



## CONCEPTION OBJET

PROJET DE PROGRAMMATION ORIENTÉ OBJET

---

**LaFac.com**

---

*Auteur :*

Jeremie BOSOM  
Erwan CHAUSSY  
Léo DELLOUVE

*Professeur :*

M. Frederic VOISIN

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 devons 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 posséderait 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 devons 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 connaître 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.

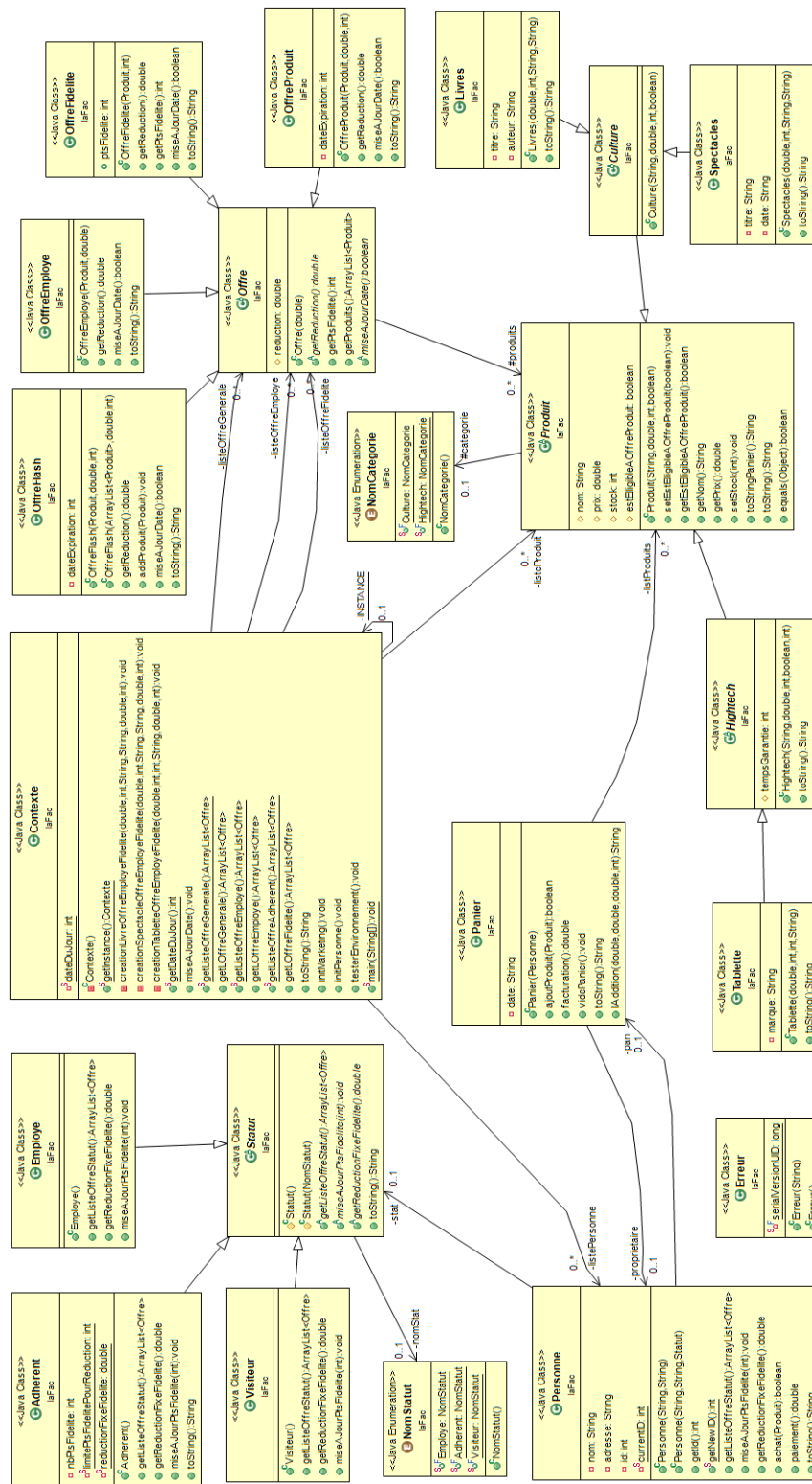


FIGURE 1 – Diagramme de classe

## Diagrammes de séquences

Nous avons décidé de détailler les interactions entre les classes lors la création du contexte et lors de la facturation du panier.

