Projet de CDAA
Système de gestion d'entreprises de distribution

Eliot Morvan
Licence 3 Informatique TP5

Novembre 2018

Contents

Listings

Les codes présentés servent d'exemple afin d'illustrer les propos rapidement, mais l'intégralité du code étant disponible, il ne sera pas détaillé et expliqué en détail dans ce rapport

1 Application POO

L'application proposée pour ce projet de CDAA permet la gestion de commandes de produits par différentes entreprises. Le système gère à la fois plusieurs type d'entreprise, le panier de commande de chaque entreprise ainsi que les différents type de produits. L'exposition et l'explication des spécifications fonctionnelles se fera en trois parties, corréspondant aux trois principaux package du système. La première partie se focalisera sur le package entreprise, avec la description et le fonctionnement d'une entreprise dans le système, la deuxième partie sur les différents produits, et enfin la gestion des différents paniers avec le lien effectué entre entreprise et produit.

1.1 Package Entreprise

Une entreprise telle que définie dans ce système est définie principalement par un nom nom d'entreprise, une date de fondation, un fondateur et une adresse mail. Chaque entreprise possède également une liste de catégorie de vente. Les catégories sont donc différentes selon les entreprises et peuvent être modifiées à tout moment. Enfin, une entreprise possède un panier de commandes qui lui est propre et unique. La gestion du panier permet d'ajouter un produit à tout moment, d'en retirer une entité précise ou tout un type d'un coup.

1.2 Package Produit

Il existe énormément de produits dans les domaines de papeterie, maroquinerie, etc... Par soucis de clareté, trois produits principaux ont été créés. L'objectif principal étant de démontrer la possibilité d'en créer un très grand nombre sans surcharger inutilement la conception. Chaque produit possède un prix, une marque et une catégorie. Un ID sera par la suite généré pour chaque type produit. Les classes Agenda, Bagage, Feutre, héritent des fonctionnalités de Produit et ajoutent leur propres attributs. On a ainsi Agenda possédant l'année qu'il couvre, ses dimensions (en cm), une couleur, et peut être un agenda ligné et/ou quadrillé. Bagage possède une capacité (en L), une couleur et peut être un bagage quatre-roues. Enfin Feutre possède une épaisseur (en mm), et une couleur d'encre. Peu d'opérations sont effectuées sur les produits, ils ne servent que de type plus complexe pour la gestion du panier.

1.3 Package Panier

Le panier est défini tel qu'étant une liste de produits. Lors de la création du panier par l'entreprise, une taille maximum peut être allouée de façon à ce qu'aucun produit ne puisse être ajouté si le panier est plein. Il est possible d'ajouter, de supprimer des éléments du panier ainsi que de trier les éléments existants du panier. L'utilisateur peut également pouvoir récupérer le nombre de places restantes dans le panier à tout moment.

1.3.1 Elements clés du système

1.3.2 Classes Enum

Les différentes catégories sont définies dans une classe Enum Categorie et sont; papeterie (carnet, agenda, etc..), maroquinerie (bagage, trousses, etc...) et fourniture (stylos, gomme, etc...). Idem pour les couleurs de la couverture d'agenda. La création de ces classes permet d'empécher les incohérences durant la manipulation des entités. Elles sont applicables à bien d'autres attributs comme les marques ou les noms d'entreprises, mais servent d'exemple global pour le système, une fois encore, afin de ne pas ajouter de surcharges inutiles.

1.3.3 Génération de l'ID

L'ID d'un produit un texte représentant chaque type d'élément possible. Il n'est pas propre à chaque instance mais différe d'un produit à un autre pour chaque attribut différant. Par exemple, on calcule l'ID de l'agenda défini ci-dessous:

Listing 1: Instanciation d'un type Agenda

```
1 Agenda agenda1 = new Agenda(20, "Moleskine", Color.rouge,
2     "2017-2018", "24-32", false, true);
```

Le résultat sera Pr.Ag.20.M.p.2.2.r.qu

Pr - classe Produit
Ag - classe Agenda
20 - prix (les deux premiers chiffres)
M - marque (la première lettre)
p - catégorie (la première lettre)
2.2 - dimensions (les deux premiers chiffres)
r - couleur (la première lettre)
qu - est quadrillé (".l" aurait été ajouté si l'agenda était ligné)

1.3.4 Surcharge d'opérateurs

La seule surcharge d'opérateur nécessaire pour ce système est la méthode ToString() pour chaque classe. La surcharge de la méthode Equals() est dispensable puisque les seules comparaisons à effectuées sont entre deux ID de produits.

