CNAM

M-NFA032 Algorithmique et programmation : bibliothèques et patterns

Huitième 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.

Thème de la semaine

Cette semaine, nous poursuivons l'interface graphique de l'application par un formulaire de saisie d'un produit (Cf. Illustration 1: Formulaire de saisie d'un produit)

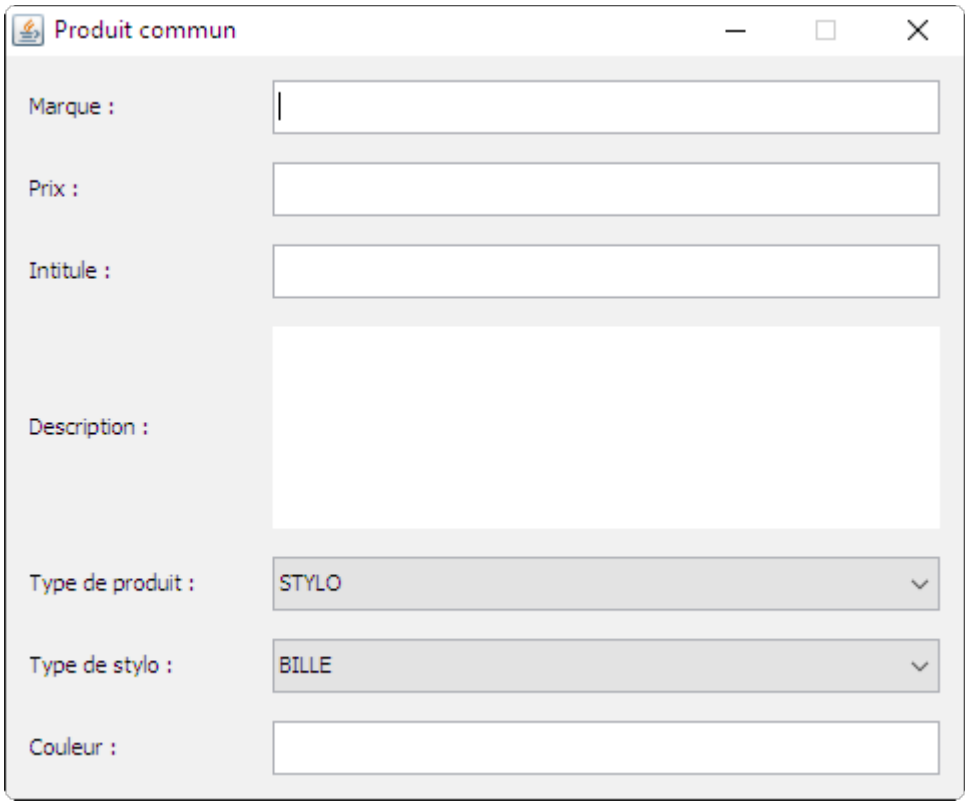

Illustration 1: Formulaire de saisie d'un produit

Le formulaire sera un panneau (un JPanel) avec deux sous-panneaux (des JPanel) :

- le premier destiné à recevoir les paramètres communs à tous les produits (marque, prix, intitulé, description et type de produit),

- le second destiné à recevoir les paramètres spécifiques à chaque type de produit.

La seconde partie de ce formulaire, la partie spécifique, s'adaptera en fonction du type de produit choisi par l'utilisateur.

Exercice 1 : Paramètres communs

Pour l'instant, nous nous contentons de construire le panneau. Nous ajouterons des méthodes à cette classe par la suite.

1. Créer un package ui.produit. Tout le travail de la semaine se fera dans ce package.
2. Créez « ProduitCommunJPane » une classe héritière de JPanel.
3. Créer son constructeur pour obtenir le résultat montré en Illustration 2: Partie commune à tous les produits.

