

**Université Cadi Ayyad**  
**Ecole supérieure de technologie Essaouira**

Rapport de mini-projet Modélisation orienté objet  
Filière : Ingénierie des systèmes informatique et logiciels

**Une Solution Élaborée par les Design Patterns**  
**Command dans le Domaine de la Conception**

Réalisé par :

- Aya Attioui
- Oussama Bouslim

Encadré par:

-M.Hannane Grissette

# Tables des matières

Introduction-----	3
Présentation du projet-----	4
Problématique-----	10
Solution-----	11
Conclusion-----	20

# Introduction

Dans le contexte complexe du développement logiciel, la quête incessante de solutions élégantes et performantes pour résoudre des problèmes récurrents représente un défi omniprésent. Les Design Patterns, également connus sous le nom de modèles de conception, émergent comme des outils précieux dans cette recherche constante d'efficacité et de maintenabilité du code. Ces modèles fournissent des solutions prêtes à l'emploi pour des problèmes courants, tout en encourageant une approche modulaire et flexible tout au long du processus de conception logicielle.

Au cœur de cette exploration, le Design Pattern Command se distingue comme une stratégie ingénieuse qui encapsule une requête en tant qu'objet, offrant ainsi la possibilité de paramétrer les clients avec différentes requêtes, de mettre en file des opérations, et d'autoriser l'annulation d'opérations. Ce rapport s'engage dans une étude approfondie de l'application des Design Patterns Command, en mettant particulièrement l'accent sur leur utilisation pour résoudre des problèmes spécifiques dans le domaine du développement logiciel.

À travers une analyse approfondie, des exemples concrets et des retours d'expérience, l'objectif de ce rapport est de démontrer la puissance et la pertinence des Design Patterns Command dans la résolution de problématiques complexes. En explorant leurs avantages, leurs limitations, et en illustrant leur mise en œuvre pratique, nous visons à fournir une vision claire et concrète des bénéfices qu'ils peuvent apporter à la conception logicielle moderne.

# Présentation du projet

## 1. Introduction

Dans ce chapitre dédié à la réalisation d'un site e-commerce sécurisé avec un système de paiement, ainsi qu'à la détection des fraudes par carte de crédit, nous explorerons les étapes clés pour développer une plateforme e-commerce fiable et offrir une expérience utilisateur fluide lors des transactions en ligne. Nous aborderons l'importance de la détection des fraudes par carte de crédit et présenterons les techniques telles que la régression logistique et la forêt aléatoire utilisées pour identifier les transactions frauduleuses. Vous découvrirez les avantages et les différences entre ces deux approches. Préparez-vous à plonger dans le monde passionnant du commerce en ligne sécurisé et de la prévention des fraudes par carte de crédit.

## 2. Objectives

Pour atteindre ces objectifs, notre démarche consistera à développer un site e-commerce sécurisé, mettant en place un système de paiement fiable, et intégrant des mécanismes de détection des fraudes par carte de crédit.

- ✓ Concevoir une plateforme e-commerce conviviale et intuitive pour offrir une expérience utilisateur optimale lors des achats en ligne.

- ✓ Mettre en place des mesures de sécurité avancées pour protéger les données sensibles des utilisateurs, y compris les informations de paiement.

- ✓ Implémenter des mécanismes de détection des fraudes par carte de crédit pour identifier et prévenir les transactions frauduleuses.