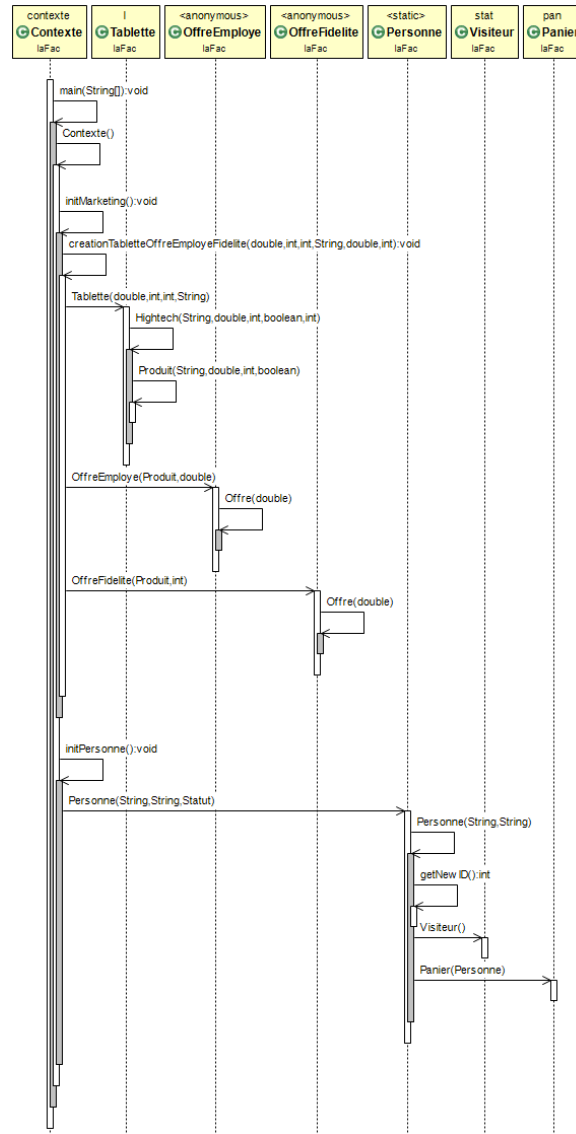


FIGURE 2 – Diagramme de séquence de la création du contexte

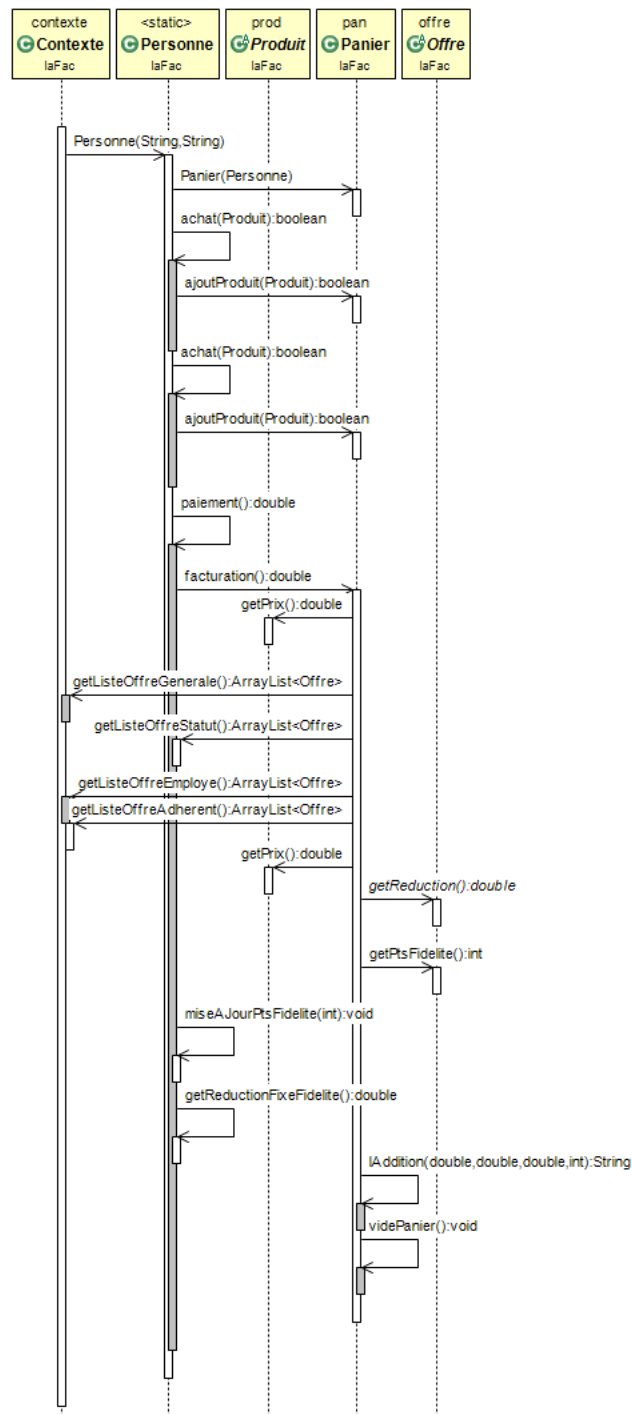


FIGURE 3 – Diagramme de séquence de la facturation

## 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

Contexte.java

```

1 package laFac;

import java.util.ArrayList;

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

    private Contexte()
    {
        listePersonne = new ArrayList<Personne>();
        listeProduit = new ArrayList<Produit>();
21 listeOffreGenerale = new ArrayList<Offre>();
        listeOffreEmploye = new ArrayList<Offre>();
        listeOffreFidelite = new ArrayList<Offre>();
        listeAlerte = new ArrayList<Alerte>();
26 }

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

    private void creationLivreOffreEmployeFidelite(double prix, int stock, String titre,
        String auteur, double reductionEmploye, int ptsFidelite)
    {
36         Livres l = new Livres(prix, stock, titre, auteur);
        listeProduit.add(l);
        listeOffreEmploye.add(new OffreEmploye(l, reductionEmploye));
        listeOffreFidelite.add(new OffreFidelite(l, ptsFidelite));
    }
41

    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);

```



```

46         listeOffreEmploye.add(new OffreEmploye(1, reductionEmploye));
        listeOffreFidelite.add(new OffreFidelite(1, ptsFidelite));
    }

