Le patron de conception État

Source « https://www.math.univ-paris13.fr/~chaussar/ »

Exercice 1

On souhaite créer un distributeur de billets. Ce dernier peut être vu comme une machine d'états avec trois états : pas de carte insérée, en attente d'opération, et en attente de retirer des espèces. On pourra faire quatre actions sur la machine : insérer une carte, entrer un code, retirer des espèces, retirer la carte. Évidemment, selon l'état de la machine, les actions auront différentes conséquences.

Voici un squelette du code de la machine :

```
public class Distributeur {
    final static int etat_attente_carte = 0;
    final static int etat_attente_code = 1;
    final static int etat_attente_operation = 2;
    private int etat;
    private Carte c;

    public Distributeur() {
        etat = etat_pas_de_carte;
    }

    public inserer_carte( Carte client ) {
        if (etat == etat_pas_de_carte) {
            System.out.println("Insertion carte Client");
            c = client;
        } else if (etat == etat_entrer_code) {
            ...
        }
         ...
    }
}
```

La classe Carte est une classe avec un constructeur qui prend un code secret, et une méthode qui répond si le code entré est bon. Si le Client entre trois fois un mauvais code, sa carte doit être avalée. Codez votre Distributeur.

La machine semble bien marcher, mais certains clients se plaignent. En effet, certaines fois, la machine ne donne pas d'argent… Rapidement, les ingénieurs de la machine comprennent que lorsque la machine n'a plus de billets, et ne peut pas donner d'argent. On doit donc rajouter un état à la machine, qui sera l'état vide. Cependant, vous voyez bien que rajouter un état change beaucoup de code et n'est donc pas très pratique.

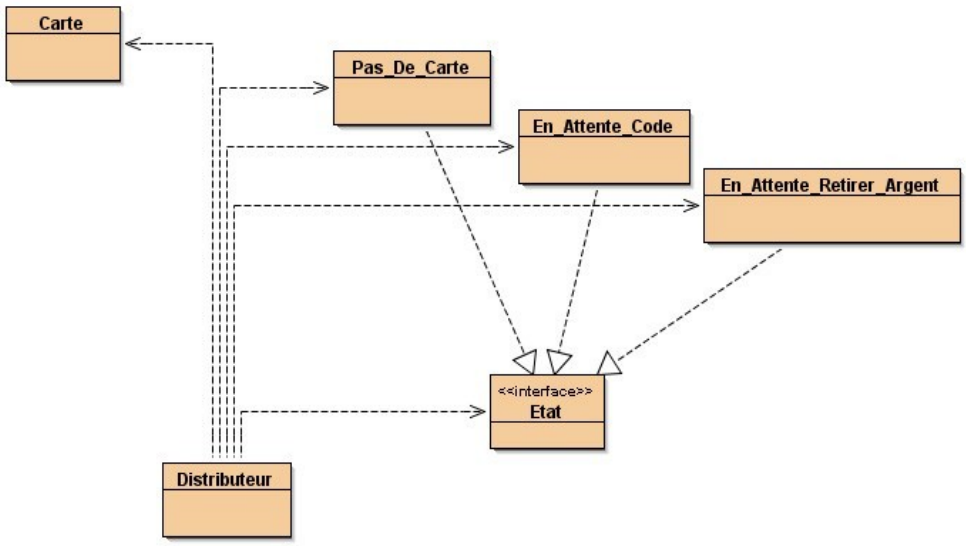
Nous allons donc, avant de rajouter l'état vide, modifier les classes et organiser mieux le projet.

Exercice 2

Créez donc une interface (ou pourquoi pas une classe abstraite…) `Etat` état qui contiendra les quatre méthodes de la machine (insérer la carte, entrer code, retirer argent, reprendre carte). Créer trois classes `Etat`, une par état possible de la machine et écrivez-y du code.

Le Distributeur devra posséder trois variables de classes (une par chaque type d'état possible), plus une variable de classe de type `Etat` qui sera son état courant. À chaque appel d'une méthode (insérer carte…), il devra appeler la méthode sur son état courant. Ce sont les états qui décideront de la marche à suivre. Cependant, le distributeur doit posséder une méthode permettant de modifier son état courant.

Pour vous aider, voici le diagramme de classe que vous devriez obtenir.



Testez avec un Client que tout fonctionne.

Exercice 3

Ajoutez un état MachineVide pour empêcher de retirer de l'argent quand la machine est vide. Vous allez donc ajouter au Distributeur une variable de classe permettant de connaître son stock d'argent, et vous vérifierez, lorsque vous donnerez de l'argent à un Client, qu'il y a assez d'espèces dans le Distributeur. Était-ce compliqué ?

Exercice 4

Ajoutez aussi une méthode `lireSoldeCompte()`, qui permettra au client d'accéder à son solde lorsque le Distributeur est en mode En_Attente_Retrait_Argent.