Illustration 2: Partie commune à tous les produits

Quelques conseils

- Utilisez un GridBagLayout.
- Créez une variable d'instance dans la classe pour chaque zone active.
- Pour le type de produit, utilisez une JComboBox<TypeProduit> et veillez à initialiser le contenu avec TypeProduit.values(). Ainsi, la liste sera initialisée avec la liste des valeurs définies dans TypeProduit. Pour l'affichage des valeurs, c'est la méthode toString() portée par les objets qui sera utilisée.
- Afin de faciliter le travail ultérieur, nommez « typeProduitJC » la variable d'instance destinée à désigner la liste de choix du type de produit car nous y ferons référence plus loin.

Exercice 2 : Factorisation des parties spécifiques

Pour faciliter le travail par la suite, nous allons créer une classe abstraite qui implémentera tout ce qui est commun aux panneaux des paramètres spécifiques. Notamment, elle permettra de récupérer ou de modifier la valeur des champs de la partie commune à tous les produits, et de gérer l'insertion de lignes dans le formulaire.

Pour l'instant, cette classe se contente d'initialiser une variable d'instance permettant d'identifier le panneau contenant les informations communes et d'offrir un service d'ajout d'une ligne (étiquette et champ de saisie). Le reste de la classe sera développé par la suite.

1. Créer une classe abstraite nommée « AbstractOptionJPane » héritière de JPanel.

2. Prévoir une variable d'instance « final » nommée « communJP » de type « ProduitCommunJPane ». Cette variable nous permettra plus tard plus tard de récupérer facilement les valeurs des champs communs à tous les produits.

3. Prévoir une seconde variable d'instance « gbc » privée de type « GridBagConstraints »

4. Créer un constructeur prenant en paramètre la valeur de « communJP », utilisez un GridBagLayout pour l'arrangement du JPanel, initialisez gbc avec une nouvelle instance de « GridBagConstraints ». Initialisez les différents champs utiles de gbc.

5. Créer une méthode protected nommée addLine pour ajouter une ligne au panneau courant. Cette ligne sera formée d'un JLabel dont le contenu de type String sera pris en paramètre et d'un champ de saisie lui aussi pris en paramètre. Vous pourrez utiliser le type JComponent pour typer le champ de saisie. De plus, cette méthode devra gérer la variable d'instance gbc et surtout, avant de sortir, elle devra en modifier les champs gridx et gridy pour que la prochaine insertion se fasse en début d'une nouvelle ligne.
   Cette méthode est déclarée protected car on souhaite que seules les héritières de la classe aient le droit d'ajouter une ligne à ce formulaire.

Exercice 3 : Parties spécifiques du formulaire

L'objectif de cet exercice est de créer une classe pour la partie spécifique de chaque type de produit pour le formulaire de saisie des produits. A chaque fois, il s'agira de créer une extension de « AbstractOptionJPane » et d'utiliser des variables d'instance pour conserver la référence vers les diverses zones actives qui seront créées dans le constructeur afin de retrouver facilement ce que l'utilisateur aura saisi.

Gomme

Créez une classe « GommeJP » avec un constructeur prenant en paramètre la valeur de « communJP » à transmettre au constructeur de « AbstractOptionJPane ». Comme il n'y a pas de paramètre spécifique, il n'y a rien de plus à faire dans ce panneau.

Lot

Nous allons laisser en attente ce panneau. Contentez-vous de procéder comme pour Gomme : un panneau vide.

Crayon

Procédez comme pour une gomme, puis pensez à ajouter une JComboBox pour la dureté en utilisant la méthode addLine de l'exercice précédent.

Stylo

Toujours le même départ, puis pensez aux deux paramètres : type de stylo et couleur.

Cahier

De nouveau le même départ ! Pensez aux 6 paramètres.

Exercice 4 : Création de la partie variable en fonction du type de produit choisi

Nous allons utiliser « TypeProduit » comme fabrique de la partie variable du formulaire.

1. Dans « TypeProduit », définir une méthode abstraite « newOptionJPane » de type

« AbstractOptionJPane » et prenant un paramètre un « ProduitCommunJPane ».
2. Spécialisez cette méthode pour chaque valeur de l'énumération afin qu'elle retourne une nouvelle instance du sous-formulaire correspondant au type de produit.

