LaFac.com

BOSOM - CHAUSSY - DELLOUVE

20 décembre 2013

Introduction

Afin de travailler la conception de systèmes réels, nous avons créé un modélisé un pseudo-site de vente en ligne.

L'implémentation s'est faite en Java et nous devions répondre à plusieurs contraintes, à savoir un système de gestion de personnes, de produits et de différentes offres applicables.

Diagramme de classe

La classe principale de notre projet est la classe Contexte. Il s'agit d'un **Singleton**, c'est-à-dire qu'il n'en existe qu'une seule instance dans le programme. Ce contexte représente, en quelques sorte, la base de donnée que possèderait un site réel. Si nous devions nous passer de celui-ci, nous basculerions les offres employés dans le statut *Employe* qui deviendrait alors un **Singleton**. Nous ajouterions aussi un autre **Singleton** qui contiendrait les *OffreFidelites* pour les *Adherents*.

Parmi les autre **Design Pattern** utilisés nous avons le design **State** pour les statuts des *Personnes*. En effet, nous devions laisser la possibilité aux *Personnes* de changer de *Statut* en cours d'exécution, rendant ce design indispensable.

Un autre **Design Pattern** que nous utilisons est celui de l'**Observer**. Il consiste en une classe qui en "observe" une autre. Ainsi, lors d'un changement de l'instance de cette classe, l'**Observer** associé effectuera une suite d'actions prédéfinies. Dans notre projet, la classe *Alerte* est un **Observer** sur la classe *Panier*. Lors d'un changement de celui-ci, l'*Alerte* affichera un message.

Enfin nous avons fait le choix d'une liaison faible entre les *Produits* et les *Offres* afin que la modification d'une offre n'impacte pas le produit en lui même. En effet, une offre doit connaitre le produit sur lequel elle s'applique, mais le contraire n'est pas vrai. De plus, cela évite qu'un changement dans les *Offres* ne produise une erreurs ailleurs, améliorant la robustesse de notre programme.

La généricité du langage Java est utilisée lors des affichages. En effet, chaque type de *Produit* à un affichage personnalisé, tout comme les *Offres*, qui est géré lors de l'exécution du programme. Celui-ci va chercher la bonne fonction à appeler et la définir dans la classe abstraite force la redéfinition de celle-ci dans les sous-classes.

Hook (le crochet) n'est pas un DP mais un idiome. Il consiste à définir une méthode template dans la classe du haut et à laisser les sous-classes redéfinir les méthodes virtuelles qui les concernent. C'est quand même une bonne habitude de programmation qui peut être citée ici, par exemple pour implémenter l'affichage des offres en laissant chaque classe afficher les spécificités (nombre de produits, ...) qui les concernent.