    private void creationTabletteOffreEmployeFidelite(double prix, int stock, int
        tmpsGarantie, String marque, double reductionEmploye, int ptsFidelite)
51    {
        Tablette l = new Tablette(prix, stock, tmpsGarantie, marque);
        listeProduit.add(l);
        listeOffreEmploye.add(new OffreEmploye(1, reductionEmploye));
        listeOffreFidelite.add(new OffreFidelite(1, ptsFidelite));
56    }

    public static int getDateDuJour()
    {
        return dateDuJour;
61    }

    /** Incrémente la date de 1 et supprime toutes les Offres révolues. */
    public void miseAJourDate()
    {
66        dateDuJour++;

        for(Offre ofr : listeOffreGenerale)
        {
            if(!ofr.miseAJourDate())
71                listeOffreGenerale.remove(ofr);
        }

        for(Offre ofr : listeOffreEmploye)
        {
76            if(!ofr.miseAJourDate())
                listeOffreGenerale.remove(ofr);
        }

        for(Offre ofr : listeOffreFidelite)
81        {
            if(!ofr.miseAJourDate())
                listeOffreGenerale.remove(ofr);
        }
    }

86    public static ArrayList<Offre> getListeOffreGenerale()
    {
        final ArrayList<Offre> listTmp = new ArrayList<Offre>();
        listTmp.addAll(Contexte.getInstance().getLOffreGenerale());
91        return listTmp;
    }

    public ArrayList<Offre> getLOffreGenerale()
    {
96        return listeOffreGenerale;
    }

