Université Paris Diderot
Licence Informatique

Programmation Orientée Objet
Année 2009-2010

TP noté n°3

On veut réaliser un mini-système permettant de calculer le prix de pièces produites par une entreprise. Chaque pièce possède un nom unique qui permet de la repérer.

Les pièces produites sont de deux types : les *pièces de base* et les *pièces composées*.

Les pièces de base

Les pièces de base sont indivisibles. Ce sont des objets de la classe `PieceBase`. Une pièce de base possède un nom et un prix susceptible de modification. L'affichage d'une pièce de base donne son nom, son type (`PIECE_BASE`) et son prix. Dans l'exemple donné ci-dessous, une pièce de nom "*boite*" est de type `PIECE_BASE` et son prix est 5 euros.

Les pièces composées

Les pièces composées sont obtenues par l'assemblage d'autres pièces (de base ou composée). Ce sont des objets de la classe `PieceComposee`. Une pièce composée possède un nom, l'ensemble des pièces qui entrent dans sa composition et un type (`PIECE_COMPOSEE`).

Le prix d'une pièce composée est calculé selon les pièces qui entrent dans sa composition. L'affichage d'une pièce composée donne son nom, son type, son prix et la composition détaillée. Dans l'exemple ci-dessous, un jouet se compose d'une boite (5 euros), de 4 ressorts (1.5 euros chacun) et d'un guignol (6 euros). Son prix est de 17.0 euros.

La classe Piece

La classe `Piece` est une classe abstraite qui forme la classe de base des classes `PieceBase` et `PieceComposee`. Elle réunit les éléments communs aux deux classes `PieceBase` et `PieceComposee`. Déterminez les attributs et les méthodes de cette classe.

Travail à faire

1. Définissez la classe abstraite `Piece` qui possède les attributs et les méthodes communs aux deux classes `PieceBase` et `PieceComposee`. Quelques remarques :
   - Elle possède la méthode abstraite `estUn` qui retourne le type (`PIECE_BASE` ou `PIECE_COMPOSEE`) de la pièce.
   - Elle implémente la méthode `equals(Piece p)` qui retourne si la pièce a le même nom que la pièce en argument de la méthode.
   - Elle implémente la méthode `toString` qui affiche son nom, son type et son prix.
   - Elle possède la méthode abstraite `setPrix(double prix)` qui modifie son prix si la pièce est du type `PIECE_BASE`. Sinon, elle lance une exception informant que l'on ne peut modifier que le prix d'une pièce du type `PIECE_BASE`.
2. Définissez la classe `PieceBase` dérivée de la classe `Piece`. On crée une pièce de base avec son nom et son prix.

3. Définissez la classe `PieceComposee` dérivée de `Piece`. Pour chaque pièce entrant dans la composition d'une pièce composée, on dispose du nombre d'exemplaires nécessaires pour produire la pièce composée. Ainsi par exemple un *jouet* est une pièce composée constituée d'un guignol (pièce de base), de 4 ressorts (pièce de base) et d'une boîte (pièce de base). Un *paquet* est une pièce composée constituée d'une caisse, et de 10 jouets.

   Vous définirez une classe imbriquée `Composant` pour gérer un tel composant. Elle regroupe une pièce et son nombre d'exemplaires nécessaire. On peut modifier la composition d'une pièce composée en lui ajoutant ou en lui retirant un composant. La classe `Composant` possède la méthode `contenir(Piece p)` qui retourne si sa pièce est égal à la pièce en argument de la méthode, et la méthode `toString` qui affiche sa pièce avec son nombre d'exemplaires.

   On crée une pièce composée avec son seul nom et le nombre maximum de pièces différents (c'est-à-dire, le nombre maximum de composants) qu'elle possède.

   La classe possède les méthodes :
   - Méthode `ajouter` qui ajoute n exemplaires d'une pièce à la composition. Si la pièce existe déjà dans la composition, il faut augmenter son nombre d'exemplaires par n. La méthode lance une exception si le nombre de composants dépasse le nombre maximum de pièces différents que la pièce composée peut posséder.
   - Méthode `oter` qui supprime n exemplaires d'une pièce de sa composition. Si la pièce existe déjà et son nombre d'exemplaires est inferieur à n, on supprime le composant. La méthode lance une exception si la pièce n'existe pas dans la composition.
   - Méthode `entreDansComposition` qui retourne si une pièce argument de la méthode entre dans la composition d'une pièce composée.
   - Méthode `toString` qui permet un affichage détaillé de la pièce composée.
4. Ecrivez une classe `Entreprise` dans laquelle vous testez et montrez la fonctionnalité des méthodes et des messages d'information (exceptions) que vous avez écrite.
5. Vous avez un exemple d'exécution avec 4 pièces de base : *boite, ressort, guignol, caisse* et 2 pièces composées : *jouet* et *paquet*. Ainsi un jouet se compose d'une boite, de 4 ressorts et d'un guignol. Un paquet se compose d'une caisse, et de 10 jouets. Dans l'affichage, la composition détaillée de chaque composant apparaît. Ainsi la composition de jouet dans l'affichage du paquet apparaît. Prenez soin dans l'affichage de décaler de quelques espaces les éléments qui entrent dans une composition (on peut se servir d'une variable statique pour le décalage d'espace).