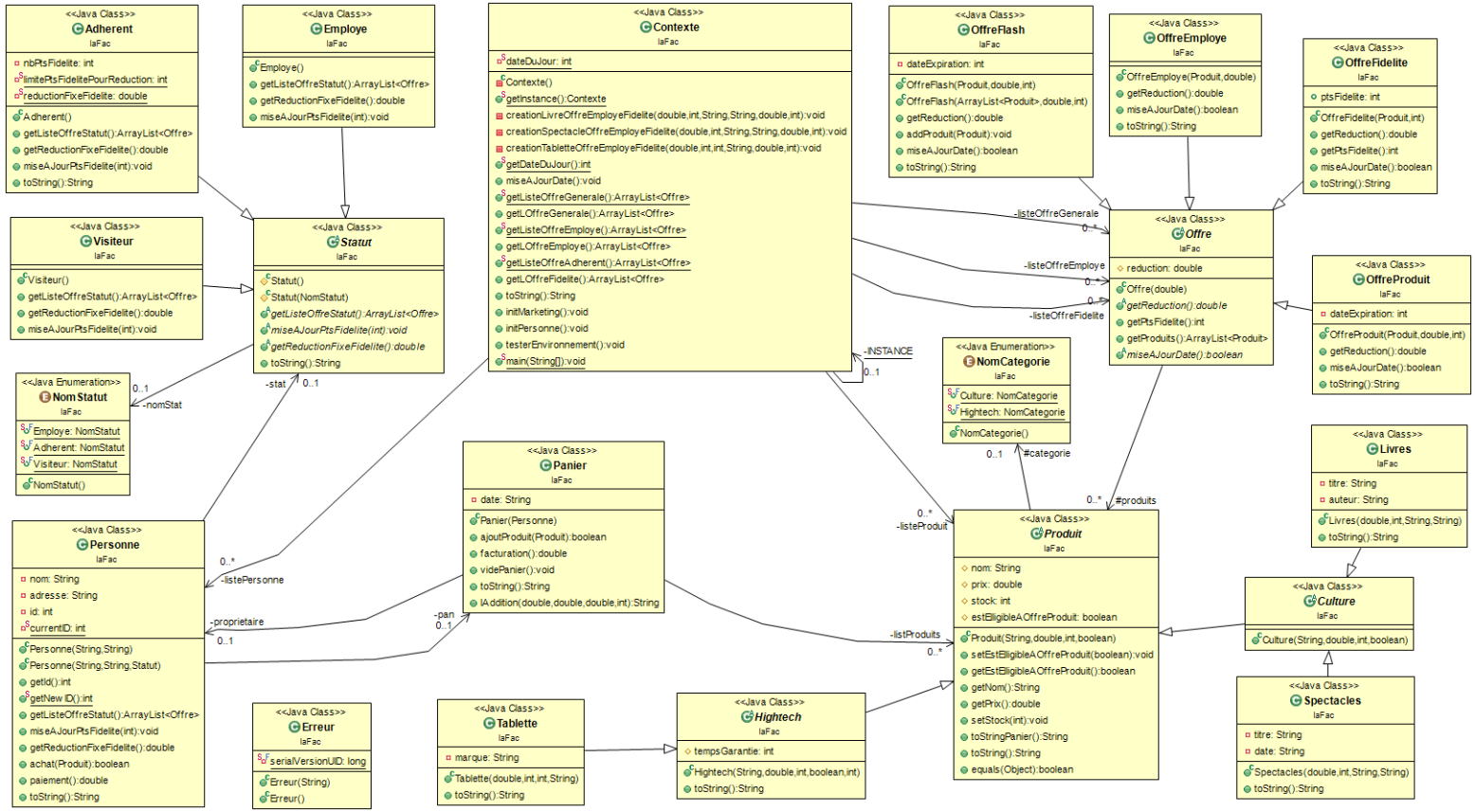

FIGURE 1 – Diagramme de classe

Diagrammes de séquences

Conclusion

Ce projet nous a permis de réfléchir à un problème concret, mettant en œuvre les connaissances acquises durant le cours.

Nous avons utilisé plusieurs **Design Pattern**, afin de répondre aux différentes contraintes du sujet, et la généricité du langage afin que les interactions entre les objets soient le plus simple et le plus clair possible, tout en offrant une possibilité d'évolution du code.

En résumé, il s'agissait d'un exercice enrichissant qui nous aura permis d'acquérir de nouvelles connaissances.

