Génie logiciel CB2
TP2 individuel noté
Suite et hiérarchie de cartes

Jean-Claude Royer

2 janvier 2011

Il s'agit dans ce TP d'appliquer le patron composite pour faire des suites de cartes dans un jeu de 32 cartes. Il faut pouvoir créer des séquences de cartes et les trier. Pour simplifier vous pouvez supposer que les séquences ne contiennent pas de carte en double (même valeur et même couleur). Les objectifs de ce TP sont :

- Estimer votre maîtrise des outils : ArgoUML et Eclipse.
- Estimer votre modélisation d'un problème simple et sa conception en Java.
- Estimer votre compréhension du patron composite et de la redéfinition de méthodes.

Votre travail est découpé de la façon suivante :

1. Modéliser le diagramme des classes avec ArgoUML
2. Générer le code Java correspondant
3. Créer un nouveau projet Eclipse et importer votre code
4. Compléter et tester votre code

1 Modélisation avec ArgoUML

La première phase est découpée en étapes :

1. Modéliser, avec ArgoUML, la classe `Couleur`.
2. Modéliser, avec ArgoUML, la classe `Valeur`.
3. Modéliser, avec ArgoUML, la classe `Carte` avec une couleur et une valeur.
4. Définir une égalité, `equals`, pour les couleurs.
5. Définir une comparaison, `compareTo`, pour les valeurs qui rendra -1, 0, 1 suivant la valeur de la comparaison (respectivement `<`, `==` et `>`).
6. Modéliser, avec ArgoUML, la classe `Main` en utilisant le patron composite, une main représente une séquence de carte dans l'ordre ou elles ont été distribuées. Les deux sous-classes seront nommées `SansCarte` et `AvecCarte`.
7. Les opérations abstraites pour `Main` sont :
   - `taille : Main -> int` : calcule le nombre total de cartes.
   - `inserer : Main Carte -> Main` : insère dans une main triée une carte et en respectant l'ordre croissant des valeurs.

- `tri : Main -> Main` : trie dans l'ordre croissant des valeurs la main.
- `estTriee : Main -> boolean` : teste si la séquence est dans l'ordre croissant des valeurs.

8. Vous devez accorder un peu de soin dans la conception de votre diagramme de classe mais également placer correctement les méthodes et les redéfinir si besoin dans la hiérarchie.
9. Faites la génération de code de votre modèle.
10. Produire un diagramme de classes au format png.

Notez que dans les signatures données ci-dessus la classe de l'objet receveur est le premier type d'argument, et qu'en Java ce premier type est celui de la classe de déclaration et est implicite dans la signature de la méthode.

2 Conception sous Eclipse

Quelques indications pour vous aider dans l'écriture des méthodes.

1. Réfléchissez à votre modèle, et avant de générer le code, assurez vous que le dessin, les noms, les types des signatures Java, les paramètres, les redéfinitions sont corrects.
2. Une version en programmation fonctionnelle pure est plus facile à réaliser, mais une version dans un style impératif sera correcte (mais plus délicate à écrire).
3. Une fois votre code généré et importé sous Eclipse il est important de vérifier que vous savez représenté les séquences. Ceci signifie que vous avez défini le (ou les) constructeur adéquat et que vous devez les testez.
4. Essayez ensuite d'écrire quelques méthodes simples comme `toString` et surtout testez les.
5. Pour les méthodes `equals, compareTo` vous pouvez faire des redéfinitions des méthodes existantes dans `Object` mais ce n'est pas nécessaire. Pour la comparaison une solution simple est d'attribuer un entier à chaque valeur.
6. Les méthodes `taille` et `inserer` ne devraient pas vous poser de problème.
7. Pour le tri il faut utiliser la méthode `inserer` pour réaliser récursivement un tri dit par insertion.
8. Pour la méthode `estTriee` penser aux différents cas de longueur de la séquence. Vous avez besoin d'accéder au deuxième élément d'une main `m`, si on sait que celui-ci existe, alors la conversion de type (cast) `(AvecCarte) (m.suite).premier` est correcte et calcule cet élément. Il s'agit d'un problème similaire à celui vu dans le TP1 avec `putLast()`.
9. Ecrivez et exécutez des tests en vous basant sur les propriétés suivantes :
   (a) `m.estTriee() AND m.inserer(c).estTriee()` doit être vrai quelque soit la carte `c` et la main `m`.
   (b) `m.tri().estTriee()` doit être vrai quelque soit la main `m`.
10. Implémentez le patron singleton dans votre application.

3 Le travail à rendre

Vous devez produire un projet Java bien structuré, commenté et contenant le résultat de votre conception ainsi que le diagramme de classes sous format `png`. Vous fournirez une archive `.zip` de votre projet, avec les sources et ressources, que vous déposerez sur campus. Vous n'êtes pas obligé de rendre le fichier `.zargo`.