2 Généricité

2.1 Description du fonctionnement du panier

Afin de rendre cohérent le système, le panier, bien qu'étant un tableau ne découlant de collections génériques, ne dispose que d'une liste de produits. Les méthodes a testées sont appliqués directement sur le panier de l'entreprise. On peut utiliser le panier de la façon suivante:[1]

Listing 2: Ajout puis retrait de l'agenda dans le panier

```
1 //Creation d'une entite Agenda
2 Agenda agenda1 = new Agenda(20, "Moleskine", Color.rouge,
3     "2017-2018", "24-32", false, true);
4
5 //Creation du panier et ajout d'une instance agenda
6 PanierClassique panierC = new PanierClassique(20);
7 panierC.AjouterProduit(agenda1);
8
9 //Retrait de l'entite
10 panierC.RetirerProduit(agenda1);
```

L'ajout et le retrait de produits du panier sont toujours unitaires, dus au type tableau simple du panier. De plus, dans la gestion du panier, l'ajout d'un produit à la suite de ceux déja présents dans le panier requiert la manipulation d'un indice global, démarrant à 0 et s'incrémentant à chaque ajout de produit. Cette méthode permet de garder une trace des ajouts du panier mais présente un inconvéniant conséquent; lors du retrait d'un produit, une place devient vide dans le panier et il est impossible de le remplace, l'index ne prenant pas en compte les éléments vides du tableau. La solution pour remédier à ce problème est l'appel d'une fonction permettant de trier le tableau en décalant simplement les éléments de façon à combler ces vides, et de calculer l'index de ce nouveau tableau. Cette méthode est implémentée en même temps que l'algorithme de tri QuickSort().

[1] NOTE: l'exmple présenté se passe dans la version finale du système. Le panier générique sera remplacé par la suite, ce qui explique pourquoi le panier tel que décrit dans cette partie n'est plus relié à une entreprise. La manipulation du panier est toujours possible mais n'est plus lié à aucune entreprise.

2.2 Algorithme de tri QuickSort

2.2.1 Implémentation

J'ai choisi l'algorithme de tri QuickSort le considérant comme étant l'algorithme de tri le plus simple et succin sur un tableau classique C. L'inconvénient de cette méthode est que l'algorithme manipule des nombres bien plus aisément que des textes. Afin de surmonter ce problème, on transforme le panier existant en un tableau d'entier aux indices similaires. Chaque produit du panier ayant le même ID aura le même entier attribué dans le tableau converti. On peut ensuite appliquer la méthode QuickSort sur le tableau converti et appliquer chaque opération sur le panier en même temps.[2]

2.2.2 Problèmes rencontrés

Lors de la conversion du panier en tableau d'entiers, l'attribut null étant l'élément 0 et chaque nouvel ID prenait une valeur X+1 où X est la valeur du dernier entier entré. Ainsi, lors de l'exécution de QuickSort, tous les éléments null se retrouvaient en haut du tableau et le nouvel index pointait sur un élément existant du tableau. Afin de donner à null une valeur plus haute que n'importe quel entier du tableau, la solution trouvée fut de donner au tableau la valeur négative correspondante. On a ainsi null à 0 et tous les éléments du tableau converti avec des valeurs négatives, rendant le QuickSort applicable sans inconvénients.

2.2.3 Appel de la méthode

Listing 3: Exemple d'appel de la fonction de tri

```
1 //Tri du panier
2 panierC.TriParId();
```

Il n'y a aucune restriction lors de l'usage de cette fonction. L'algorithme fonctionne avec un tableau aussi bien plein que rempli, et pour un grand nombre d'ID différents. L'appel étant récursif, la fonction est plus optimale qu'avec une combinaison d'itérations et restera donc rapide dans l'ensemble indépendamment de la taille du panier.

[2] L'algorithme de tri QuickSort n'a pas directement été créé mais implémenté depuis http://snipd.net/quicksort-in-c
Seule son adaptation a été imaginée afin de rendre l'algorithme applicable dans le système.

3 Collections

3.1 Choix de la fonctionalité principale à implémenter

Afin d'implémenter l'une des fonctionalités proposées dans l'énoncé, j'ai décidé de remplacer la gestion du panier par un système de gestion ne stockant chaque type de produit qu'une fois, avec la quantité commandée. Ce type de panier correspond à un système de stockage sous forme de Dictionary, avec un clé (l'ID du produit) et une valeur(le nombre commandé). Afin de conserver les propriétés de l'énoncé précédent, il doit être possible de générer une taille maximale du panier et de récupérer son montant total à tout moment. L'implémentation de cette méthode possède l'avantage de ne pas avoir à gérer quelconque algorithme de tri, chaque élément étant automatiquement ajouté à la liste de son ID (si elle existe).

3.1.1 Utilisation

Chaque instance d'entreprise possède un panier initialisé lors de sa création. La manipulation s'effectue par le biais de l'entreprise, cela permet d'associer plus facilement le panier unique à son entreprise. La gestion des exceptions est effectuée pour tout ajout ou retrait depuis le panier. Si, lors de l'ajout d'un nombre de produits, la quantité dépasse le place restante dans le panier, l'opération ne s'effectue pas et un message est envoyé à l'utilisateur. De plus, lors du retrait d'un nombre de produits, l'ID doit exister dans le panier, et le nombre à retirer ne doit pas dépasser la quantité présente.