Annexe - Code source

Contexte

```java
package laFac;

import java.util.ArrayList;

/** Classe Singleton */
public final class Contexte
{
	private ArrayList<Personne> listePersonne;
	private ArrayList<Produit> listeProduit;
	private ArrayList<Offre> listeOffreGenerale;
	private ArrayList<Offre> listeOffreEmploye;
	private ArrayList<Offre> listeOffreFidelite;
	private ArrayList<Alerte> listeAlerte;
	/** Instance unique pre-initialisee */
	private static Contexte INSTANCE = new Contexte();
	private static int dateDuJour = 0;

	private Contexte()
	{
		listePersonne = new ArrayList<Personne>();
		listeProduit = new ArrayList<Produit>();
		listeOffreGenerale = new ArrayList<Offre>();
		listeOffreEmploye = new ArrayList<Offre>();
		listeOffreFidelite = new ArrayList<Offre>();
		listeAlerte = new ArrayList<Alerte>();
	}

	/** Point d'acces pour l'instance unique du singleton */
	public static Contexte getInstance()
	{
		return INSTANCE;
	}

	private void creationLivreOffreEmployeFidelite(double prix, int stock, String titre,
	    String auteur, double reductionEmploye, int ptsFidelite)
	{
		Livres l = new Livres(prix, stock, titre, auteur);
		listeProduit.add(l);
		listeOffreEmploye.add(new OffreEmploye(l, reductionEmploye));
		listeOffreFidelite.add(new OffreFidelite(l, ptsFidelite));
	}

	private void creationSpectacleOffreEmployeFidelite(double prix, int stock, String titre,
	    String date, double reductionEmploye, int ptsFidelite)
	{
		Spectacles l = new Spectacles(prix, stock, titre, date);
		listeProduit.add(l);
		listeOffreEmploye.add(new OffreEmploye(l, reductionEmploye));
		listeOffreFidelite.add(new OffreFidelite(l, ptsFidelite));
```

```
48          }
49
50          private void creationTabletteOffreEmployeFidelite(double prix, int stock, int
                tmpsGarantie, String marque, double reductionEmploye, int ptsFidelite)
51          {
52                  Tablette l = new Tablette(prix, stock, tmpsGarantie, marque);
53                  listeProduit.add(l);
54                  listeOffreEmploye.add(new OffreEmploye(l, reductionEmploye));
55                  listeOffreFidelite.add(new OffreFidelite(l, ptsFidelite));
56          }
57
58          public static int getDateDuJour()
59          {
60                  return dateDuJour;
61          }
62
63          /** Incremente la date de 1 et supprime toutes les Offres revolues. */
64          public void miseAJourDate()
65          {
66                  dateDuJour++;
67
68                  for(Offre ofr : listeOffreGenerale)
69                  {
70                          if(!ofr.miseAJourDate())
71                                  listeOffreGenerale.remove(ofr);
72                  }
73
74                  for(Offre ofr : listeOffreEmploye)
75                  {
76                          if(!ofr.miseAJourDate())
77                                  listeOffreGenerale.remove(ofr);
78                  }
79
80                  for(Offre ofr : listeOffreFidelite)
81                  {
82                          if(!ofr.miseAJourDate())
83                                  listeOffreGenerale.remove(ofr);
84                  }
85          }
86
87          public static ArrayList<Offre> getListeOffreGenerale()
88          {
89                  final ArrayList<Offre> listTmp = new ArrayList<Offre>();
90                  listTmp.addAll(Contexte.getInstance().getLOffreGenerale());
91                  return listTmp;
92          }
93
94          public ArrayList<Offre> getLOffreGenerale()
95          {
96                  return listeOffreGenerale;
97          }
98
99          public static ArrayList<Offre> getListeOffreEmploye()
100         {
```

```
101             final ArrayList<Offre> listTmp = new ArrayList<Offre>();
102             listTmp.addAll(Contexte.getInstance().getLOffreEmploye());
103             return listTmp;
104     }
105
106     public ArrayList<Offre> getLOffreEmploye()
107     {
108             return listeOffreEmploye;
109     }
110
111     public static ArrayList<Offre> getListeOffreAdherent()
112     {
113             final ArrayList<Offre> listTmp = new ArrayList<Offre>();
114             listTmp.addAll(Contexte.getInstance().getLOffreFidelite());
115             return listTmp;
116     }
117
118     public ArrayList<Offre> getLOffreFidelite()
119     {
120             return listeOffreFidelite;
121     }
122
123     public String toString()
124     {
125             String ret = new String();
126
127             ret = "ENVIRONNEMENT\n--------------------\nListe des PRODUITS : \n-----";
128             for (Produit prod : listeProduit)
129             {
130                     ret += "\n" + prod.toString() + "\n-----";
131             }
132
133             ret += "\n--------------------\nListe des OFFRES : \n-----";
134             for (Offre ofr : listeOffreGenerale)
135             {
136                     ret += "\n" + ofr.toString() + "\n-----";
137             }
138
139             for (Offre ofr : listeOffreEmploye)
140             {
141                     ret += "\n" + ofr.toString() + "\n-----";
142             }
143
144             for (Offre ofr : listeOffreFidelite)
145             {
146                     ret += "\n" + ofr.toString() + "\n-----";
147             }
148
149             ret += "\n--------------------\nListe des PERSONNES : \n-----";
150             for (Personne pers : listePersonne)
151             {
152                     ret += "\n" + pers.toString() + "\n-----";
153             }
154
```

```
155                 ret += "\n--------------------";
