Licence MASS – INF F3 – TP n° 4

2013–2014

Le but de ce TP est de réaliser une classe puis de s'en servir comme moteur derrière une interface graphique. La classe demandée devra implémenter les règles du jeu « Le compte est bon » présent dans l'émission « Des chiffres et des lettres » (`http://fr.wikipedia.org/wiki/Des_chiffres_et_des_lettres#Le_Compte_est_bon`).

1 Le compte est bon

1.1 Description

Le compte est bon est un jeu où, à partir de 6 nombres et des 4 opérations de base, un joueur doit atteindre (ou se rapprocher le plus possible) d'un total fixé entre 100 et 999.

À chaque étape du calcul, deux nombres sont utilisé pour produire un résultat, diminuant ainsi la quantité de nombres disponibles. Les résultats intermédiaire pouvant être utilisés dans la suite des calculs, il n'y a donc que 5 étapes possible avant d'atteindre le résultat final (ou le nombre le plus proche possible).

Les seules restriction sur l'utilisation des opérations est qu'un résultat intermédiaire ne peut pas être ni négatif, ni décimal. Du reste, chaque nombre (résultat intermédiaire compris) ne peut être utilisé qu'une seule fois.

1.2 Exercice

1.2.1 Moteur de jeu

Écrire une classe (normale, pas visuelle) `CompteEstBon` qui va permettre d'implémenter les règles de ce jeu.

Le constructeur de la classe va initialiser le total à atteindre de façon aléatoire (utilisation de `Math.random()` ou d'un objet `Random`) entre 100 et 999. Il va aussi initialiser les 6 nombres de départ. Dans les règles initiales du jeu, les nombres étaient choisis à l'aide de plaques numérotées. Il y avait des plaques de 1 à 10 présentes en double exemplaire et des plaques 25, 50, 75 et 100 présentes en simple exemplaire.

Ajoutez une méthode `public String toString()` qui retourne une chaîne de caractères indiquant le total à atteindre et la liste des nombres actuellement utilisables (cette liste est susceptible d'être modifiée en cours de jeu).

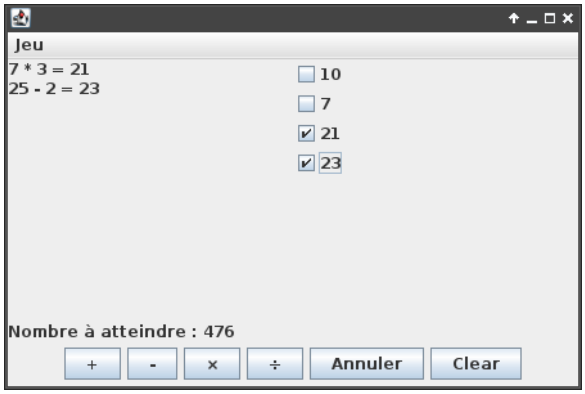
Ajoutez une fonction `main` pour tester votre programme. Créez plusieurs instances de la classe `CompteEstBon` et affichez-les sur la console. N'hésitez pas à utiliser cette fonction pour tester les prochaines méthodes que vous implémenterez.

Ajoutez 4 méthodes, une pour chaque opération mathématique, dont la signature est `public int operation(int, int)`. Chaque méthode commence par vérifier si les deux chiffres passés en paramètre sont disponibles et utilisables. Si ce n'est pas le cas, la méthode retourne 0. Si c'est le cas, les deux nombres sont utilisés (et ne sont, donc, plus utilisables) et le résultat de l'opération est ajouté aux nombres utilisables. Faites attention aux restrictions sur le résultat des calculs et retournez 0 si l'opération n'est pas possible. Si l'opération est possible, retournez

le résultat. N'hésitez pas à utiliser des méthodes privées pour éviter de recopier les mêmes lignes de code dans chacune de ces méthodes.

Ajoutez une méthode `public int[] getPossibilites()` qui retourne la liste des nombres actuellement utilisables et modifiez la méthode `toString` pour qu'elle utilise cette méthode.

1.2.2 Interface graphique



Écrire une classe visuelle `LeCompteEstBon` qui possède un attribut de type `CompteEstBon` représentant l'état interne du jeu.

Séparez l'interface graphique en trois parties, à gauche, à droite et en bas. La partie de gauche servira à afficher un récapitulatif des opérations effectuées, la partie droite présente les nombres qu'il est possible d'utiliser (sous forme de cases à cocher) et la partie basse propose des boutons pour chaque action (les 4 opérations mathématiques). Un exemple d'interface graphique est donné ci-dessus.

Comme il sera nécessaire d'effacer le contenu de la zone de droite et d'y rajouter les nouvelles possibilités de nombre, écrivez une méthode privée qui se charge de demander au moteur de jeu la liste des nombres possibles et qui remplit la zone de droite de ces possibilités.

Terminez par les `actionPerformed` des boutons pour les 4 opérations. Ces boutons vont rechercher, parmis les cases à cocher, les deux premières cochées (méthode `isSelected`) qui vont fournir les nombres à utiliser pour l'opération. Pour la soustraction et la division, vérifiez dans quel ordre vous passez les arguments et utilisez toujours le nombre le plus grand comme opérande de gauche de l'opération (vous pouvez effectuer cette vérification dans le `actionPerformed` ou dans la méthode de l'opération que vous appelez). S'il n'y a qu'une case cochée ou si l'opération n'est pas permise, il ne se passe rien. Sinon, mettez à jour l'affichage de la partie droite et ajoutez un label dans la partie gauche comme récapitulatif de l'opération.

Pour finir, si après l'appui sur une opération le résultat est le nombre recherché, une boite de dialogue apparait pour notifier cet évènement.

1.3 Nouvelle partie et correction d'erreurs

Ajoutez un menu « Nouvelle partie » qui va instancier à nouveau l'attribut de type `CompteEstBon` et réinintialiser l'interface graphique.

Ajoutez une méthode `public void annuler()` à la classe `CompteEstBon` ainsi qu'un bouton « Annuler » dans l'interface graphique. Cette méthode permet d'annuler la dernière opération effectuée. Elle est sans effet si aucune opération n'a encore été effectué ou si toutes les opérations précédentes ont été annulées. Pour réaliser cette action, une technique possible est de garder une trace des nombres utilisés pour les calculs précédents.

Ajoutez aussi une méthode `annulerTout` qui permet d'effacer tout l'historique d'un coup et associez l'appel de cette méthode à un clic sur un bouton « Clear » que vous ajouterez.