```

```

public static ArrayList<Offre> getListeOffreEmploye()
{
101     final ArrayList<Offre> listTmp = new ArrayList<Offre>();
    listTmp.addAll(Contexte.getInstance().getLOffreEmploye());
    return listTmp;
}

106 public ArrayList<Offre> getLOffreEmploye()
{
    return listeOffreEmploye;
}

111 public static ArrayList<Offre> getListeOffreAdherent()
{
    final ArrayList<Offre> listTmp = new ArrayList<Offre>();
    listTmp.addAll(Contexte.getInstance().getLOffreFidelite());
    return listTmp;
116 }

public ArrayList<Offre> getLOffreFidelite()
{
    return listeOffreFidelite;
121 }

public String toString()
{
    String ret = new String();

126     ret = "ENVIRONNEMENT\n-----\nListe des PRODUITS : \n-----";
    for (Produit prod : listeProduit)
    {
        ret += "\n" + prod.toString() + "\n-----";
131     }

    ret += "\n-----\nListe des OFFRES : \n-----";
    for (Offre ofr : listeOffreGenerale)
    {
        ret += "\n" + ofr.toString() + "\n-----";
136     }

    for (Offre ofr : listeOffreEmploye)
    {
        ret += "\n" + ofr.toString() + "\n-----";
141     }

    for (Offre ofr : listeOffreFidelite)
    {
        ret += "\n" + ofr.toString() + "\n-----";
146     }

    ret += "\n-----\nListe des PERSONNES : \n-----";
    for (Personne pers : listePersonne)
151     {
        ret += "\n" + pers.toString() + "\n-----";

```

```

    }

    ret += "\n-----";

156     return ret;
}

/** L'initialisation de la partie marketing consiste à initialiser les produits et offres.
 */
161 public void initMarketing()
{
    /* Initialisation des produits */
    creationLivreOffreEmployeFidelite(25, 10, "Toto à l'école des Charcutiers", "Toto",
        0.5, 10);
    creationLivreOffreEmployeFidelite(15, 2000, "Toto et Martine", "Martine", 0.5, 10);
166    creationSpectacleOffreEmployeFidelite(50, 50, "Toto est Figaro", "12 décembre
        2012", 0.4, 25);
    creationSpectacleOffreEmployeFidelite(75, 250, "Toto attend Godot", "36 avril
        2013", 0.4, 25);
    creationTabletteOffreEmployeFidelite(250, 5000, 2, "Comon", 0.25, 50);
    creationTabletteOffreEmployeFidelite(400, 5000, 3, "Gogo Gadgeto", 0.25, 200);

171    /* Initialisation des offres */
    /* Offre Produits */
    listeOffreGenerale.add(new OffreProduit(listeProduit.get(0), 0.25, 2));
    listeOffreGenerale.add(new OffreProduit(listeProduit.get(2), 0.25, 2));
    listeOffreGenerale.add(new OffreProduit(listeProduit.get(4), 0.25, 3));
176    /* Offre Flash */
    OffreFlash f = new OffreFlash(listeProduit.get(0), 0.25, 2);
    listeOffreGenerale.add(f);
    f.addProduit(listeProduit.get(1));
    ArrayList<Produit> tmp = new ArrayList<Produit>();
181    tmp.addAll(listeProduit);
    listeOffreGenerale.add(new OffreFlash(tmp, 0.25, 2));
    listeOffreGenerale.add(new OffreFlash(listeProduit.get(4), 0.25, 3));
}

186 /** L'initialisation des personnes faisant vivre le site */
public void initPersonne()
{
    listePersonne.add(new Personne("Manuel Sanchez", "manuel@rigoleunpeu.fr", new
        Visiteur()));
    listePersonne.add(new Personne("Justine PetiteGoutte", "justine@pttegoutte.fr"));
191    listePersonne.add(new Personne("Ervin PetiteGoutte", "ervin@pttegoutte.fr"));

    listePersonne.add(new Personne("Raoul alacavaplus", "raaaah@tatouille.fr", new
        Employe()));
    listePersonne.add(new Personne("Jose ventrerlraligo", "Jose@plucher.fr", new
        Employe()));

196    listePersonne.add(new Personne("Jessica Scroutepourmidi",
        "JessicaScroutepourmidi@gfaim.fr", new Adherent()));
}

```

```

public void initAlerte()
{
    for(Personne p : listePersonne)
    {
        Alerte al = new Alerte();
        p.ajoutOberserver(al);
        listeAlerte.add(al);
        al.setSeuilPanier(100);
        ArrayList<Produit> combinaisonProduit = new ArrayList<Produit>();
        combinaisonProduit.add(listeProduit.get(0));
        combinaisonProduit.add(listeProduit.get(1));
        al.setCombinaisonProduit(combinaisonProduit);
    }
}

public void testerEnvironnement()
{
    Personne p = listePersonne.get(5);
    p.achat(listeProduit.get(4));
    System.out.println("-----");
    double paiement = p.paiement();
    System.out.println("Le Client " + p.getId() + " a payé " + paiement + " €.");
}

public static void main(String[] args)
{
    Contexte environnement = Contexte.getInstance();
    environnement.initMarketing();
    environnement.initPersonne();
    environnement.initAlerte();
    //System.out.println(environnement.toString() + "\n");

    environnement.testeEnvironnement();
    // System.out.println(environnement.toString());
}
}