156
157                 return ret;
158         }
159
160         /** L'initialisation de la partie marketing consiste a initialiser les produits et
                offres. */
161         public void initMarketing()
162         {
163                 /* Initialisation des produits */
164                 creationLivreOffreEmployeFidelite(25, 10, "Toto a l'ecole des Charcutiers",
                        "Toto", 0.5, 10);
165                 creationLivreOffreEmployeFidelite(15, 2000, "Toto et Martine", "Martine", 0.5, 10);
166                 creationSpectacleOffreEmployeFidelite(50, 50, "Toto est Figaro", "12 decembre
                        2012", 0.4, 25);
167                 creationSpectacleOffreEmployeFidelite(75, 250, "Toto attend Godot", "36 avril
                        2013", 0.4, 25);
168                 creationTabletteOffreEmployeFidelite(250, 5000, 2, "Comon", 0.25, 50);
169                 creationTabletteOffreEmployeFidelite(400, 5000, 3, "Gogo Gadgeto", 0.25, 200);
170
171                 /* Initialisation des offres */
172                 /* Offre Produits */
173                 listeOffreGenerale.add(new OffreProduit(listeProduit.get(0), 0.25, 2));
174                 listeOffreGenerale.add(new OffreProduit(listeProduit.get(2), 0.25, 2));
175                 listeOffreGenerale.add(new OffreProduit(listeProduit.get(4), 0.25, 3));
176                 /* Offre Flash */
177                 OffreFlash f = new OffreFlash(listeProduit.get(0), 0.25, 2);
178                 listeOffreGenerale.add(f);
179                 f.addProduit(listeProduit.get(1));
180                 ArrayList<Produit> tmp = new ArrayList<Produit>();
181                 tmp.addAll(listeProduit);
182                 listeOffreGenerale.add(new OffreFlash(tmp, 0.25, 2));
183                 listeOffreGenerale.add(new OffreFlash(listeProduit.get(4), 0.25, 3));
184         }
185
186         /** L'initialisation des personnes faisant vivre le site */
187         public void initPersonne()
188         {
189                 listePersonne.add(new Personne("Manuel Sanchez", "manuel@rigoleunpeu.fr", new
                        Visiteur()));
190                 listePersonne.add(new Personne("Justine PetiteGoutte", "justine@pttegoutte.fr"));
191                 listePersonne.add(new Personne("Ervin PetiteGoutte", "ervin@pttegoutte.fr"));
192
193                 listePersonne.add(new Personne("Raoul alacavaplus", "raaaah@tatouille.fr", new
                        Employe()));
194                 listePersonne.add(new Personne("Jose ventrerlaligo", "Jose@plucher.fr", new
                        Employe()));
195
196                 listePersonne.add(new Personne("Jessica Scroutepourmidi",
                        "JessicaScroutepourmidi@gfaim.fr", new Adherent()));
197         }
198
199         public void initAlerte()
200         {
```

```
201                 for(Personne p : listePersonne)
202                 {
203                         Alerte al = new Alerte();
204                         p.ajoutOberserver(al);
205                         listeAlerte.add(al);
206                         al.setSeuilPanier(100);
207                         ArrayList<Produit> combinaisonProduit = new ArrayList<Produit>();
208                         combinaisonProduit.add(listeProduit.get(0));
209                         combinaisonProduit.add(listeProduit.get(1));
210                         al.setCombinaisonProduit(combinaisonProduit);
211                 }
212         }
213 
214         public void testerEnvironnement()
215         {
216                 Personne p = listePersonne.get(5);
217                 p.achat(listeProduit.get(4));
218                 System.out.println("-----");
219                 double paiement = p.paiement();
220                 System.out.println("Le Client " + p.getId() + " a paye " + paiement + " euros.");
221         }
222 
223         public static void main(String[] args)
224         {
225                 Contexte environnement = Contexte.getInstance();
226                 environnement.initMarketing();
227                 environnement.initPersonne();
228                 environnement.initAlerte();
229                 //System.out.println(environnement.toString() + "\n");
230 
231                 environnement.testerEnvironnement();
232                 // System.out.println(environnement.toString());
233         }
234 }
```

```
1  package laFac;
2 
3  import java.util.ArrayList;
4  import java.util.Observable;
5  import java.util.Observer;
6 
7  public class Alerte implements Observer
8  {
9         private int seuilPanier;
10        private ArrayList<Produit> combinaisonProduit;
11 
12        public Alerte()
13        {
14                super();
15                seuilPanier = 0;
16                combinaisonProduit = new ArrayList<Produit>();
17        }
18 
```

```
19          public void setCombinaisonProduit(ArrayList<Produit> combinaisonProduit)
20          {
21                  this.combinaisonProduit.addAll(combinaisonProduit);
22          }
23
24          public void setSeuilPanier(int seuilPanier)
25          {
26                  this.seuilPanier = seuilPanier;
27          }
28
29          public void update(Observable arg0, Object arg1)
30          {
31                  Panier pan = (Panier) arg0;
32                  System.out.println("Dans l'alerte !");
33                  if(pan.getListProduits().containsAll(combinaisonProduit))
34                          System.out.println("-----\nALERTE\nLe panier de " +