4 Exemple d'utilisation général

Listing 4: Programme principal

```
1 //Creation d'une entreprise distribuant toute sorte de
      categories de produits. (default)
2 Entreprise EV_All = new Entreprise("BuroVale", 1990, "B.
      Peyroles", "bp@gmail.com");
3 //Display
4 Console.WriteLine(EV_All.ToString());
5
6 //Creation d'une entreprise ne distribuant pas de maroquinerie
7 //On enleve la categorie de la liste d'activites de l'entreprise
8 Entreprise EV_NoMaroquinerie = new Entreprise("Only Papeterie
      entreprise", 2018, "K. Gillotes", "kg@gmail.com");
9 EV_NoMaroquinerie.EnleverCategoriesVente(Categorie.maroquinerie)
      ;
10 //Display
11 Console.WriteLine(EV_NoMaroquinerie.ToString());
12
13 //Creation d'une entreprise ne distribuant que de la
      maroquinerie
14 //On peut retirer toutes les categories de la liste et les
      ajouter une par une
15 Entreprise EV_OnlyMaroquinerie = new Entreprise("Only
      Maroquinerie entreprise", 2016, "H.Pottiers", "hp@gmail.com"
      );
16 EV_OnlyMaroquinerie.EnleverToutesCategoriesVente();
17 EV_OnlyMaroquinerie.AjouterCategoriesVente(Categorie.
      maroquinerie);
18 //Display
19 Console.WriteLine(EV_OnlyMaroquinerie.ToString());
20
21 //Creation d'une entite de chaque produit manuellement
22 Agenda agenda1 = new Agenda(20, "Moleskine", Color.rouge, "
      2017-2018", "24-32", false, true);
23 Console.WriteLine(agenda1.ToString());
24
25 Bagage bagage1 = new Bagage(80, "Nava", 47, true, Color.noire);
26 Console.WriteLine(bagage1.ToString());
27
28 Feutre feutre1 = new Feutre(1, "Pilot", 10, "doree");
29 Console.WriteLine(feutre1.ToString());
30
31 //Creation d'un panier lors de la creation
32 //Distribue tous les types de produit et ajout d'un produit
33 EV_All.AjouterAuPanier(agenda1, 100);
34 Console.WriteLine(EV_All.Panier.ToString());
35
36 //ajout d'un autre produit
37 EV_All.AjouterAuPanier(bagage1, 20);
38 Console.WriteLine(EV_All.Panier.ToString());
39
40 //ajout du meme produit agenda1 (expect: 150 unites d'agenda1)
41 EV_All.AjouterAuPanier(agenda1, 50);
42 Console.WriteLine(EV_All.Panier.ToString());
43
```

```
44 //ajout d'un produit en quantite trop grande pour la capacite du
       panier
45 EV_All.AjouterAuPanier(agenda1, 1000);
46 Console.WriteLine(EV_All.Panier.ToString());
47
48 //Retrait de 5 bagage1 (expect: 15 unites de bagage1)
49 EV_All.RetirerDuPanier(bagage1, 5);
50 Console.WriteLine(EV_All.Panier.ToString());
51
52 //Retrait de feutres non presents dans les produits du panier
53 EV_All.RetirerDuPanier(feutre1, 1);
54
55 //Ajout de valeurs negatives
56 EV_All.AjouterAuPanier(feutre1, -2);
57 EV_All.AjouterAuPanier(bagage1, -2);
58 Console.WriteLine(EV_All.Panier.ToString());
59
60 //Retrait de plus de bagages que de presents dans le panier
61 EV_All.RetirerDuPanier(bagage1, 60);
62 Console.WriteLine(EV_All.Panier.ToString());
63
64 //Ajout d'un produit incompatible avec le domaine de
       distribution de l'entreprise
65 EV_NoMaroquinerie.AjouterAuPanier(bagage1, 10);
66 Console.WriteLine(EV_NoMaroquinerie.Panier.ToString());
67
68 //Ajout d'un produit et verification de l'allocation memoire
69 Console.WriteLine("taille avant ajout : " + EV_NoMaroquinerie.
       Panier.GetPlacesRestantes());
70 EV_NoMaroquinerie.AjouterAuPanier(agenda1, 10);
71 Console.WriteLine(EV_NoMaroquinerie.Panier.ToString());
72 Console.WriteLine("taille apres ajout : " + EV_NoMaroquinerie.
       Panier.GetPlacesRestantes());
73
74 //Permet de maintenir l'application ouverte
75 Console.ReadLine();
```

Cette capture montre l'utilisation globale du système avec les différentes instanciations et applications possible. Bien qu'incomplet, cet exemple couvre la majorité des évènements du système, avec la majorité des méthodes de chaque classe et la gestion des exceptions sur la manipulation du panier.