CNAM

M-NFA032 Algorithmique et programmation : bibliothèques et patterns

Deuxième partie

Travail à effectuer

Rappels

Quelques éléments importants avant de commencer le fil rouge :

- à terme, votre programme comportera une interface graphique ; cette interface ne vous permettra pas de lire au clavier ni d'écrire à l'écran ; donc, ne pas mettre d'ordre de lecture ou d'écriture dans vos classes ; si vous souhaitez valider vos classes, prévoyez alors une petite bibliothèque temporaire pour les entrées-sorties.
- Dans vos classes, mettez systématiquement vos champs private ou, à la limite, protected ; vous pourrez ensuite donner des accès à vos variables d'instances via des getters et des setters.

Exercice 1 : Equals et hashCode dans Produit

Les produits peuvent être identifiés uniquement par la référence. Afin d'avoir une égalité en profondeur, il faut redéfinir equals et hashCode dans la classe Produit.

Méthode sous Eclipse

1. Ouvrez, si ce n'est pas déjà fait, la classe Produit
2. Créez les deux méthodes : Menu Source > Generate hashCode and equals
    1. cliquez Deselect All pour tout désélectionner
    2. cochez reference pour n'utiliser que cette variable d'instance dans les deux méthodes
    3. Dans Insertion point : choisissez où vous voulez insérer les deux méthodes. Last member n'est pas un mauvais choix.
    4. Cochez au besoin Use instanceof to compare types
    5. Ok

Exercice 2 : Gomme

Une gomme n'a pas de propriété particulière. Construire une classe Gomme simplement héritière de la classe Produit. Notons toutefois que le type d'une gomme est GOMME !

Méthode sous Eclipse

1. Dans la fenêtre « Package Explorer », sélectionnez le package produits
2. On va créer Gomme comme classe héritière de Produit : créez une nouvelle classe (File>New>Class)
    1. assurez-vous que dans Package il y ait bien produits
    2. Name => Gomme
    3. Pour choisir la classe « Produit » comme classe mère de Gomme, sur la ligne « Superclass » cliquez sur le bouton « Browse »
        1. Dans « Choose a type » remplacez « java.lang.Object » par Produit
        2. Dans « Matching items » sélectionnez la ligne « Produit produits »
        3. « Ok »

        Dans « Superclass » vous devez avoir produits.Produit
    4. Cochez la boîte à cocher « Constructors from superclass » afin qu'Eclipse vous insère automatiquement un constructeur compatible avec celui de Produit.
    5. « Finish » Vous devez obtenir :

```
package produits;

public class Gomme extends Produit {

	public Gomme(String marque, double prix, String intitule,
			String description, TypeProduit type,
			Object... complement) {
		super(marque, prix, intitule, description, type, complement);
		// TODO Auto-generated constructor stub
	}

}
```

On va maintenant adapter le constructeur.

3. Supprimer la ligne contenant « TODO ». Il n'y a rien d'autre à faire pour ce constructeur.
4. On va ajuster la liste des paramètres du constructeur.
    1. Paramètre type

1. supprimez `TypeProduit type,` . Le type d'article étant déterminé, il n'est pas nécessaire de le prendre en paramètre.

   Une erreur doit apparaître sur type dans la ligne commençant par « super ». C'est normal puisque nous venons de supprimer la définition de type.

2. Dans la ligne commençant par « super », remplacer type par `TypeProduit.GOMME` . On force le type de produit à être une gomme...

2. Paramètre complement

   1. Toujours dans la liste des paramètres du constructeur, supprimez `Object... complement` et la virgule après `String intitule,`. Ce paramètre est destiné à recevoir les paramètres spécifiques de ce type d'articles. Comme il n'y en a pas, il n'est pas utile.

      De nouveau une erreur doit apparaître sur complement dans la ligne commençant par « super ». C'est normal puisque nous venons de supprimer la définition de type.

   2. Supprimez le paramètre complement de l'appel à super.

N'ayant pas de paramètre spécifique, le suffixe de la référence pour une Gomme sera une chaîne vide.

5. Ouvrez TypeProduit
6. Créez si ce n'est pas déjà fait les méthodes getSuffixe pour chacune des constantes
7. Dans Gomme
   1. enlevez la ligne TODO
   2. remplacez return null par return "" (deux guillemets sans espace entre).

Exercice 3 : Lots

Créez la classe Lot comme Gomme. Nous la compléterons plus tard. Comme pour Gomme, le suffixe de la référence d'un lot sera vide.

