Exercice 1 : Dans un répertoire local « TestCdeJava » par ex, vous allez réaliser votre premier programme java en utilisant un éditeur simple et la ligne de commande

- 1.1 Écrire une classe "Hello.java", ajouter une méthode main pour afficher le message "bonjour et bienvenu dans mon programme java"
- 1.2 puis ajouter "quel est votre nom ?" + saisir le nom puis afficher "salut nom"
- 1.3 faire de même pour le prénom pour obtenir par ex : "bienvenu nom prénom !"

Exercice 2 : programme sur les conditions (éditeur simple + ligne de commande)

- 2.1 Écrivez une classe "Test.java", ajouter une méthode main qui lit un nombre entier et indique s'il est positif, négatif ou s'il vaut zéro (n'oubliez pas de gérer l'interaction)
- 2.2 Ajouter la possibilité de savoir si le nombre est pair ou impair.
- 2.3 Que se passe-t-il si on saisit une valeur inattendue et comment régler ce problème ?
- 2.4 Prévoir l'option permettant à l'utilisateur de saisir une valeur directement en ligne de commande, ce qui donnera le résultat suivant si vous exécutez en ligne de commande

$java Test 5

$valeur positive et impair

Exercice 3 : Créer un projet Java par exercice sous Eclipse, ex ici : « BaseEx3Loop »

- 3.1 Trouvez le moyen de faire la même chose que dans l'exercice 2.3 sous Eclipse
- 3.2 Que se passe-t-il si on saisit une valeur inattendue et comment régler ce problème ?
- 3.3 Ajouter l'option selon laquelle vous pouvez reproduire l'action précédente autant de fois qu'il y a d'arguments en ligne de commande puis tester le tout pour obtenir ce résultat :

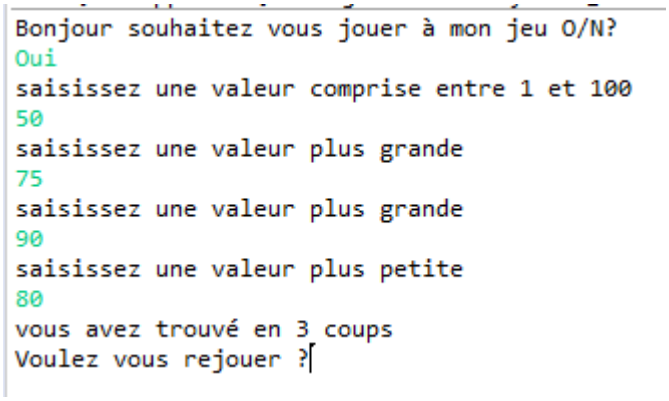
→ java Test 5 10 3 -4

```
5 positif et impair
10 positif et pair
3 positif et impair
-4 négatif et pair
```

TP 1 : [BaseTP1Game] Écrire un programme demandant à l'utilisateur s'il souhaite jouer à notre jeu, si non alors sortir du programme, si oui, alors proposer de saisir un chiffre entre 1 et 100 et dire à chaque itération si le chiffre est plus petit ou plus grand. Une fois le chiffre trouvé, afficher un message : vous avez trouvé en x coups !

- Puis rendre le jeu permanent tant que le joueur veut jouer, il doit pouvoir continuer

NB : la méthode Math.random() permet de renvoyer une valeur comprise entre 0 et 0,9



Exercice 4 : Les Tableaux [BaseEx4Array] & qq collections

- 4.1 Parcourir un tableau contenant des notes, écrire la note la plus petite, la plus grande et la moyenne (Utiliser des méthodes statiques pour gérer les différentes fonctionnalités)
- 4.2 : Dans une autre classe, demander au professeur de saisir nom et prénom d'un élève puis de saisir ses notes, une fois fini, afficher nom et prénom + moyenne, l'action est répétée autant de fois qu'il y a d'élèves.
- 4.3 : Le programme doit maintenant indiquer si un élève saisi au clavier est présent dans la liste du professeur, si oui, afficher ses notes et sa moyenne.
    - Utiliser la structure de donnée qui permet de gérer à la fois nom et prénom d'une part, suite de notes d'autre part.
- 4.4 : Écrire un programme qui va trier dans l'ordre décroissant la liste des prénoms de votre promo puis croissant.
- 4.5 : Que fait ce programme ? qu'affichera-t-il ?

```java
public class Unknown {
    public static void main(String[] args) {
        int table[] = { -5, 2, -8, 31, 15, 4 };

        displayTab(table);
        fonction(table);
        displayTab(table);
    }

    static void fonction(int[] tab) {
        int tmp = 0;
        for (int i = 0; i < tab.length; i++) {
            for (int j = 1; j < (tab.length - i); j++) {
                if (tab[j - 1] > tab[j]) {
                    tmp = tab[j - 1];
                    tab[j - 1] = tab[j];
                    tab[j] = tmp;
                }
            }
        }
    }

    static void displayTab(int[] tab) {
        for (int i = 0; i < tab.length; i++) {
            System.out.print(tab[i] + " ");
        }
        System.out.println();
    }
}
```

