TP1 – Syntaxe de base

2ème Année - TP Conception et Programmation Objet Année 2021/2022

Objectif du TP

Le but de ce premier TP est de vous familiariser avec la syntaxe Java, de manipuler les opérations de base : affectation, boucles, manipulation de tableaux, subtilités de java … il s'agit d'un TP préparatoire avant la pleine utilisation du langage objet.

Déroulé de la séance, modalités de rendu

1. Chaque exercice doit faire l'objet d'un nouveau projet dédie, dont le nom doit respecter exactement la nomenclature demandée.
2. Pour chaque exercice, vous modifierez l'en-tete du fichier principal et **apporterez un soin particulier à commenter votre code**.
3. Il est recommandé de faire valider chaque étape par votre encadrant
4. A la fin de la séance, n'oubliez pas de sauvegarder votre travail (en utilisant GitHub)

1 Saisie et manipulation de nombres

Nom du projet : **TP1_manipNombresInt_*NOM1_NOM2*** en remplaçant ***NOM1*** et ***NOM2*** par les noms des membres du binôme (en majuscule)

1. Ecrivez un programme, à l'intérieur de la fonction `Main()`, qui vous demande de saisir deux entiers (type `int`) que vous stockerez dans deux variables. Vous utilisez pour cela un Scanner ainsi que la méthode `nextInt()` comme vous avez pu voir sur le premier TP0, et affichez les à l'écran.
2. Affichez ensuite à l'écran la somme de ces deux nombres, leur différence, et leur produit, avec un message de votre choix
3. Ajoutez ensuite une ligne pour afficher le quotient entier et le reste de la division euclidienne du premier par le second. Pour cela on utilisera deux opérateurs **`/`** (division) et **`%`** (reste de la division euclidienne). Voir le point cours ci-après.

Point cours :

L'opérateur **`/`** est celui de la division. Mais il possède une particularité intéressante (ou déroutante, selon le cas) : Si les deux opérandes sont des `int`, alors la valeur retournée est arrondie à l'entier inférieur. Ainsi :

`12 / 5` retournera 2 et non 2.4.

Si on veut la valeur exacte, en `double`, il faut forcer au moins une des opérandes à être un double, par exemple en ajoutant une partie décimale, même nulle, ou en multipliant par 1.0. Ainsi :

`12.0 / 5` retournera bien `2.4`

il en serait de même pour `12 / 5.0` ou `(12*1.0) / 5`. Mais attention, sans priorisation (/ et * sont au même niveau de priorisation) les opérations se lisent de la gauche vers la droite, ainsi :

12 /5 * 1.0 , sans parenthèses, retournera 2.0 et non 2.4 ! Car l'exécution prioritaire de `12 / 5` retournerait d'abord 2 en `int`, qui multiplié par `1.0` donnerait `2.0` en `double`.

2 Un convertisseur de températures

Nom du projet : **TP1_convertisseur_*NOM1_NOM2*** en remplaçant ***NOM1*** et ***NOM2*** par les noms des membres du binôme (en majuscule)

L'objectif de cet exercice est de réaliser un convertisseur de températures utilisant des méthodes.

1. Ecrivez un programme, à l'intérieur de la fonction `Main()`, qui vous demande de saisir une valeur réelle (type `double`) que vous stockerez dans une variable. Vous utilisez pour cela un Scanner ainsi que la méthode `nextDouble()`, qui est une méthode similaire à celle de `nextInt()` vue sur le TP0 mais adaptée au type `double`, et affichez le à l'écran.
2. Supposant que cette valeur est une température en degré Celcius, calculez et afficher cette valeur de température en degré Kelvin et affichez-la à l'écran (cherchez les formules de conversion sur Internet)
3. Modifiez votre programme en ajoutant une méthode : hors de la méthode `Main()`, ajoutez une nouvelle méthode nommée `CelciusVersKelvin()` qui prend en paramètre une valeur de type `double` correspondant à une température en degré Celcius et retourne cette même valeur en degré Kelvin, toujours de type `double`. Il vous faut donc ajouter la méthode suivante (nous reviendrons sur le mot clé 'public' une autre fois) :

```
public static double CelciusVersKelvin (double tCelcius) {
  … // à completer
}
```

   Il est important pour des raisons d'organisation, d'organisation de code, que cette méthode se limite à des opérations de calculs et n'affiche rien à l'écran, ou ne demande pas à l'utilisateur de saisir quoi que ce soit. Une méthode doit avoir une fonction unique, et une seule. Terminez de modifier votre programme de sorte que ce dernier appelle cette méthode avec en paramètre une valeur saisie, et affiche la valeur retournée à l'écran.