Exercice 4 : Crayons

Créons maintenant Crayons

Méthode sous Eclipse

1. Procédez comme pour Gomme et arrêtez-vous après l'étape 4.1 Paramètre type. Pensez qu'il s'agit d'un crayon cette fois-ci.

   Contrairement aux gommes les crayons ont un paramètre spécifique : leur

dureté. On va le créer pour les crayons.

Dans un premier temps, nous allons créer une énumération membre de la classe Crayon pour la dureté car la liste des valeurs est finie. On aurait pu aussi définir cette énumération comme TypeProduit mais voyons ce nouveau concept.

Attention ! Le nom des constantes que nous allons mettre dans l'énumération ne peut pas commencer par un chiffre. Afin que les valeurs soient homogènes, nous les précéderons d'un _.

2. Cliquez sur la ligne qui suit la définition de la classe Crayon
3. Faites File > New > Enum
    1. nommez l'énumération DureteCrayon
    2. cochez « Enclosing Type » et vérifiez que vous avez bien produits.Crayon dans le champ à droite afin que la définition soit incluse dans Crayon.
    3. Finish
4. Afin de faciliter l'utilisation de ce type, il vaut mieux ajouter static devant enum.
5. A l'intérieur de l'enum, listez les valeurs séparées par des virgules : _4B, _2B, _B, ...
6. Ajoutez une variable d'instance durete de type DureteCrayon après l'enum, par exemple. Ce champ pourra être private et final.
7. Créez l'accesseur à cette variable.

Maintenant, le constructeur est souligné en rouge car le champ durete n'est pas initialisé.

8. Dans la liste des paramètres du constructeur, remplacez complement par durete de type DureteCrayon
9. Dans la ligne commençant par super, remplacez complement par durete.
10. Ajoutons après la ligne commençant par super la ligne d'initialisation de la variable d'instance durete : `this.durete = durete;`

Nous en avons fini avec Crayon. Définissons le suffixe pour un Crayon.

Lors de son appel dans le constructeur de Crayon (ligne commençant par super), le constructeur de Produit recevra un tableau d'Object ne contenant qu'un élément : la dureté du crayon. C'est ce tableau qui sera transmis à la méthode getSuffixe pour la valeur CRAYON de TypeProduit. Nous allons simplement retourner les deux derniers caractères du texte correspondant comme suffixe.

11. Ouvrez TypeProduit et allez à la méthode getSuffixe.
12. Remplacez la ligne TODO par : `DureteCrayon durete = (DureteCrayon)`

`complement[0];` . DureteCrayon doit être souligné en rouge deux fois.

Cette ligne a pour but d'interpréter le premier élément du tableau comme un objet de type DureteCrayon. Ceci n'est possible que dans la mesure où l'objet converti est vraiment une instance de DureteCrayon ou d'une classe héritière.

13. Pour corriger les deux erreurs, faites Ctrl-Maj-O pour mettre à jour les import.

Ceci a pour effet d'ajouter une ligne `import produits.Crayon.DureteCrayon;` au début du fichier après la ligne package.

14. Dans la ligne return, remplacez null par durete.toString().substring(1).

Exercice 5 : Stylos et cahiers

Par un procédé similaire, créez les classes Stylo et Cahier.

Pour les stylos, le type de stylo pourra être un type énuméré comme pour la dureté des crayons. La couleur pourra être un simple texte. Le suffixe sera formé de la première lettre du type de stylo suivi des deux premières lettres de la couleur.

Quelques indications :

- Le constructeur de Stylo devra prendre deux paramètres à la place de complément. Ces deux paramètres seront transmis à la méthode getSuffixe de Stylo sous forme de tableau de deux Object via le constructeur de Produit. Notez bien l'ordre dans lequel vous transmettez ces deux paramètres car il est important pour exploiter le tableau dans getSuffixe : le premier élément du tableau contiendra la valeur du premier des deux paramètres, et le second élément du tableau, le second paramètre.
- Pour récupérer le premier caractère du type de stylo, vous pouvez utiliser la méthode charAt de la classe String : type.toString().charAt(0)
- Pour les deux premiers caractères de la couleur : couleur.substring(0,2).toUpperCase() ;
  0 à partir du caractère d'indice 0, 2 jusqu'au caractère d'indice 2-1 = 1

Les cahiers ont 6 paramètres spécifiques : la couture, la couleur, le carroyage, le grammage, les dimensions et le nombre de feuilles.