```

---

#### Alerte.java

---

```

1 package laFac;

import java.util.ArrayList;
import java.util.Observable;
import java.util.Observer;

6 public class Alerte implements Observer
{
    private int seuilPanier;
    private ArrayList<Produit> combinaisonProduit;

11 public Alerte()
{
    super();
}

```

```

    seuilPanier = 0;
    combinaisonProduit = new ArrayList<Produit>();
16     }

    public void setCombinaisonProduit(ArrayList<Produit> combinaisonProduit)
    {
21         this.combinaisonProduit.addAll(combinaisonProduit);
    }

    public void setSeuilPanier(int seuilPanier)
    {
26         this.seuilPanier = seuilPanier;
    }

    public void update(Observable arg0, Object arg1)
    {
31         Panier pan = (Panier) arg0;
        System.out.println("Dans l'alerte !");
        if(pan.getListProduits().containsAll(combinaisonProduit))
            System.out.println("-----\nALERTE\nLe panier de " +
                pan.getNomProprietaire() + " contient la combinaison recherchée.");
        if(pan.coutPanier() > seuilPanier)
36         System.out.println("-----\nALERTE\nLe panier de " +
            pan.getNomProprietaire() + " a dépassé le seuil des " + seuilPanier +
            " €.\n-----");
    }
}

```

---

### Panier.java

---

```

1  package laFac;

    import java.util.ArrayList;
    import java.util.Observable;

6  public class Panier extends Observable
    {
        private String date;
        private ArrayList<Produit> listProduits;
        private Personne proprietaire;

11     public Panier(Personne proprietaire)
        {
            // La date est fixée à aujourd'hui pour plus de simplicité
            date = "Aujourd'hui";
16         listProduits = new ArrayList<Produit>();
            this.proprietaire = proprietaire;
        }

        public boolean ajoutProduit(Produit p)
21     {
        if (p != null)

```

```

    {
        listProduits.add(p);
        this.notifyObservers();
        return true;
    } else
    {
        System.out.println("Impossible d'ajouter au panier, produit inexistant");
        return false;
    }
}

public ArrayList<Produit> getListProduits()
{
    return listProduits;
}

public String getNomProprietaire()
{
    return proprietaire.getNom();
}

public double facturation()
{
    double coutAchat = 0;
    double coutReductionGenerale = 0;
    double coutReductionStatut = 0;
    int ptsFidelite = 0;

    for (Produit prod : listProduits)
    {
        coutAchat += prod.getPrix();
    }

    for (Offre ofr : Contexte.getListeOffreGenerale())
    {
        if (!ofr.getProduits().isEmpty() &&
            listProduits.containsAll(ofr.getProduits()))
            coutReductionGenerale += ofr.getReduction();
    }

    for (Offre ofr : proprietaire.getListeOffreStatut())
    {
        if (!ofr.getProduits().isEmpty() &&
            listProduits.containsAll(ofr.getProduits()))
        {
            coutReductionStatut += ofr.getReduction();
            ptsFidelite += ofr.getPtsFidelite();
        }
    }

    if (ptsFidelite != 0)
    {
        proprietaire.miseAJourPtsFidelite(ptsFidelite);
        coutReductionStatut += proprietaire.getReductionFixeFidelite();
    }
}

```

```

    }

76
    // On présente l'addition
    System.out.println(lAddition(coutAchat, coutReductionGenerale,
        coutReductionStatut, ptsFidelite));

    // Avant de retourner ce que doit payer le Client on vide le panier
81
    videPanier();
    return coutAchat - coutReductionGenerale - coutReductionStatut;
}

/**Renvoie le cout du panier sans prendre en compte les réductions eventuellement
    possibles.*/
86
public double coutPanier()
{
    double coutAchat = 0;

    for (Produit prod : listProduits)
91
    {
        coutAchat += prod.getPrix();
    }

    return coutAchat;
96
}

public void videPanier()
{
    listProduits.removeAll(listProduits);
101
}

public String toString()
{
    String s = new String();
106
    s = "\n Date de création : " + date;

    if (listProduits.isEmpty())
        s += "\n Panier Vide.";
    else
111
    {
        s += "\n Produits contenus :";

        for (Produit prod : listProduits)
116
        {
            s += "\n - " + prod.toStringPanier();
        }

        return s;
121
    }

    public String lAddition(double coutAchat, double coutReductionGenerale, double
        coutReductionStatut, int ptsFidelite)
    {

