
4. Richelle
5. RRC Mormont
6. Tilleur
Choix ? 5

Journée 1
Aische - RRC Mormont

Journée 2
RRC Mormont - RES Durbuy

Journée 3
RRC Mormont - Tilleur

Journée 4
Richelle - RRC Mormont

Journée 5
FC Huy - RRC Mormont

Compétition de football
1. Encoder les équipes
2. Afficher le calendrier de la compétition
3. Afficher le calendrier d'une équipe
4. Quitter
Choix ? 4

Fin du programme.
```

EXERCICE 3 : PRÉVISIONS MÉTÉO

Des météorologues belges souhaitent un programme leur facilitant l'analyse des températures prévisionnelles relatives à différents endroits du pays pour les cinq jours à venir.

Par exemple, le tableau suivant recense les températures prévisionnelles exprimées en °C pour quatre localités :

	MER 30	JEU 1	VEN 2	SAM 3	DIM 4
Bruxelles	4	6	8	6	4
Eupen	3	2	5	3	2
Liège	4	5	7	5	4
Ostende	5	7	9	7	4

L'analyse des températures prévisionnelles consiste en la détermination des résultats suivants :

- → la localité où il fera en moyenne le plus chaud lors des 5 prochains jours
- → la localité où il fera en moyenne le plus froid lors des 5 prochains jours
- → le jour où il fera en moyenne le plus chaud
- → le jour où il fera en moyenne le plus froid

Il doit être également possible d'ajouter à tout moment une localité et les températures prévisionnelles correspondantes.

AVANT-PROPOS

Afin de valider vos fonctions au fur et à mesure du développement, il est conseillé de réaliser systématiquement des tests unitaires.

PRÉPARATION

1. Téléchargez la classe **Date**, puis importez-la dans le projet **prb** en la plaçant dans le package **util** du répertoire **src**
2. Dans la classe **TableauChaines** du package **util**, déclarez une fonction nommée **ajouterElement** qui permet d'obtenir un tableau de chaînes de caractères qui est la copie d'un autre tableau augmentée d'un nouvel élément :

   ```
   public static String[] ajouterElement(String[] t, String element)
   ```

   Par exemple, le tableau `[Bruxelles, Eupen, Liège]` augmenté de la chaîne `Ostende` donne le tableau `[Bruxelles, Eupen, Liège, Ostende]`

   Utilisez la fonction *System.arraycopy*

3. Dans le package **util**, ajoutez une classe nommée **TableauReels**

4. Dans la classe **TableauReels**, déclarez :

   → une fonction nommée **minimum** qui permet d'obtenir la position de la plus petite valeur dans un tableau de réels :

   ```
   public static int minimum(double[] t)
   ```

   Par exemple, la position du minimum dans le tableau `[4.6, 13.0, 7.5, 2.8, 3.6]` est 3

   Si le tableau est vide, la fonction retourne `-1`

   → une fonction nommée **maximum** qui permet d'obtenir la position de la plus grande valeur dans un tableau de réels :

   ```
   public static int maximum(double[] t)
   ```

   Par exemple, la position du maximum dans le tableau `[4.6, 13.0, 7.5, 2.8, 3.6]` est `1`

   Si le tableau est vide, la fonction retourne `-1`

   → une fonction nommée **moyenne** qui permet d'obtenir la moyenne des valeurs dans un tableau de réels :

   ```
   public static double moyenne(double[] t)
   ```

   Par exemple, la moyenne des éléments du tableau `[4.6, 13.0, 7.5, 2.8, 3.6]` est `6.3`

   Si le tableau est vide, la fonction retourne `Double.NaN`

5. Dans la classe **Tableau2D** du package **util**, déclarez

   → une fonction nommée **ajouterLigne** qui permet d'obtenir un tableau de réels qui est la copie d'un autre tableau augmentée d'une nouvelle ligne :

   ```
   public static double[][] ajouterLigne(double[][] t, double[] ligne)
   ```

   Par exemple, le tableau `[[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]]` augmenté de la ligne `[7.0, 8.0, 9.0]` donne le tableau `[[1.0, 2.0, 3.0], [4.0, 5.0, 6.0], [7.0, 8.0, 9.0]]`

   Utilisez la fonction *System.arraycopy*

   [Astuce] Le nouveau tableau doit pouvoir contenir les références des lignes du tableau `t` et de `ligne`. Une fois créé, il suffit d'y copier les références précitées

   → une fonction nommée **moyennesParLigne** qui permet d'obtenir un tableau de réels contenant les moyennes des lignes d'un tableau à deux dimensions :

   ```
   public static double[] moyennesParLigne(double[][] t)
   ```

   Par exemple, les moyennes des lignes du tableau `[[7.0, 2.5, 4.0], [1.0, 8.0, 6.0]]` sont `[4.5, 5.0]`

   Utilisez la fonction *TableauReels.moyenne*

Si le tableau `t` ne contient aucune ligne, la fonction retourne un tableau vide

→ une fonction nommée **extraireColonne** qui permet d'obtenir un tableau de réels contenant les éléments de l'une des colonnes d'un tableau à deux dimensions :