                                pan.getNomProprietaire() + " contient la combinaison recherchee.");
35                  if(pan.coutPanier() > seuilPanier)
36                          System.out.println("-----\nALERTE\nLe panier de " +
                                pan.getNomProprietaire() + " a depasse le seuil des " + seuilPanier +
                                " euros.\n-----");
37          }
38
39  }
```

```
1   package laFac;
2
3   import java.util.ArrayList;
4   import java.util.Observable;
5
6   public class Panier extends Observable
7   {
8           private String date;
9           private ArrayList<Produit> listProduits;
10          private Personne proprietaire;
11
12          public Panier(Personne proprietaire)
13          {
14                  // La date est fixee a aujourd'hui pour plus de simplicite
15                  date = "Aujourd'hui";
16                  listProduits = new ArrayList<Produit>();
17                  this.proprietaire = proprietaire;
18          }
19
20          public boolean ajoutProduit(Produit p)
21          {
22                  if (p != null)
23                  {
24                          listProduits.add(p);
25                          this.notifyObservers();
26                          return true;
27                  } else
28                  {
```

```
29                  System.out.println("Impossible d'ajouter au panier, produit inexistant");
30                  return false;
31          }
32  }
33
34  public ArrayList<Produit> getListProduits()
35  {
36          return listProduits;
37  }
38
39  public String getNomProprietaire()
40  {
41          return proprietaire.getNom();
42  }
43
44  public double facturation()
45  {
46          double coutAchat = 0;
47          double coutReductionGenerale = 0;
48          double coutReductionStatut = 0;
49          int ptsFidelite = 0;
50
51          for (Produit prod : listProduits)
52          {
53                  coutAchat += prod.getPrix();
54          }
55
56          for (Offre ofr : Contexte.getListeOffreGenerale())
57          {
58                  if (!ofr.getProduits().isEmpty() &&
                        listProduits.containsAll(ofr.getProduits()))
59                     coutReductionGenerale += ofr.getReduction();
60          }
61
62          for (Offre ofr : proprietaire.getListeOffreStatut())
63          {
64                  if (!ofr.getProduits().isEmpty() &&
                        listProduits.containsAll(ofr.getProduits()))
65                  {
66                          coutReductionStatut += ofr.getReduction();
67                          ptsFidelite += ofr.getPtsFidelite();
68                  }
69          }
70
71          if (ptsFidelite != 0)
72          {
73                  proprietaire.miseAJourPtsFidelite(ptsFidelite);
74                  coutReductionStatut += proprietaire.getReductionFixeFidelite();
75          }
76
77          // On presente l'addition
78          System.out.println(lAddition(coutAchat, coutReductionGenerale,
                coutReductionStatut, ptsFidelite));
79
```

```
80              // Avant de retourner ce que doit payer le Client on vide le panier
81              videPanier();
82              return coutAchat - coutReductionGenerale - coutReductionStatut;
83      }
84
85      /**Renvoie le cout du panier sans prendre en compte les reductions eventullement
            possibles.*/
86      public double coutPanier()
87      {
88              double coutAchat = 0;
89
90              for (Produit prod : listProduits)
91              {
92                      coutAchat += prod.getPrix();
93              }
94
95              return coutAchat;
96      }
97
98      public void videPanier()
99      {
100             listProduits.removeAll(listProduits);
101     }
102
103     public String toString()
104     {
105             String s = new String();
106             s = "\n Date de creation : " + date;
107
108             if (listProduits.isEmpty())
109                     s += "\n Panier Vide.";
110             else
111             {
112
113                     s += "\n Produits contenus :";
114
115                     for (Produit prod : listProduits)
116                     {
117                             s += "\n - " + prod.toStringPanier();
118                     }
119             }
120
121             return s;
122     }
123
124     public String lAddition(double coutAchat, double coutReductionGenerale, double
            coutReductionStatut, int ptsFidelite)
125     {
126             String s = new String();
127             s += "VOICI L'ADDITION\n--------------------";
128
129             for (Produit prod : listProduits)
130             {
131                     s += "\n - " + prod.toStringPanier();
```

```
132                 }
133
134                 s += "\n--------------------\nTotal des Achats\t" + coutAchat + " euros";
135                 if (coutReductionGenerale != 0)
136                         s += "\nTotal des Promotions\t" + coutReductionGenerale + " euros";
137                 if (coutReductionStatut != 0)
138                         s += "\nTotal des reductions grace a votre Statut\t" + coutReductionStatut
                            + " euros";
139                 if (ptsFidelite != 0)
140                         s += "\nTotal des points de fidelite gagne\t" + ptsFidelite;
141                 s += "\n--------------------\nTotal\t" + (coutAchat - coutReductionGenerale -
                        coutReductionStatut) + " euros";
142
143                 return s;
144         }
145
146 }
```