```

```

126     String s = new String();
        s += "VOICI L'ADDITION\n-----";

        for (Produit prod : listProduits)
        {
131             s += "\n - " + prod.toStringPanier();
        }

        s += "\n-----\nTotal des Achats\t" + coutAchat + " €";
        if (coutReductionGenerale != 0)
136             s += "\nTotal des Promotions\t" + coutReductionGenerale + " €";
        if (coutReductionStatut != 0)
            s += "\nTotal des réductions grâce à votre Statut\t" + coutReductionStatut
                + " €";
        if (ptsFidelite != 0)
            s += "\nTotal des points de fidélité gagné\t" + ptsFidelite;
141     s += "\n-----\nTotal\t" + (coutAchat - coutReductionGenerale -
            coutReductionStatut) + " €";

        return s;
    }
146 }

```

---

## Personnes

Personne.java

---

```

package laFac;

import java.util.ArrayList;
4
public class Personne
{
    private String nom;
    private String adresse;
9    private int id;
    private Statut stat;
    private Panier pan;
    private static int currentID = 0;

14    /** La personne est créée avec un statut de visiteur, elle doit se connecter pour changer
        de Statut */
    public Personne(String nom, String adresse)
    {
        this.nom = nom;
        this.adresse = adresse;
19        this.id = getNewID();
        stat = new Visiteur();
        pan = new Panier(this);
    }
}

```



```

24     public Personne(String nom, String adresse, Statut statut)
    {
        this(nom, adresse);
        stat = statut;
    }

29     public void ajoutObserver(Alerte al)
    {
        pan.addObserver(al);
    }

34     public void connexion(Statut statut)
    {
        this.stat = statut;
    }

39     public void deconnexion()
    {
        this.stat = new Visiteur();
    }

44     public String getNom()
    {
        return nom;
    }

49     public int getId()
    {
        return id;
    }

54     public static int getNewID()
    {
        return currentID++;
    }

59     public ArrayList<Offre> getListeOffreStatut()
    {
        return stat.getListeOffreStatut();
    }

64     public void miseAJourPtsFidelite(int ptsFidelite)
    {
        stat.miseAJourPtsFidelite(ptsFidelite);
    }

69     public double getReductionFixeFidelite()
    {
        return stat.getReductionFixeFidelite();
    }

74     public boolean achat(Produit prod)
    {
        System.out.println(nom + " ajoute à son panier : \n - " + prod.toStringPanier());
    }

```

```

    return pan.ajoutProduit(prod);
79 }

    public double paiement()
    {
        System.out.println(nom + " passe à la caisse !");
84     return pan.facturation();
    }

    public String toString()
    {
89     return "PERSONNE : \nNom : " + nom + "\nE-mail : " + adresse + "\nId : " + id +
        "\nStatut : " + stat.toString() + "\nPanier : " + pan.toString();
    }
}

```

---

#### NomStatut.java

---

```

package laFac;

3 public enum NomStatut
{
    Employe, Adherent, Visiteur;
}

```

---

#### Statut.java

---

```

package laFac;

import java.util.ArrayList;
4 public abstract class Statut
{
    private NomStatut nomStat;

9    protected Statut()
    {
        nomStat = NomStatut.Visiteur;
    }

14    protected Statut(NomStatut statut)
    {
        nomStat = statut;
    }

19    public abstract ArrayList<Offre> getListeOffreStatut();

    public abstract void miseAJourPtsFidelite(int ptsFidelite);

    public abstract double getReductionFixeFidelite();

24    public String toString()

