Université de Nice–Sophia Antipolis — Vendredi 12 décembre 2003
E.S.I.N.S.A. 1ère Année

Contrôle de TP n°1b

Durée : 2h
Documents non autorisés

Attention : la qualité des commentaires (avec notamment, la présence des antécédents et conséquents avant chaque méthode), les noms donnés aux variables, l'utilisation à bon escient des majuscules et la bonne indentation rentreront pour une part **importante** dans l'appréciation du travail.
Les exercices ne sont pas tous indépendants. Cependant, si vous n'avez pas réussi à écrire la méthode demandée à l'un des exercices, vous pouvez faire les autres exercices en considérant que vous disposez quand même de ladite méthode. Le sujet comporte un verso.

Avant de commencer

Vous mettrez votre solution dans votre compte sur `djinn`.

Monnaie et monnayeur

Vous allez représenter de la monnaie à l'aide d'une classe Java. La *monnaie* est définie comme un ensemble de billets et de pièces de valeurs différentes. Vous vous limiterez aux billets de 10 et 5 euros, et aux pièces de 2 et 1 euros. Les exercices qui suivent vont vous guider dans la réalisation de cette classe et des opérations qu'elle permet. En parallèle, vous testerez chaque méthode à partir d'une classe `Test` .

1) Définissez la classe `Test` qui ne contient que la fonction :

```
public static void main(String[] args) throws IOException .
```

N'oubliez pas la directive : `import java.io.*;` .

2) Définissez la classe `Monnaie` qui comporte quatre attributs privés de type entier, un pour chaque catégorie de billets et de pièces. Chaque attribut contiendra le nombre de billets (ou de pièces) disponibles dans la catégorie qu'il représente.

3) Ajoutez à la classe `Monnaie` deux constructeurs et testez-les dans la fonction `main` de la classe `Test` .

3.a) Le premier constructeur ne prend aucun paramètre et crée une monnaie vide en initialisant les attributs à zéro.

3.b) Le second constructeur possède quatre paramètres de type entier, un par catégorie de monnaie, qui sont les nombres de billets et de pièces avec lesquels initialiser les attributs.

4) Ajoutez (dans `Monnaie`) et testez (dans `Test`) la fonction `toString` qui retourne une représentation du contenu de la monnaie sous la forme[1] :

[1] Les '?' sont évidemment à remplacer par les valeurs correspondantes.


```
nombre de billets de 10 = ?
nombre de billets de 5 = ?
nombre de pièces de 2 = ?
nombre de pièces de 1 = ?
```

5) Définissez et testez la fonction `total` qui ne possède aucun paramètre et qui retourne un entier correspondant à la somme en euros contenue dans la monnaie courante.

6) Définissez et testez la procédure `public void retirer(Monnaie àRetirer)` qui possède un paramètre de type `Monnaie` et qui diminue la monnaie courante de la monnaie passée en paramètre. Cette méthode présuppose que la monnaie courante contienne suffisamment de billets et de pièces pour que l'opération soit possible. *Exemple :*

monnaie à retirer	monnaie courante	
	avant retrait	après retrait
nombre de billets de 10 = 0 nombre de billets de 5 = 1 nombre de pièces de 2 = 2 nombre de pièces de 1 = 3	nombre de billets de 10 = 5 nombre de billets de 5 = 5 nombre de pièces de 2 = 5 nombre de pièces de 1 = 5	nombre de billets de 10 = 5 nombre de billets de 5 = 4 nombre de pièces de 2 = 3 nombre de pièces de 1 = 2

7) Vous allez maintenant définir une méthode `ajouter` qui permet d'ajouter un nombre donné de billets (ou de pièces) à une catégorie donnée de billets (ou de pièces).

7.a) Pour cela définissez quatre constantes de classe (`final`) de type entier qui vous permettront de distinguer les quatre catégories de billets (ou de pièces). En choisissant judicieusement le nom des constantes, il vous sera alors possible de désigner clairement une catégorie de billets (ou de pièces) grâce à la constante correspondante. Vous initialiserez ces constantes aux valeurs qui vous conviennent pourvu qu'elles soient différentes.

7.b) À l'aide des constantes ainsi définies, écrivez et testez une procédure

```
public void ajouter(int catégorieBillet, int nbBillets)
```

qui, dans la monnaie courante, ajoute `nbBillets` billets (ou pièces) au nombre courant de billets (ou de pièces) de la catégorie `catégorieBillet` . *Exemple :* Cette méthode permet d'ajouter 3 billets de 5 euros dans la monnaie courante.

8) Définissez et testez la méthode `public Monnaie décomposer(int somme)` qui retourne un objet monnaie contenant la décomposition en petite monnaie de la somme passée en paramètre, *et cela en fonction des billets et pièces disponibles dans la monnaie courante.* Si le contenu de la monnaie courante ne permet pas l'opération, vous afficherez un message indiquant clairement quelle est l'erreur et vous terminerez l'application grâce à un appel à `System.exit(1)` [2]. *Exemples :*

[2] Vous verrez plus tard dans le cours comment gérer plus élégamment les cas d'erreur.

monnaie courante	décomposition de somme = 120
nombre de billets de 10 = 10 nombre de billets de 5 = 5 nombre de pièces de 2 = 5 nombre de pièces de 1 = 4	nombre de billets de 10 = 10 nombre de billets de 5 = 4 nombre de pièces de 2 = 0 nombre de pièces de 1 = 0

monnaie courante	décomposition de somme = 7
nombre de billets de 10 = 10 nombre de billets de 5 = 0 nombre de pièces de 2 = 2 nombre de pièces de 1 = 1	erreur : pas de monnaie correspondante. fin du programme.

9) En vous aidant des méthodes précédemment définies, ajoutez à la classe `Monnaie` une procédure `faireMonnaie` dont le rôle est de faire la monnaie sur un montant donné. Cette procédure possède un paramètre de type entier représentant le montant sur lequel il faut faire la monnaie. Cette procédure affiche la monnaie à rendre et modifie la monnaie courante pour tenir compte de la monnaie rendue. En revanche, le montant versé n'est pas ajouté à la monnaie courante. N'oubliez pas de tester cette procédure. *Exemple :*
Soit à faire la monnaie sur un montant de 50 euros :

monnaie courante avant l'opération	monnaie rendue	monnaie courante après l'opération
nombre de billets de 10 = 3 nombre de billets de 5 = 3 nombre de pièces de 2 = 4 nombre de pièces de 1 = 4	nombre de billets de 10 = 3 nombre de billets de 5 = 3 nombre de pièces de 2 = 2 nombre de pièces de 1 = 1	nombre de billets de 10 = 0 nombre de billets de 5 = 0 nombre de pièces de 2 = 2 nombre de pièces de 1 = 3

10) Pour terminer, ajoutez et complétez dans la fonction `main` de votre classe `Test` :

```
int montant;

//définition d'une variable, de nom ''monnayeur'', de type
//Monnaie, initialisée à des valeurs différentes de zéro.
...

//ajout de 3 billets de 10 euros dans monnayeur
...

//affichage du détail de monnayeur
...

//lecture sur l'entrée standard du montant sur lequel on va
//faire la monnaie
System.out.print("montant = ");
...
```



```
  //avec monnayeur faire la monnaie sur le montant lu
  ...

  //affichage du nouveau contenu de monnayeur
  ...
}
```