Exercice 5 : Les Strings [BaseEx5String]

- 5.1 : Trouver plusieurs moyens de concaténer 2 chaînes de type String.
- 5.2 : Dans une chaîne(phrase) donnée, trouver s'il existe un mot, si oui, afficher trouvé.
  ex : "il fait beau aujourd'hui" → mot recherché "Beau" donc trouvé !
- 5.3 : Reprendre exercice précédent et remplacer le mot si trouvé par un autre saisi par l'utilisateur. ex : "il fait beau aujourd'hui" donne "il fait chaud aujourd'hui"
- 5.4 : Écrire un programme qui dit si une chaîne est un palindrome, ex : akka, ottO
- 5.5 : Tester votre programme avec : "Elu par cette crapule" (c'est un palindrome), que faire ?

Exercice 6 : Les Opérations [BaseEx6Operation]

- 6.1 : Réaliser une classe Operation qui regroupe les méthodes Add, Sub, Mul et Div permettant de réaliser les opérations basiques telles que :
  → `System.out.println(add(5,2));` *va afficher 7*
  → `System.out.println(sub(5,2));` *va afficher 3*
  → *Comment gérez vous le cas suivant :*
  `System.out.println(div(3,0));`

- 6.2 : Trouver un moyen de dessiner un triangle sur la console, il en existe au moins 2.

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

- 6.3 : Écrire un code qui doit, pour un tableau contenant X nombres donné, retourner la somme des 2 plus grands nombres présent dans ce tableau. Exemple : avec le tableau 78, 6, -250, 2, 12, 9, le résultat sera 90.

- 6.4 : Dans une classe Matrix censé représenté une matrice, on souhaite additionner 2 matrices via une méthode, assurez-vous qu'elles ont bien le même nombre de colonnes et lignes avant tout pour respecter la formule.
  Mais avant tout voilà à quoi une ressemble l'opération d'addition de 2 matrices :

$$\begin{bmatrix} 1 & 2 & 0 \\ 4 & 3 & -1 \end{bmatrix} + \begin{bmatrix} 5 & 2 & 3 \\ 1 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 6 & 4 & 3 \\ 5 & 6 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 0 \\ 4 & 3 & -1 \end{bmatrix} - \begin{bmatrix} 5 & 2 & 3 \\ 1 & 3 & 4 \end{bmatrix} = \begin{bmatrix} -4 & 0 & -3 \\ 3 & 0 & -5 \end{bmatrix}$$

  Le premier exemple représente le résultat de l'addition de 2 matrices.
  En effet, une matrice est un tableau à 2 dimensions formé donc de lignes et de colonnes entouré de crochet ici. Par ex, sur la 1ère matrice, à la 2ème ligne et à la 3ème colonne, nous avons la valeur -1. Sur la 2ème matrice toujours à la 2ème ligne et à la 3ème colonne, nous avons la valeur 4. Delors, l'addition de ces 2 valeurs donne une valeur dans la 3ème matrice à droite (à la 2ème ligne et à la 3ème colonne) égal à 3, en effet -1 + 4 = 3.
  Le second exemple montre la soustraction d'une matrice par une autre avec au passage toujours les mêmes valeurs : -1 – 4 = -5

A l'aide de notre langage préféré, une matrice est comme dans l'exemple ci-dessous représenté par un tableau à 2 dimensions, cad que pour chaque ligne de notre tableau accessible via le 1er indice ici « *firstMatrix[0] » nous avons avons un tableau dont les éléments sont accessibles via le deuxième indice soit « firstMatrix[0][1] » contient l'entier 2.* ***Prenez le temps de bien comprendre les jeux d'essais et résultats plus bas pour comprendre les besoins.***

- 6.5 : Ajouter la méthode soustraire qui prend donc en argument 2 matrices (A et B si vous voulez) et renvoi une troisième que sera le résultat de A – B = C ;

- 6.6 : puis multiplier par une valeur (appelé scalaire) une matrice, ce qui revient à multiplier tous les éléments d'une matrice par une valeur donné (voir les résultats plus bas)