Personnes

```
1  package laFac;
2
3  import java.util.ArrayList;
4
5  public class Personne
6  {
7          private String nom;
8          private String adresse;
9          private int id;
10         private Statut stat;
11         private Panier pan;
12         private static int currentID = 0;
13
14         /** La personne est creee avec un statut de visiteur, elle doit se connecter pour changer
               de Statut */
15         public Personne(String nom, String adresse)
16         {
17                 this.nom = nom;
18                 this.adresse = adresse;
19                 this.id = getNewID();
20                 stat = new Visiteur();
21                 pan = new Panier(this);
22         }
23
24         public Personne(String nom, String adresse, Statut statut)
25         {
26                 this(nom, adresse);
27                 stat = statut;
28         }
29
30         public void ajoutOberserver(Alerte al)
```

```
31          {
32                  pan.addObserver(al);
33          }
34
35          public void connexion(Statut statut)
36          {
37                  this.stat = statut;
38          }
39
40          public void deconnexion()
41          {
42                  this.stat = new Visiteur();
43          }
44
45          public String getNom()
46          {
47                  return nom;
48          }
49
50          public int getId()
51          {
52                  return id;
53          }
54
55          public static int getNewID()
56          {
57                  return currentID++;
58          }
59
60          public ArrayList<Offre> getListeOffreStatut()
61          {
62                  return stat.getListeOffreStatut();
63          }
64
65          public void miseAJourPtsFidelite(int ptsFidelite)
66          {
67                  stat.miseAJourPtsFidelite(ptsFidelite);
68          }
69
70          public double getReductionFixeFidelite()
71          {
72                  return stat.getReductionFixeFidelite();
73          }
74
75          public boolean achat(Produit prod)
76          {
77                  System.out.println(nom + " ajoute a son panier : \n - " + prod.toStringPanier());
78                  return pan.ajoutProduit(prod);
79          }
80
81          public double paiement()
82          {
83                  System.out.println(nom + " passe a la caisse !");
84                  return pan.facturation();
```

```
85          }
86
87          public String toString()
88          {
89                  return "PERSONNE : \nNom : " + nom + "\nE-mail : " + adresse + "\nId : " + id +
                        "\nStatut : " + stat.toString() + "\nPanier : " + pan.toString();
90          }
91
92  }
```

```
1   package laFac;
2
3   public enum NomStatut
4   {
5           Employe, Adherent, Visiteur;
6   }
```

```
1   package laFac;
2
3   import java.util.ArrayList;
4
5   public abstract class Statut
6   {
7           private NomStatut nomStat;
8
9           protected Statut()
10          {
11                  nomStat = NomStatut.Visiteur;
12          }
13
14          protected Statut(NomStatut statut)
15          {
16                  nomStat = statut;
17          }
18
19          public abstract ArrayList<Offre> getListeOffreStatut();
20
21          public abstract void miseAJourPtsFidelite(int ptsFidelite);
22
23          public abstract double getReductionFixeFidelite();
24
25          public String toString()
26          {
27                  return "" + nomStat;
28          }
29
30  }
```

```
1   package laFac;
2
3   import java.util.ArrayList;
```

```
4
5  public class Adherent extends Statut
6  {
7          private int nbPtsFidelite;
8          private static int limitePtsFidelitePourReduction = 200;
9          private static double reductionFixeFidelite = 10;
10
11         public Adherent()
12         {
13                 super(NomStatut.Adherent);
14                 nbPtsFidelite = 0;
15         }
16
17         public ArrayList<Offre> getListeOffreStatut()
18         {
19                 return Contexte.getListeOffreAdherent();
20         }
21
22         public double getReductionFixeFidelite()
23         {
24                 int nbLimite = 0;
25
26                 //Si l'adherent a assez de points pour avoir une reduction
27                 while(nbPtsFidelite >= limitePtsFidelitePourReduction)
28                 {
29                         nbLimite++;
30                         nbPtsFidelite -= limitePtsFidelitePourReduction;
31                 }
32
33                 return nbLimite * reductionFixeFidelite;
34         }
35
36         public void miseAJourPtsFidelite(int ptsFidelite)
37         {
38                 nbPtsFidelite += ptsFidelite;
39         }
40
41         public String toString()
42         {
43                 return super.toString() + "\nNombre de Points de Fidelite : " + nbPtsFidelite;
44         }
45 }
```

```
1  package laFac;
2
3  import java.util.ArrayList;
4
5  public class Employe extends Statut
6  {
7
8          public Employe()
9          {
10                 super(NomStatut.Employe);
```

```
        }

        public ArrayList<Offre> getListeOffreStatut()
        {
                return Contexte.getListeOffreEmploye();
        }

        public double getReductionFixeFidelite()
        {
                return 0;
        }

        public void miseAJourPtsFidelite(int ptsFidelite)
        {
        }

}
```

```
package laFac;

import java.util.ArrayList;

public class Visiteur extends Statut
{
        public Visiteur()
        {
                super(NomStatut.Visiteur);
        }

        public ArrayList<Offre> getListeOffreStatut()
        {
                return new ArrayList<Offre>();
        }

        public double getReductionFixeFidelite()
        {
                return 0;
        }

        public void miseAJourPtsFidelite(int ptsFidelite)
        {
        }

}
```