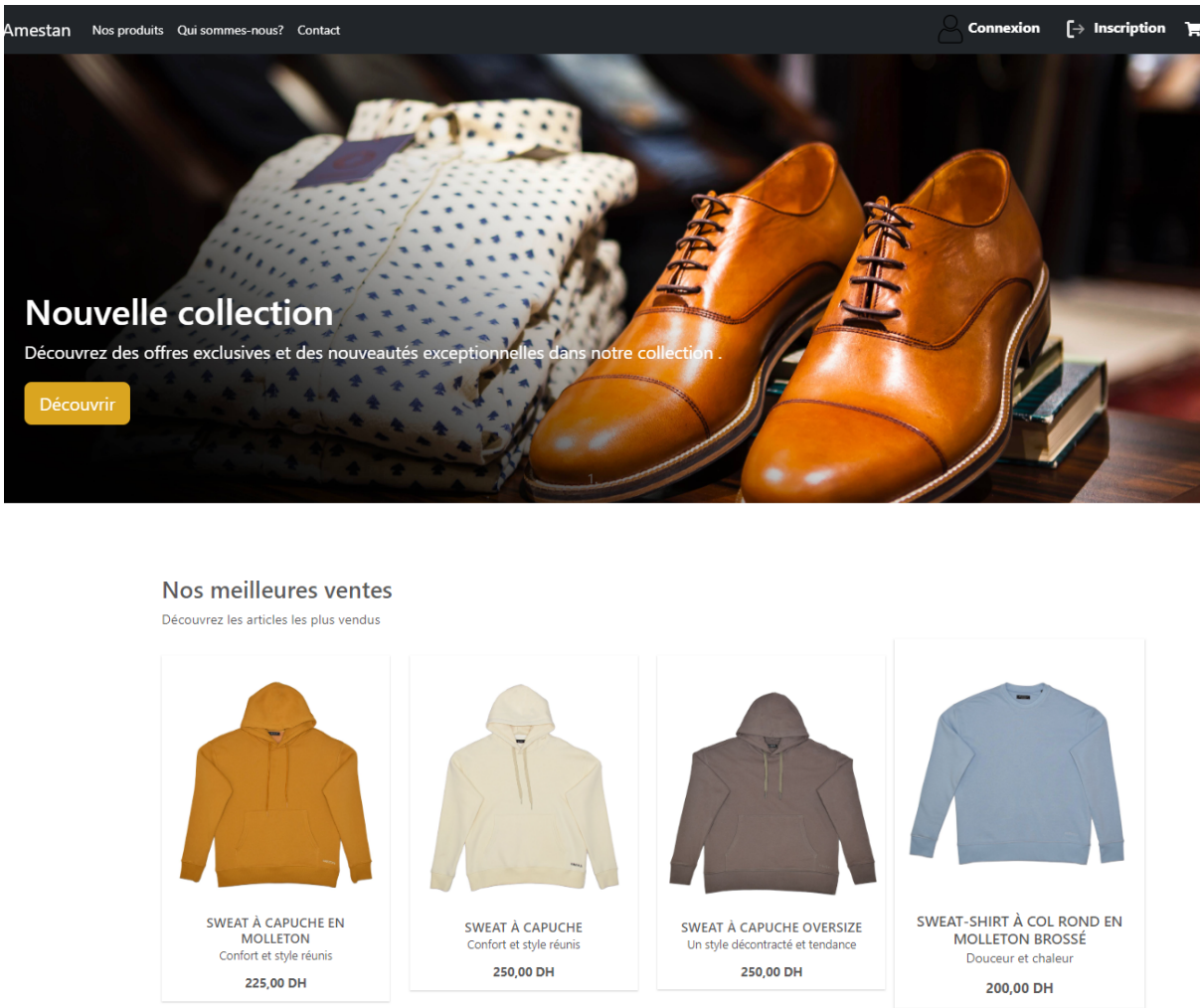
- ✓ Utiliser des techniques avancées telles que la régression logistique et la forêt aléatoire pour améliorer l'efficacité de la détection des fraudes.

- ✓ Fournir des recommandations et des bonnes pratiques pour assurer la conformité aux normes de sécurité et de confidentialité des données dans le domaine du commerce électronique.

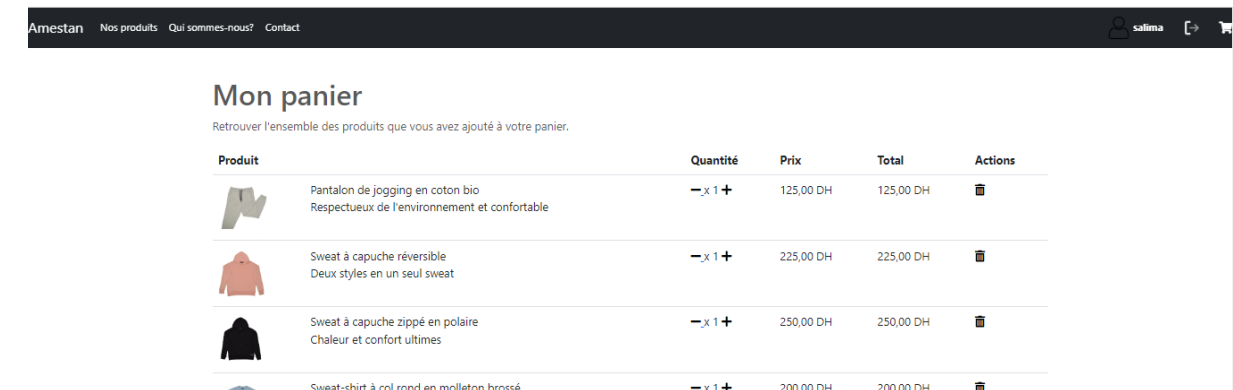
En atteignant ces objectifs, nous visons à créer un environnement e-commerce sécurisé, fiable et propice à la confiance des utilisateurs, tout en réduisant les risques de fraudes par carte de crédit