4. Dans le même ordre d'idée et sur le même principe, rajoutez les méthodes suivantes qui convertissent une température d'une unité citée en une autre :
   - **`KelvinVersCelcius(..)`**
   - **`FarenheitVersCelcius( )`**
   - **`CelciusVersFarenheit()`**
   - **`KelvinVersFarenheit()`**
   - **`FarenheitVersKelvin()`**

   Pour ces deux dernières méthodes, si vous êtes malins vous n'avez même pas besoin de chercher la formule de conversion, sachant que, par exemple, une conversion Kelvin vers Farenheit peut faire avec une conversion Kelvin vers Celcius puis Celcius vers Farenheit. Il vous suffit, au sein d'une méthode, d'appeler les deux autres méthodes en prenant le résultat de l'une comme paramètre de l'autre. Si vous n'êtes pas malins, procédez comme précédemment.

   Testez vos méthodes en les appelant et en affichant le résultat à l'écran t.
5. Complétez enfin votre programme pour présenter un vrai menu complet. Selon le choix de l'utilisateur, la conversion adéquate sera appelée et le résultat affiché comme dans l'exemple suivant

```
Bonjour, saisissez une valeur :
50
Saisissez la conversion que vous souhaiter effectuer :
1) De Celcius vers Kelvin
2) De Kelvin vers Celcius
…
6) De Farenheit vers Kelvin
2
50 degré Kelvin est égal à -223.15 degrés Celcius
```

3 GuessMyNumber

Nom du projet : **TP1_guessMyNumber_*NOM1_NOM2*** en remplaçant ***NOM1*** et ***NOM2*** par les noms des membres du binôme (en majuscule)

L'objectif de cet exercice est de réaliser un jeu de type « Guess My Number » : un nombre aléatoire est choisi, l'utilisateur doit le deviner. Manipuler des nombres aléatoires est une étape clé pour instaurer un certain dynamisme dans vos programmes et vos jeux

1. Commençons par générer quelques nombres aléatoires : écrivez un programme, à l'intérieur de la fonction `Main()`, rajoutez cette instruction :

```
Random generateurAleat = new Random();
```

   Cette instruction crée un générateur de nombres aléatoires nommé `generateurAleat` qui va nous fournir des nombres aléatoires. Cette instruction n'est à écrire qu'une seule fois. Si on veut obtenir un nombre aléatoire `n` inférieur ou égal à 100, on écrira :

```
int n = generateurAleat.nextInt(100);
```

   Testez votre générateur en affichant 5 nombres aléatoires à l'écran.

2. Ecrivez le cœur de votre programme :
   - Générez un nombre aléatoire entre 0 et 100
   - Demandez à l'utilisateur de saisir un nombre entre 0 et 100
   - Comparez le nombre saisi avec le nombre aléatoire, et affichez le message « trop petit » « trop grand » ou « gagné » selon que l'utilisateur rentre un nombre trop petit, trop grand, ou égal au nombre généré aléatoirement.
   - Ajoutez une boucle `while () { …}` qui demande à l'utilisateur de saisir une valeur et affiche un message similaire, tant que la valeur entrée n'est pas égale à la valeur cherchée
3. Ajoutez un compteur qui indique le nombre de tentatives exécutées et l'affiche lorsque le nombre est trouvé
4. Modifiez enfin votre programme pour instaurer un mode de difficulté : au début de votre programme, demandez à l'utilisateur quel niveau de difficulté il souhaite (facile, normal, difficile ?). Selon son choix, modifiez les règles de votre jeu à votre convenance : nombre de coups limités, modification de l'intervalle…ou messages d'aide plus faciles « vraiment trop petit », « vraiment trop grand » si l'écart est très grand.
5. ***Bonus*** : instaurez un mode cauchemar : lorsque le nombre proposé n'est pas le nombre recherché votre programme donne la mauvaise indication avec une probabilité de 30% (trop grand alors que le nombre entré est trop petit, et réciproquement).

4 Statistiques

Nom du projet : **TP1_stats_*NOM1_NOM2*** en remplaçant ***NOM1*** et ***NOM2*** par les noms des membres du binôme (en majuscule)

L'objectif de cet exercice est vérifier si la distribution de nombres du générateur de nombre aléatoire est bien aléatoire, en simulant un lancer de dés à répétition. Il vous est demandé de :

1. Créer un tableau de 6 entiers. N'oubliez pas d'initialiser sa mémoire avec l'instruction new. L'ensemble des cases de ce tableau est initialisé à 0. Chaque cellule correspond à une face du dé.
2. Demander à l'utilisateur de saisir un nombre entier m
3. Faites une boucle de m itérations, et à chaque itérations, tirez un nombre aléatoire entre 0et 5 (utilisez les instructions vues précédemment) et incrémentez la case d'indice correspondant.
4. Affichez enfin le tableau résultat à l'écran (faites une boucle pour afficher la valeur de chaque cellule)

5. Modifiez un peu l'affichage pour afficher les résultats obtenus sous forme de pourcentages (en `double`)