Produits

```
package laFac;

public abstract class Produit
```

```
4  {
5          //Le nom est suppose unique et sert donc de cle primaire
6          protected String nom;
7          protected double prix;
8          protected int stock;
9          protected boolean estElligibleAOffreProduit;
10         protected NomCategorie categorie;
11
12         public Produit(String nom, double prix, int stock, boolean estElligibleAOffreProduit)
13         {
14                 this.nom = nom;
15                 this.prix = prix;
16                 this.stock = stock;
17                 this.estElligibleAOffreProduit = estElligibleAOffreProduit;
18         }
19
20         public void setEstElligibleAOffreProduit(boolean estElligibleAOffreProduit)
21         {
22                 this.estElligibleAOffreProduit = estElligibleAOffreProduit;
23         }
24
25         public boolean getEstElligibleAOffreProduit()
26         {
27                 return estElligibleAOffreProduit;
28         }
29
30         public String getNom()
31         {
32                 return nom;
33         }
34
35         public double getPrix()
36         {
37                 return prix;
38         }
39
40         public void setStock(int stock)
41         {
42                 this.stock = stock;
43         }
44
45         public String toStringPanier()
46         {
47                 return nom + "\t" + prix + " euros";
48         }
49
50         public String toString()
51         {
52                 String s = new String();
53                 s = "Nom Produit : " + nom + "\nPrix : " + prix + " euros\nCategorie : " +
                       categorie;
54                 if (stock == 0)
55                         s = s + "\nRupture de stock.";
56                 else
```

```
57                      s = s + "\nEn stock : " + stock;
58
59              return s;
60      }
61
62      public boolean equals(Object obj)
63      {
64              if (obj == null)
65                      return false;
66
67              try
68              {
69                      Produit prod = (Produit) obj;
70                      return this.nom.equals(prod.nom);
71              } catch (ClassCastException e)
72              {
73                      return false;
74              }
75      }
76
77  }
```

```
1   package laFac;
2
3   public enum NomCategorie
4   {
5           Culture, Hightech;
6   }
```

```
1   package laFac;
2
3   public abstract class Culture extends Produit
4   {
5           public Culture(String nom, double prix, int stock, boolean estElligibleAOffreProduit)
6           {
7                   super(nom, prix, stock, estElligibleAOffreProduit);
8                   this.categorie = NomCategorie.Culture;
9           }
10
11  }
```

```
1   package laFac;
2
3   public abstract class Hightech extends Produit
4   {
5           protected int tempsGarantie;
6
7           public Hightech(String nom, double prix, int stock, boolean estElligibleAOffreProduit ,
                int tempsGarantie)
8           {
9                   super(nom, prix, stock, estElligibleAOffreProduit);
```

```
                this.tempsGarantie = tempsGarantie;
                this.categorie = NomCategorie.Hightech;
        }

        public String toString()
        {
                return super.toString() + "\nGarantie : " + tempsGarantie + " ans";
        }

}
```

```
package laFac;

public class Livres extends Culture
{
        private String titre;
        private String auteur;

        public Livres(double prix, int stock, String titre, String auteur)
        {
                super(titre, prix, stock, false);
                this.titre = titre;
                this.auteur = auteur;
        }

        public String toString()
        {
                return "LIVRE : \nTitre : " + titre + "\nAuteur : " + auteur + "\n" +
                    super.toString();
        }

}
```

```
package laFac;

public class Spectacles extends Culture
{
        private String titre;
        private String date;

        public Spectacles(double prix, int stock, String titre, String date)
        {
                super(titre, prix, stock, true);
                this.titre = titre;
                this.date = date;
        }

        public String toString()
        {
                return "SPECTACLE : \nTitre : " + titre + "\nDate de representation : " + date +
                    "\n" + super.toString();
        }
```

```
}
```

```
package laFac;

public class Tablette extends Hightech
{
	private String marque;

	public Tablette(double prix, int stock, int tempsGarantie, String marque)
	{
		super("Tablette " + marque, prix, stock, true, tempsGarantie);
		this.marque = marque;
	}

	public String toString()
	{
		return "TABLETTE : \nMarque : " + marque + "\n" + super.toString();
	}

}
```

