

TP 1

Histogrammes



1 – Présentation du TP

Objet du TP

Ce TP consiste à afficher des valeurs fournies à un programme sous la forme d'un histogramme.

Par exemple, si les valeurs fournies au programme sont :

7	5	6	9	8	5	5	3	3	8	5	6
---	---	---	---	---	---	---	---	---	---	---	---

Alors le programme produira un affichage de la forme suivante dans la console :

```
3 **
4
5 ****
6 **
7 *
8 **
9 *
```

A la fin du TP, on vous propose 3 défis facultatifs et de difficulté croissante pour personnaliser l'affichage de cet histogramme.

Compétences travaillées dans ce TP

- Utilisation de l'IDE IntelliJ
- Conversion de types
- Manipulation de tableaux 1D
- Manipulation de la sortie standard
- Utilisation de la documentation de l'[API Java](#)

2 – Démarrage du TP

Cette procédure sera à suivre pour démarrer chaque TP.

Création du projet

1. Lancez l'environnement de programmation IntelliJ.
2. Créez un nouveau projet (File > New > Project).
3. Au besoin, sélectionnez « Java » dans la partie gauche et cliquez sur « Next ».
4. **Attention : Assurez-vous que la case « Create project from template » est DECOCHÉE. Cliquez ensuite sur « Next ».**
5. Dans le champ « Project name », entrez « tp1 ».

6. De base, les projets sont enregistrés dans un répertoire « IdeaProjects » dans le répertoire de l'utilisateur courant. Si vous le souhaitez, vous pouvez changer ce répertoire en modifiant le champ « Project location » ou en cliquant sur « ... ».
7. Cliquez sur « Finish ».

Création de la classe principale

Faites un clic droit sur le répertoire « src » de la partie gauche, et sélectionnez New > Java Class.

Entrez le nom de la classe à créer : « Main », ce qui signifie « classe principale ».

Ajoutez une méthode main(), point d'entrée du programme :

```
public static void main(String[] args)
```

Première exécution de la classe principale

Faites un clic droit sur le code de la classe Main, et sélectionnez « Run Main.main() ».

IntelliJ va alors faire ce que vous avez fait à la main dans le TP précédent, à savoir :

- Compiler le programme avec `javac` si nécessaire (s'il y a eu des modifications dans le code)
- Exécuter le programme avec `java`.

Comme le main() est vide, le programme ne va rien faire et se terminer normalement, ce qui est matérialisé par le message :

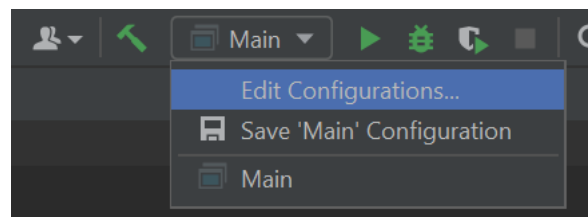
```
Process finished with exit code 0
```

Le code 0 correspond à une fin de programme sans erreur.

3 – Affichage des arguments en ligne de commande

Comme ce n'est plus vous, mais IntelliJ, qui lance le programme, il faut indiquer à l'IDE les arguments en ligne de commande éventuels à communiquer à l'exécution de la commande java.

Pour ce faire, cliquez sur « Main » en haut à droite puis sur « Edit Configurations » :



Entrez des valeurs séparées par des espaces dans le champ de texte appelé « Program arguments », par exemple :



Au moment du lancement du programme, l'IDE lancera une commande de la forme :

```
java Histogram 1 2 2 4
```

Complétez le code de la méthode `main()` pour que le programme affiche le contenu du tableau `args` sur une ligne, comme ceci :

```
1 2 2 4
```

4 – Conversion des entrées

Implémentez une fonction :

```
public static int[] getIntegerValues(String[] args)
```




qui convertit le tableau de chaînes de caractères `args` en tableau d'entiers.

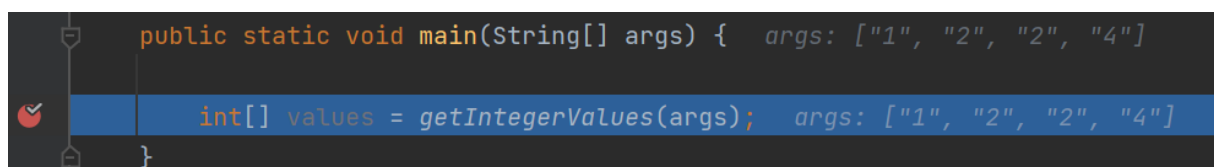
Note : Le mot clé `static` nous permet de faire des fonctions utilitaires, nous détaillerons ceci dans un prochain cours.