```

```

    {
        return "" + nomStat;
    }
}

```

---

### Adherent.java

---

```

package laFac;

import java.util.ArrayList;

5 public class Adherent extends Statut
{
    private int nbPtsFidelite;
    private static int limitePtsFidelitePourReduction = 200;
    private static double reductionFixeFidelite = 10;

10 public Adherent()
{
    super(NomStatut.Adherent);
    nbPtsFidelite = 0;

15 }

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

20 }

    public double getReductionFixeFidelite()
    {
        int nbLimite = 0;

25
        //Si l'adherent a assez de points pour avoir une reduction
        while(nbPtsFidelite >= limitePtsFidelitePourReduction)
        {
            nbLimite++;
            nbPtsFidelite -= limitePtsFidelitePourReduction;

30 }

        return nbLimite * reductionFixeFidelite;
    }

35 public void miseAJourPtsFidelite(int ptsFidelite)
{
    nbPtsFidelite += ptsFidelite;
}

40 public String toString()
{
    return super.toString() + "\nNombre de Points de Fidelite : " + nbPtsFidelite;
}

45 }

```

## Employe.java

---

```

package laFac;

import java.util.ArrayList;

5 public class Employe extends Statut
{
    public Employe()
    {
10        super(NomStatut.Employe);
    }

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

    public double getReductionFixeFidelite()
    {
20        return 0;
    }

    public void miseAJourPtsFidelite(int ptsFidelite)
    {
25    }
}

```

---

## Visiteur.java

---

```

package laFac;

3 import java.util.ArrayList;

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

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

    public double getReductionFixeFidelite()
    {
18        return 0;
    }

    public void miseAJourPtsFidelite(int ptsFidelite)

```

```

23     {
        }
    }
}

```

---

## Produits

Produit.java

---

```

package laFac;

public abstract class Produit
4 {
    //Le nom est supposé unique et sert donc de clé primaire
    protected String nom;
    protected double prix;
    protected int stock;
9    protected boolean estEligibleAOffreProduit;
    protected NomCategorie categorie;

    public Produit(String nom, double prix, int stock, boolean estEligibleAOffreProduit)
    {
14        this.nom = nom;
        this.prix = prix;
        this.stock = stock;
        this.estEligibleAOffreProduit = estEligibleAOffreProduit;
    }

19    public void setEstEligibleAOffreProduit(boolean estEligibleAOffreProduit)
    {
        this.estEligibleAOffreProduit = estEligibleAOffreProduit;
    }

24    public boolean getEstEligibleAOffreProduit()
    {
        return estEligibleAOffreProduit;
    }

29    public String getNom()
    {
        return nom;
    }

34    public double getPrix()
    {
        return prix;
    }

39    public void setStock(int stock)
    {
        this.stock = stock;
    }
}

```

```

44     public String toStringPanier()
    {
        return nom + "\t" + prix + " €";
    }

49     public String toString()
    {
        String s = new String();
        s = "Nom Produit : " + nom + "\nPrix : " + prix + " €\nCategorie : " + categorie;
54         if (stock == 0)
            s = s + "\nRupture de stock.";
        else
            s = s + "\nEn stock : " + stock;

59         return s;
    }

    public boolean equals(Object obj)
    {
64         if (obj == null)
            return false;

        try
        {
69             Produit prod = (Produit) obj;
            return this.nom.equals(prod.nom);
        } catch (ClassCastException e)
        {
            return false;
74         }
    }
}

```

---

#### NomCategorie.java

---

```

package laFac;

3 public enum NomCategorie
{
    Culture, Hightech;
}

```

---

#### Culture.java

---

```

package laFac;

public abstract class Culture extends Produit
4 {
    public Culture(String nom, double prix, int stock, boolean estEligibleAOffreProduit)
    {
        super(nom, prix, stock, estEligibleAOffreProduit);
    }
}

```

```

    this.categorie = NomCategorie.Culture;
9      }
}

```

---

#### Hightech.java

---

```

package laFac;

public abstract class Hightech extends Produit
4 {
    protected int tempsGarantie;

    public Hightech(String nom, double prix, int stock, boolean estEligibleAOffreProduit ,
        int tempsGarantie)
    {
9        super(nom, prix, stock, estEligibleAOffreProduit);
        this.tempsGarantie = tempsGarantie;
        this.categorie = NomCategorie.Hightech;
    }

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

19 }

```

---

#### Livres.java

---

```

1 package laFac;

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

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

    public String toString()
16    {
        return "LIVRE : \nTitre : " + titre + "\nAuteur : " + auteur + "\n" +
            super.toString();
    }

}

```

---

## Spectacles.java

---

```

package laFac;

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

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

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

```

---

## Tablette.java

---

```

1 package laFac;

public class Tablette extends Hightech
{
    private String marque;

    6     public Tablette(double prix, int stock, int tempsGarantie, String marque)
    {
        super("Tablette " + marque, prix, stock, true, tempsGarantie);
        this.marque = marque;
    11     }

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