...de sorte que pour tel jeux d'essai

```java
public class Matrix {
    public static void main(String[] args) {
        int[][] firstMatrix  = { {5, 2, 4} ,
                                 {0, 1, 1} ,
                                 {6, 3, 1} };
        int[][] secondMatrix = { {1, 1, 3} ,
                                 {2, 5, 6} ,
                                 {3, 0, 5} };
        int[][] thirdMatrix =  { {8, 7} ,
                                 {5, 4} ,
                                 {2, 1} };
        int[][] fourthMatrix = { {0, 2} ,
                                 {1, 2} ,
                                 {1, 1} };

        addMatrix(firstMatrix, secondMatrix);
        System.out.println("--------------");
        addMatrix(firstMatrix, thirdMatrix);
        System.out.println("--------------");
        subMatrix(thirdMatrix, fourthMatrix);
        System.out.println("--------------");
        mulScalMatrix(fourthMatrix, 2);
    }
```

Nous obtenons les résultats suivants :

```
5   2   4
0   1   1
6   3   1
+
1   1   3
2   5   6
3   0   5
=
6   3   7
2   6   7
9   3   6
--------------
pour les additionner, les matrices doivent avoir le même nb de col/ligne
--------------
8   7
5   4
2   1
-
0   2
1   2
1   1
=
8   5
4   2
1   0
--------------
2 *
0   2
1   2
1   1
=
0   4
2   4
2   2
```

TP 2 [BaseTP2Resto] On se fait un « resto » à la maison ?

En effet, votre programme doit simuler la prise d'une commande afin d'obtenir ce résultat :

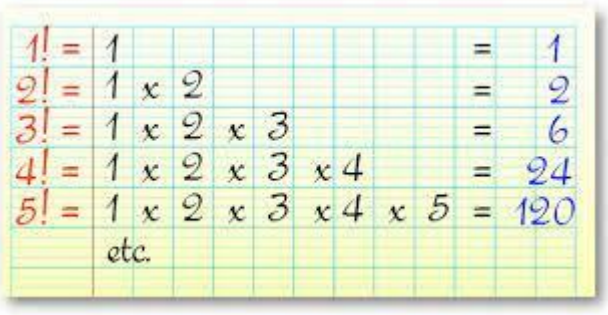
```
bonjour, combien de menus souhaitez vous ?
2
Commande numéro 1
choix entrée :
[1 - SALADE][2 - SOUPE][3 - QUICHE][4 - AUCUNE]
que souhaitez vous comme entrée ? [saisir le chiffre correspondant]
1
choix plats :
[1 - POULET][2 - BOEUF][3 - POISSON][4 - VÉGÉTARIEN][5 - VEGAN][6 - AUCUN]
que souhaitez vous comme plats ? [saisir le chiffre correspondant]
3
choix accompagnements :
[1 - RIZ][2 - PATES][3 - FRITES][4 - LÉGUMES][5 - AUCUN]
que souhaitez vous comme accompagnements ? [saisir le chiffre correspondant]
4
choix boissons :
[1 - EAU PLATE][2 - EAU GAZEUZE][3 - SODA][4 - VIN][5 - AUCUNE]
que souhaitez vous comme boissons ? [saisir le chiffre correspondant]
2
choix desserts :
[1 - TARTE MAISON][2 - MOUSSE AU CHOCOLAT][3 - TIRAMISU][4 - AUCUN]
que souhaitez vous comme desserts ? [saisir le chiffre correspondant]
4
Résumé de la commande 1
[salade, poisson, légumes, eau gazeuze]

Commande numéro 2
choix entrée :
[1 - SALADE][2 - SOUPE][3 - QUICHE][4 - AUCUNE]
que souhaitez vous comme entrée ? [saisir le chiffre correspondant]
4
choix plats :
[1 - POULET][2 - BOEUF][3 - POISSON][4 - VÉGÉTARIEN][5 - VEGAN][6 - AUCUN]
que souhaitez vous comme plats ? [saisir le chiffre correspondant]
5
choix accompagnements :
[1 - RIZ][2 - PATES][3 - FRITES][4 - LÉGUMES][5 - AUCUN]
que souhaitez vous comme accompagnements ? [saisir le chiffre correspondant]
5
choix boissons :
[1 - EAU PLATE][2 - EAU GAZEUZE][3 - SODA][4 - VIN][5 - AUCUNE]
que souhaitez vous comme boissons ? [saisir le chiffre correspondant]
1
```

NB : vous pouvez modifier les menus de-lors que les fonctionnalités répondent aux besoins.

Exercice 6 : Factorielle (récursivité)

```
Algorithme : Factorielle
  Entrées :  n un entier
  Sorties :  n! la factorielle de n
1 fonction Fact(n)
2 début
3     si n = 0 alors
4         retourner 1
5     sinon
6         retourner n * Fact(n-1)
7 fin
```



Ecrire un programme java qui met en œuvre la notion de factorielle telle que n ! = n*(n-1) …*1