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

Contrôle de TP n°1a

Documents non autorisés — Durée : 2h

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] :

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

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

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 deux procédures, `ajouter` et `retirer` , qui possèdent chacune un paramètre de type `Monnaie` .

6.a) `public void ajouter(Monnaie àAjouter)` qui augmente la monnaie courante de la monnaie contenue dans le paramètre. *Exemple :*

monnaie à ajouter	monnaie courante	
	avant ajout	après ajout
nombre de billets de 10 = 0	nombre de billets de 10 = 5	nombre de billets de 10 = 5
nombre de billets de 5 = 2	nombre de billets de 5 = 5	nombre de billets de 5 = 7
nombre de pièces de 2 = 1	nombre de pièces de 2 = 5	nombre de pièces de 2 = 6
nombre de pièces de 1 = 4	nombre de pièces de 1 = 5	nombre de pièces de 1 = 9

6.b) Et sur le même modèle, `public void retirer(Monnaie àRetirer)` 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.

7) 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 :*

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

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

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

8) En vous aidant des méthodes précédemment définies, ajoutez à la classe `Monnaie` une procédure `encaisser` dont le rôle est de gérer un achat par un client. Le client donne de la monnaie pour payer le montant qui lui est demandé, vous devez encaisser ce paiement et calculer la monnaie à rendre si nécessaire. Cette procédure possède :

- un paramètre de type `Monnaie` représentant le paiement du client ;
- un paramètre de type entier représentant le montant à payer.

Cette procédure modifie la monnaie courante pour tenir compte du paiement du client et de la monnaie rendue en fonction du montant à payer. Elle affiche aussi le détail de la monnaie rendue au client. N'oubliez pas de tester cette procédure. *Exemple :*
Soit la situation avant encaissement :

monnaie courante	paiement	montant à payer
nombre de billets de 10 = 0	nombre de billets de 10 = 3	35
nombre de billets de 5 = 0	nombre de billets de 5 = 2	
nombre de pièces de 2 = 0	nombre de pièces de 2 = 3	
nombre de pièces de 1 = 1	nombre de pièces de 1 = 1	

Après encaissement :

monnaie courante	monnaie rendue
nombre de billets de 10 = 2	nombre de billets de 10 = 1
nombre de billets de 5 = 2	nombre de billets de 5 = 0
nombre de pièces de 2 = 2	nombre de pièces de 2 = 1
nombre de pièces de 1 = 2	nombre de pièces de 1 = 0

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

```
int montant;

//définition de deux variables, de noms ''monnayeur'' et
//''paiement'', de type Monnaie, initialisées à des valeurs
//différentes.
...

//affichage du détail de monnayeur
...

//lecture sur l'entrée standard du montant à payer
System.out.print("montant␣=␣");
...

//encaissement dans monnayeur du paiement pour le montant
//donné
...
```



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