```

---

## Offres

## Offre.java

---

```

package laFac;

2 import java.util.ArrayList;

```



```

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

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

    public abstract double getReduction();

    17     public int getPtsFidelite()
    {
        return 0;
    }

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

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

```

---

#### OffreEmploye.java

---

```

package laFac;

    2     public class OffreEmploye extends Offre
    {

        /** Reduction en pourcentage. */
    7     public OffreEmploye(Produit prod, double reduction)
    {
        super(reduction);
        if (prod != null)
            produits.add(prod);
    12    }

    public double getReduction()
    {
        if (!produits.isEmpty())
    17         return produits.get(0).getPrix() * reduction;
        else
            return 0;
    }

```

```

22     public boolean miseAJourDate()
    {
        return true;
    }

27     public String toString()
    {
        if (!produits.isEmpty())
            return "OFFRE EMPLOYE : \nProduit : " + produits.get(0).getNom() +
                "\nReduction : " + (reduction * 100) + " %";
        else
32         return "OFFRE EMPLOYE NON APPLICABLE";
    }
}

```

---

#### OffreFidelite.java

---

```

package laFac;

public class OffreFidelite extends Offre
{
5     public int ptsFidelite;

    public OffreFidelite(Produit prod, int ptsFidelite)
    {
        super(0);
10        this.ptsFidelite = ptsFidelite;
        if (prod != null)
            produits.add(prod);
    }

15    public double getReduction()
    {
        return reduction;
    }

20    public int getPtsFidelite()
    {
        return ptsFidelite;
    }

25    public boolean miseAJourDate()
    {
        return true;
    }

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

```

```

35         return "OFFRE FIDELITE NON APPLICABLE";
    }
}

```

---

### OffreFlash.java

---

```

package laFac;

import java.util.ArrayList;

public class OffreFlash extends Offre
{
    // La date limite est défini en nombre de mise à 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)
    {
        super(reduction);
        this.dateExpiration = dateExpiration;
        if (prods != null)
            produits.addAll(prods);
    }

    public double getReduction()
    {
        double prix = 0;

        for (Produit prod : produits)
        {
            prix += prod.getPrix() * reduction;
        }

        return prix;
    }

    public void addProduit(Produit prod)
    {
        if (prod != null)
            produits.add(prod);
    }

    public boolean miseAJourDate()

```

```

47     {
        dateExpiration--;
        return (dateExpiration > 0);
    }

52     public String toString()
    {
        String s = new String();

        if (!produits.isEmpty())
67         {
            s += "OFFRE FLASH : \nProduits : \n";
            for (Produit prod : produits)
            {
                s += " - " + prod.getNom() + "\n";
62            }

            s += "Reduction : " + (reduction * 100) + " %\nDate d'Expiration : " +
                (dateExpiration - Contexte.getDateDuJour()) + " jours restants.";

            return s;
        } else
67         {
            return "OFFRE FLASH NON APPLICABLE";
        }
    }
}

```

---

#### OffreProduit.java

---

```

package laFac;

public class OffreProduit extends Offre
4 {
    // La date limite est défini en nombre de mise à jour du contexte.
    private int dateExpiration;

    /** Reduction en pourcentage. Si le produit n'est pas éligible on ne l'ajoute pas à la
        liste de produit. */
9    public OffreProduit(Produit prod, double reduction, int dateExpiration)
    {
        super(reduction);
        this.dateExpiration = dateExpiration;

14        if (prod.getEstEligibleAOffreProduit())
        {
            produits.add(prod);
        }
    }

19    public double getReduction()
    {
        if (!produits.isEmpty())
            return produits.get(0).getPrix() * reduction;
    }
}

```

```
24         else
            return 0;
    }

    public boolean miseAJourDate()
29    {
        dateExpiration--;
        return (dateExpiration > 0);
    }

34    public String toString()
    {
        if(!produits.isEmpty())
            return "OFFRE PRODUIT : \nProduit : " + produits.get(0).getNom() +
                "\nReduction : " + (reduction * 100) + " %\nDate d'Expiration : " +
                (dateExpiration - Contexte.getDateDuJour()) + " jours restants.";
        else
39        return "OFFRE PRODUIT NON APPLICABLE";
    }
}
```

---