Offres

```
package laFac;

import java.util.ArrayList;

public abstract class Offre
{
	protected ArrayList<Produit> produits;
	protected double reduction;

	public Offre(double reduction)
	{
		produits = new ArrayList<Produit>();
		this.reduction = reduction;
	}

	public abstract double getReduction();

	public int getPtsFidelite()
	{
		return 0;
	}

	public ArrayList<Produit> getProduits()
	{
		ArrayList<Produit> l = new ArrayList<Produit>();
		l.addAll(produits);
		return l;
```

```
28          }
29
30          /**Renvoie true si l'offre a encore raison d'exister, false sinon*/
31          public abstract boolean miseAJourDate();
32
33  }
```

```
1   package laFac;
2
3   public class OffreEmploye extends Offre
4   {
5
6           /** Reduction en pourcentage. */
7           public OffreEmploye(Produit prod, double reduction)
8           {
9                   super(reduction);
10                  if (prod != null)
11                          produits.add(prod);
12          }
13
14          public double getReduction()
15          {
16                  if (!produits.isEmpty())
17                          return produits.get(0).getPrix() * reduction;
18                  else
19                          return 0;
20          }
21
22          public boolean miseAJourDate()
23          {
24                  return true;
25          }
26
27          public String toString()
28          {
29                  if (!produits.isEmpty())
30                          return "OFFRE EMPLOYE : \nProduit : " + produits.get(0).getNom() +
                                "\nReduction : " + (reduction * 100) + " %";
31                  else
32                          return "OFFRE EMPLOYE NON APPLICABLE";
33          }
34
35  }
```

```
1   package laFac;
2
3   public class OffreFidelite extends Offre
4   {
5           public int ptsFidelite;
6
7           public OffreFidelite(Produit prod, int ptsFidelite)
8           {
```

```
                super(0);
                this.ptsFidelite = ptsFidelite;
                if (prod != null)
                        produits.add(prod);
        }

        public double getReduction()
        {
                return reduction;
        }

        public int getPtsFidelite()
        {
                return ptsFidelite;
        }

        public boolean miseAJourDate()
        {
                return true;
        }

        public String toString()
        {
                if (!produits.isEmpty())
                        return "OFFRE FIDELITE : \nProduit : " + produits.get(0).getNom() +
                            "\nPoints de Fidelite a gagner : " + reduction;
                else
                        return "OFFRE FIDELITE NON APPLICABLE";
        }

}
```

```
package laFac;

import java.util.ArrayList;

public class OffreFlash extends Offre
{
        // La date limite est defini en nombre de mise a jour du contexte.
        private int dateExpiration;

        /** Reduction en pourcentage. */
        public OffreFlash(Produit prod, double reduction, int dateExpiration)
        {
                super(reduction);
                this.dateExpiration = dateExpiration;
                if (prod != null)
                        produits.add(prod);
        }

        /** Reduction en pourcentage. */
        public OffreFlash(ArrayList<Produit> prods, double reduction, int dateExpiration)
        {
```

```
22                 super(reduction);
23                 this.dateExpiration = dateExpiration;
24                 if (prods != null)
25                         produits.addAll(prods);
26         }
27
28         public double getReduction()
29         {
30                 double prix = 0;
31
32                 for (Produit prod : produits)
33                 {
34                         prix += prod.getPrix() * reduction;
35                 }
36
37                 return prix;
38         }
39
40         public void addProduit(Produit prod)
41         {
42                 if(prod != null)
43                         produits.add(prod);
44         }
45
46         public boolean miseAJourDate()
47         {
48                 dateExpiration--;
49                 return (dateExpiration > 0);
50         }
51
52         public String toString()
53         {
54                 String s = new String();
55
56                 if (!produits.isEmpty())
57                 {
58                         s += "OFFRE FLASH : \nProduits : \n";
59                         for (Produit prod : produits)
60                         {
61                                 s += " - " + prod.getNom() + "\n";
62                         }
63
64                         s += "Reduction : " + (reduction * 100) + " %\nDate d'Expiration : " +
                                (dateExpiration - Contexte.getDateDuJour()) + " jours restants.";
65
66                         return s;
67                 } else
68                         return "OFFRE FLASH NON APPLICABLE";
69         }
70
71 }
```

```
1  package laFac;
```

```
2
3  public class OffreProduit extends Offre
4  {
5          // La date limite est defini en nombre de mise a jour du contexte.
6          private int dateExpiration;
7
8          /** Reduction en pourcentage. Si le produit n'est pas elligible on ne l'ajoute pas a la
               liste de produit. */
9          public OffreProduit(Produit prod, double reduction, int dateExpiration)
10         {
11                 super(reduction);
12                 this.dateExpiration = dateExpiration;
13
14                 if (prod.getEstElligibleAOffreProduit())
15                 {
16                         produits.add(prod);
17                 }
18         }
19
20         public double getReduction()
21         {
22                 if(!produits.isEmpty())
23                         return produits.get(0).getPrix() * reduction;
24                 else
25                         return 0;
26         }
27
28         public boolean miseAJourDate()
29         {
30                 dateExpiration--;
31                 return (dateExpiration > 0);
32         }
33
34         public String toString()
35         {
36                 if(!produits.isEmpty())
37                         return "OFFRE PRODUIT : \nProduit : " + produits.get(0).getNom() +
                               "\nReduction : " + (reduction * 100) + " %\nDate d'Expiration : " +
                               (dateExpiration - Contexte.getDateDuJour()) + " jours restants.";
38                 else
39                         return "OFFRE PRODUIT NON APPLICABLE";
40         }
41
42 }
```