Exercice 5 : Le formulaire

1. Créez une extension de JPanel « ProduitBodyJPane » et implémentant « ActionListener »[1], avec uniquement deux composants :
   - un « ProduitCommunJPane » qui sera mémorisé dans la variable d'instance « communJP »,
   - un « AbstractOptionJPane » qui sera mémorisé dans la variable d'instance « optionJP ».

Pour la partie variable, choisissez au départ un sous-formulaire pour la première valeur de « TypeProduit ». Normalement ce devrait être un stylo, sinon adaptez.

Exercice 6 : Adaptation de la partie variable

1. Dans la classe Systeme construite lors de la semaine dernière, ajoutez un accesseur getMf sans paramètre et static qui retourne la valeur de la variable statique mf.
2. Au moins après l'instruction qui crée la partie commune du formulaire dans le constructeur de « ProduitBodyJPane », ajoutez la ligne ci-dessous qui a pour but de définir l'instance courante de « ProduitBodyJPane » comme observateur de la JComboBox du type de produit. La méthode « addTypeProduitChangeListener » n'existe pas encore. Nous la définirons bientôt. Cet exercice fini, chaque fois que l'utilisateur changera de type de produit, la méthode « actionPerformed » que nous définirons à l'étape suivante sera appelée. Celle-ci aura pour fonction de :
   - supprimer la partie variable actuelle du formulaire,
   - récupérer le nouveau type de produit sélectionné par l'utilisateur,
   - créer une partie variable en fonction du type de produit grâce à la classe « TypeProduit »,
   - insérer à la bonne place la nouvelle partie variable dans le formulaire,
   - redimensionner la fenêtre au mieux.

```
communJP.addTypeProduitChangeListener(this);
```

3. Ajoutez la méthode ci-dessous à « ProduitBodyJPane ». La méthode « getSelectedTypeProduit » sera définie « ProduitCommunJPane » lors de la prochaine étape.

```
@Override
// méthode permettant de changer la partie variable du formulaire quand
// l'utilisateur change de type de produit.
public void actionPerformed(ActionEvent event) {
      // on détruit la partie variable actuelle du formulaire.
      if (optionJP != null) {
            remove(optionJP);
      }

      // on récupère le type de produit selectionné par l'utilisateur.
      TypeProduit tp = communJP.getSelectedTypeProduit();
```

1 De nombreux éléments graphiques utilisent le patron de conception « Observateur ». Ceci permet d'exécuter un code spécifique quand un événement particulier arrive sur cet élément graphique. Ici, il s'agit de modifier la partie variable du formulaire quand l'utilisateur change de type de produit.

```
            // on crée la partie variable qui correspond au choix de
l'utilisateur.
            optionJP = tp.newOptionJPane(communJP);

            // pour positionner correctement la partie variable du formulaire en
son
            // sein.
            GridBagConstraints gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 1;
            gbc.anchor = GridBagConstraints.WEST;

            // on ajoute la partie variable au formulaire.
            add(optionJP, gbc);

            // on adapte la fenêtre au nouveau contenu.
            Systeme.getMf().pack();
      }
```

4. Dans la classe « ProduitCommunJPane » ajoutez les deux méthodes suivantes :

```
      // méthode permettant d'enregistrer le formulaire comme observateur de la
      // liste déroulante.
      public void addTypeProduitChangeListener(ActionListener listener) {
            // une simple délégation vers la JComboBox
            // typeProduitJC est la variable d'instance qui désigne la JComboBox
            // adaptez-le nom de la variable au besoin
            typeProduitJC.addActionListener(listener);
      }

      // méthode qui permet de récupérer le type de produit sélectionné par
      // l'utilisateur
      public TypeProduit getSelectedTypeProduit() {
            // de Nouveau une simple délégation avec un casting de type
            // Notez getSelectedItem qui retourne l'objet correspondant au choix
de
            // l'utilisateur
            return (TypeProduit) typeProduitJC.getSelectedItem();
      }
```

Exercice 7 : Un main de test

Concevoir une fenêtre ne contenant que le formulaire de saisie d'un produit.