Pour tester le bon fonctionnement de votre fonction, vous pouvez utiliser le débogueur :

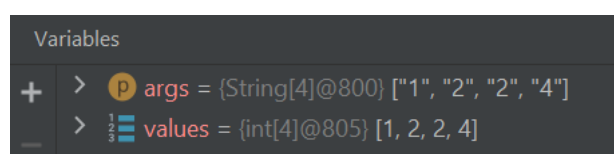
- Faites appel à votre fonction dans le `main()`, par exemple avec :

```
int[] values = getIntegerValues(args);
```

- Placez un point d'arrêt au niveau de l'appel à la fonction, à côté du numéro de ligne. Un rond rouge  apparaîtra.
- Lancez votre programme avec  plutôt qu'avec  : il s'arrêtera au niveau du point d'arrêt :



- Appuyez sur  pour passer à la ligne suivante. Dans la partie « Variables » en bas de la fenêtre, vous verrez la valeur des variables, et notamment celle de `values`, qui doit valoir `[1,2,2,4]`.



Variables	
>	<code>args</code> = {String[4]@800} ["1", "2", "2", "4"]
>	<code>values</code> = {int[4]@805} [1, 2, 2, 4]

5 – Comptage des occurrences

Pour le moment, nous allons considérer que les valeurs données en argument du programme sont des chiffres de 0 à 9.

Implémentez une fonction :

```
public static int[] getOccurrences(int[] values)
```

qui renvoie, sous la forme d'un tableau d'entiers, le nombre d'occurrences de chaque chiffre présent dans le tableau `values`.

Exemple : si `values` vaut `[1, 2, 2, 4]`, alors `getOccurrences(values)` doit renvoyer `[0, 1, 2, 0, 1]`, qui signifie :

- 0 occurrence du chiffre 0,
- 1 occurrence du chiffre 1,
- 2 occurrences du chiffre 2,
- 0 occurrence du chiffre 3,
- 1 occurrence du chiffre 4.

Testez le bon fonctionnement de votre fonction en utilisant le débogueur.

6 – Affichage de l'histogramme

Implémentez une fonction :

```
public static void printHorizontalHistogram(int[] values)
```

qui affiche la répartition des valeurs contenues dans le tableau `values` sous la forme d'un histogramme horizontal.

Une ligne de l'histogramme affichera une valeur suivie d'étoiles (une étoile par occurrence de cette valeur)

Exemple : si `values` vaut `[1, 2, 2, 4]`, alors `printHorizontalHistogram(values)` doit afficher :

```
0
1 *
2 **
3
4 *
5
6
7
8
9
```

Faites d'autres tests en modifiant les arguments en ligne de commande donnés au programme (en allant dans « Edit Configurations » comme au début du TP).

7 – Adaptation aux données

Faites en sorte que l'histogramme commence à la valeur minimale détectée dans le tableau de valeurs, et s'arrête à la valeur maximale.

Exemple : si values vaut [1, 2, 2, 4], alors printHorizontalHistogram(values) doit afficher :

```
1 *
2 **
3
4 *
```

Aide :

- Regardez la documentation de la classe Math.
- Quelles fonctions devez-vous ajouter à votre programme ?
- Quelles fonctions existantes devez-vous adapter ?

8 – [Facultatif] Paramétrage de l'histogramme

Paramétrage du symbole

Faites en sorte que l'on puisse à présent modifier le symbole affiché pour chaque occurrence en fournissant les paramètres suivants au programme :

```
-s symbole -v valeur1 valeur2 ...
```

- L'argument `-s` est facultatif : s'il est absent, alors le symbole `*` sera utilisé par défaut.
- L'argument `-v` est obligatoire : s'il n'est pas trouvé, le programme doit afficher une erreur et se terminer.

Exemple :

Si les arguments donnés au programme sont :

```
-s o -v 7 5 6 9 8 5 5 3 3 8 5 6
```

Alors le programme doit afficher :

```
3 oo
4
5 oooo
6 oo
7 o
8 oo
9 o
```

Conseils :

- Implémentez une fonction qui renvoie l'indice d'une option dans le tableau d'arguments
- Pour terminer le programme, regardez la documentation de la classe System.

Paramètres dans n'importe quel ordre

Faites en sorte que l'on puisse mettre les arguments -o et -v dans n'importe quel ordre.

Exemple : Si on communique les paramètres suivants au programme, on doit avoir le même résultat que précédemment :

```
-v 7 5 6 9 8 5 5 3 3 8 5 6 -s o
```

Paramétrage de l'orientation

Faites en sorte que l'on puisse à présent afficher l'histogramme horizontalement ou verticalement, par l'intermédiaire d'un argument supplémentaire -o :

- Si on communique l'argument -o h, alors l'histogramme sera affiché horizontalement,
- Si on communique l'argument -o v, alors l'histogramme sera affiché verticalement,
- Si l'argument -o est absent, alors l'histogramme sera affiché horizontalement par défaut.

Exemple :

Si les arguments donnés au programme sont :

```
-v 7 5 6 9 8 5 5 3 3 8 5 6 -s + -o v
```

Alors le programme doit afficher :

```

+
+
+ + + +
+ + + + +
3 4 5 6 7 8 9
```

Comme précédemment, les arguments peuvent être donnés dans n'importe quel ordre.