```
public static double[] extraireColonne(double[][] t, int j)
```

Le paramètre `j` est l'indice de la colonne à extraire

Par exemple, l'extraction de la colonne d'indice 2 du tableau `[[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]]` donne le tableau `[3.0, 6.0]`

→ une fonction nommée **moyennesParColonne** qui permet d'obtenir un tableau de réels contenant les moyennes des colonnes d'un tableau à deux dimensions :

```
public static double[] moyennesParColonne(double[][] t)
```

Par exemple, les moyennes des colonnes du tableau `[[7.0, 2.5, 4.0], [1.0, 8.0, 6.0]]` sont `[4.0, 5.25, 5.0]`

Utilisez les fonctions *extraireColonne* et *TableauReels.moyenne*

Si le tableau `t` ne contient aucune ligne ou aucune colonne, la fonction retourne un tableau vide

PROGRAMME PRINCIPAL

1. Créez une classe nommée **PrevisionsMeteo** dans le package **labo8**
2. Déclarez dans cette classe une fonction **main**
3. Dans cette fonction, réalisez le programme tel qu'il est montré dans l'exemple d'exécution de la page suivante. Pour ce faire, utilisez les fonctions précédemment déclarées dans les classes **Date**, **TableauChaines**, **TableauReels** et **Tableau2D**. Vous pouvez également déclarez d'autres fonctions pour vous faciliter le codage

   Déclarez une constante nommée `NB_JOURS` indiquant le nombre de jours sur lesquels portent les prévisions (dans ce cas, 5 jours). Faites en sorte que votre programme s'adapte selon la valeur de cette constante

   Afin de pouvoir ajouter ultérieurement une première ligne aux tableaux destinés à contenir les noms des localités et les températures prévisionnelles, ceux-ci doivent être initialement vides. Par exemple,

   ```
   String[] nomsLocalites = new String[0];
   double[][] temperatures = new double[0][];
   ```

   Lorsque l'utilisateur choisit l'option 1, le nom de la nouvelle localité est ajouté au tableau `nomsLocalites` grâce à la fonction *TableauChaines.ajouterElement* et la ligne des températures correspondantes est ajoutée au tableau `temperatures` grâce à la fonction *Tableau2D.ajouterLigne*

   Lorsque l'utilisateur choisit l'option 3, assurez-vous qu'au moins une localité est déjà encodée. Si ce n'est pas le cas, affichez le message *"Aucune localité n'a été encodée !"*

Si le choix fait par l'utilisateur ne correspond à aucune des options proposées dans le menu, affichez le message *"Choix incorrect !"*

EXEMPLE D'EXÉCUTION

Les données saisies par l'utilisateur sont représentées en vert.

```
Prévisions météo
1. Ajouter une localité et ses prévisions
2. Afficher les prévisions
3. Analyser les prévisions
4. Quitter
Choix ? 1

Nom de la localité ? Bruxelles
Mercredi 30 ? 4
Jeudi 1 ? 6
Vendredi 2 ? 8
Samedi 3 ? 6
Dimanche 4 ? 4

Prévisions météo
1. Ajouter une localité et ses prévisions
2. Afficher les prévisions
3. Analyser les prévisions
4. Quitter
Choix ? 1

Nom de la localité ? Eupen
Mercredi 30 ? 3
Jeudi 1 ? 2
Vendredi 2 ? 5
Samedi 3 ? 3
Dimanche 4 ? 2

Prévisions météo
1. Ajouter une localité et ses prévisions
2. Afficher les prévisions
3. Analyser les prévisions
4. Quitter
Choix ? 1

Nom de la localité ? Liège
Mercredi 30 ? 4
Jeudi 1 ? 5
Vendredi 2 ? 7
Samedi 3 ? 5
Dimanche 4 ? 4

Prévisions météo
1. Ajouter une localité et ses prévisions
2. Afficher les prévisions
3. Analyser les prévisions
4. Quitter
Choix ? 1

Nom de la localité ? Ostende
Mercredi 30 ? 5
Jeudi 1 ? 7
```

```
Vendredi 2 ? 9
Samedi 3 ? 7
Dimanche 4 ? 4

Prévisions météo
1. Ajouter une localité et ses prévisions
2. Afficher les prévisions
3. Analyser les prévisions
4. Quitter
Choix ? 2

                MER 30  JEU 1  VEN 2  SAM 3  DIM 4
Bruxelles          4,0    6,0    8,0    6,0    4,0
Eupen              3,0    2,0    5,0    3,0    2,0
Liège              4,0    5,0    7,0    5,0    4,0
Ostende            5,0    7,0    9,0    7,0    4,0

Prévisions météo
1. Ajouter une localité et ses prévisions
2. Afficher les prévisions
3. Analyser les prévisions
4. Quitter
Choix ? 3

La localité où il fera le plus chaud est Ostende avec 6,4°C
La localité où il fera le plus froid est Eupen avec 3,0°C
Le jour le plus chaud est le vendredi avec 7,3°C
Le jour le plus froid est le dimanche avec 3,5°C

Prévisions météo
1. Ajouter une localité et ses prévisions
2. Afficher les prévisions
3. Analyser les prévisions
4. Quitter
Choix ? 4

Fin du programme.
```