Programmation Java

TP°1 – Premiers programmes en Java

I. Système impérial

A. Description du résultat à atteindre

Avec *NetBeans*, dans un dossier *TP1*, vous devez créer le projet *SystemeImperial* contenant un programme dont l'exécution génère ce type de dialogue (les textes soulignés sont des exemples saisis par l'utilisateur du programme) :

```
Saisissez un poids en kilogrammes : 76,5
Dans le système impérial, cela représente 168lbs 10oz
Au revoir !
```

B. Méthode

Les coefficients pour la conversion sont les suivants(lbs = livre, oz = once) :

- 1kg = 2,20462lbs
- 1kg = 35,2739199982575oz

Vous procéderez pas à pas : il ne s'agit pas d'écrire tout le programme en une seule fois, mais plutôt de valider l'une après l'autre chaque fonctionnalité du programme.

C. Qualité de programmation

Chaque entreprise adopte généralement son style de présentation des programmes.
Cependant, on peut assez facilement évaluer le niveau de qualité rédactionnelle d'un programme.

Posez un œil critique sur votre programme et répondez aux questions suivantes :

- Les noms des variables et sous-programmes sont-ils explicites ?
  Evitez les *a, b, x, toto*, *tata*, *test* …
- Une convention de nommage des variables et sous-programmes est-elle appliquée ?
  En général, on utilise *camelCase* pour les noms des variables et sous-programmes, *UpperCamelCase* pour les noms des classes et *UPPER_SNAKE* pour les noms des constantes.

  On utilise des verbes à l'infinitif (ou à l'impératif) pour les procédures et des noms communs pour les fonctions.

- Une convention est-elle respectée pour l'utilisation des *{* et *}* ?
  Il y a plusieurs manières de placer les *{* et *}*. Choisissez-en une et respectez ce choix tout au long du programme.

```
if (x < 0) {
  negative(x);
} else {
  nonnegative(x);
}
```

ou

```
if (x < 0)
{
  negative(x);
}
else
{
  nonnegative(x);
}
```

- La règle *DRY* (Don't Repeat Yourself) est-elle respectée ?
- Des données sont-elles stockées "en dur" dans votre programme ?

Corrigez votre programme afin de résoudre **tous** les problèmes liés à la qualité de programmation.

D. Tests

Utilisez votre programme et vérifiez qu'il effectue bien le travail attendu.

S'il y a des cas où votre programme n'est pas pertinent, améliorez-le (exemple : que fait votre programme si l'utilisateur saisit un poids négatif ?)

Dans quels cas le programme génère-t-il une erreur d'exécution ?

Utilisez le programme de votre voisin et essayez de provoquer des erreurs d'exécution.

Vous ne pouvez pas encore prendre en compte toutes ces erreurs. Corrigez celles que vous pouvez.

E. Création d'un sous-programme

Découpez votre programme en un programme principal et deux sous-programmes :

- *public static double conversionKilogrammeVersLivre(double kilogramme)*
- *public static double conversionKilogrammeVersOnce(double kilogramme)*

Les traitements seront effectués par les sous-programmes, le programme principal se contentera de faire appel aux sous-programmes en déclenchant leurs exécutions.

Ces sous-programmes sont-ils des procédures ou des fonctions ?

II. Jeu : plus petit / plus grand

A. *Récupération d'un projet existant*

Sur Moodle, récupérez le projet NetBeans *Jeu.* Ouvrez ce projet et analysez attentivement le code qui le compose.

B. *Description du résultat à atteindre*

Modifiez le code de ce programme de manière à mettre en place le jeu du « plus petit / plus grand » dont un déroulement est décrit ci-dessous :

```
Saisissez un entier entre 0 et 100:
76
Plus petit !
Saisissez un entier entre 0 et 100:
35
Plus grand !
Saisissez un entier entre 0 et 100:
55
Plus grand !
Saisissez un entier entre 0 et 100:
63
Plus petit !
Saisissez un entier entre 0 et 100:
58
Gagné !
```

C'est le programme qui tire un nombre secret au hasard et l'utilisateur doit le découvrir.

III. Nombres entiers

Créez un nouveau projet NetBeans *NombresEntiers.*

Implémentez les sous-programmes suivant :

- public static void afficherDiviseur(int n) : qui affiche les diviseurs de n.
- public static String listeDiviseur(int n) : comme le précédent, sauf que la chaine est retournée plutôt qu'affichée.
- public static int nbDiviseurs(int n) : qui retourne le nombre de diviseurs de n.
- public static boolean estPremier(int n) : qui retourne vrai si et seulement si n est premier.
- public static String decompositionFacteursPpremiers(int n) : qui retourne, dans une chaîne de caractères la décomposition en facteurs premiers. Par exemple pour 2100, la chaîne « 2^2x3^1x5^2x7^1 » sera retournée, ou encore pour 14703, ce sera « 3^1x13^2x29^1 ».
- public static int nbDiviseursV2(int n) : qui retourne nombre de diviseurs de n, mais en utilisant la décomposition en facteurs premiers. Par exemple pour 2100, la décomposition est « 2^2x3^1x5^2x7^1 ». Pour trouver le nombre de diviseurs, il faut ajouter 1 à chaque exposant et les multiplier entre eux. Pour 2100, on obtient 3x2x3x2 = 36 diviseurs.
-

Vous testerez vos fonctions au fur et à mesure que vous les écrivez en utilisant des exemples simples pour lesquels vous connaissez la réponse. Essayez de faire produire des réponses fausses à votre programme (et corrigez le cas échéant) Conservez tout ces tests dans votre « main ».