### 3. Les pages de l'application

#### 1- Page D'accueil



#### 2- Pannier




### 3- Page login


Amestan


Nos produits

Qui sommes-nous?

Contact

 Connexion

 Inscription



### Merci de vous connecter

Email

Votre adresse email

Mot de passe

Votre mot de passe

Se connecter

Mot de passe oublié ?

Ou souhaitez-vous créer un compte ?





**À PROPOS**

Alliant style et confort, nos

**LIENS RAPIDES**

Accueil

**SUIVEZ-NOUS**

**NOTRE BOUTIQUE**

Explorez notre boutique en ligne pour découvrir les dernières tendances de la mode


### 4- Page Inscription


Amestan


Nos produits

Qui sommes-nous?

Contact

 Connexion

 Inscription



### Inscription

First Name

Merci de saisir votre prénom

Last Name

Merci de saisir votre nom

Email

Merci de saisir votre adresse email

Mot de passe

Merci de saisir votre mot de passe.

Confirmez votre mot de passe

Merci de confirmez votre mot de passe.

S'inscrire





**À PROPOS**

Alliant style et

**LIENS RAPIDES**

Accueil

**SUIVEZ-NOUS**

**NOTRE BOUTIQUE**

Explorez notre boutique en ligne pour

## 5- Modification des informations

[Amestan](#) [Nos produits](#) [Qui sommes-nous?](#) [Contact](#)

salima

### Modifier mon mot de passe

Modifier mon mot de passe

Votre mot de passe a bien été mis à jour.

Mon adresse email  
salima.boussss@gmail.com

Mon prénom  
salima

Mon nom  
bouslim

Mon mot de passe actuel  
Veuillez saisir votre mot de passe actuel

Mon nouveau mot de passe  
Merci de saisir votre nouveau mot de passe.

Confirmez votre mot de passe  
Merci de confirmez votre nouveau mot de passe.

Mettre à jour

## 6- Paiement

←

Amestan

TEST MODE

Payer Amestan

800,18 MAD

Pantalon de jogging en coton bio

125,00 MAD

Sweat à capuche réversible

225,00 MAD

Sweat à capuche zippé en polaire

250,00 MAD

Sweat-shirt à col rond en molleton  
brossé

200,00 MAD


Afficher les 5 articles


Propulsé par stripe | [Conditions d'utilisation](#) | [Confidentialité](#)

### Payer par carte

E-mail salima.boussss@gmail.com [Connexion](#)

Informations de la carte

1234 1234 1234 1234 

MM / AA CVC 

Nom du titulaire de la carte

Pays ou région

Maroc

Payer

## 7- Commande

Amestan

Nos produits

Qui sommes-nous?

Contact

salima

Je passe ma commande

Choisissez vos préférences avant de passer votre commande sur AMESTAN.

Choisissez votre adresse de livraison

Ajouter une nouvelle adresse

Ma maison

ain daab

casablanca-MA

Choisissez votre transporteur

adward


la livraison gratuite

0,18 DH

Valider ma commande


Récap de ma commande

Retrouvez le récapitulatif de vos produits.




Pantalon de jogging en coton bio  
Respectueux de l'environnement et confortable

x 1




Sweat à capuche réversible  
Deux styles en un seul sweat

x 1



Sweat à capuche zippé en polaire  
Chaleur et confort ultimes

x 1



Sweat-shirt à col rond en molleton brossé  
Douceur et chaleur

x 1

## 8- Confirmation

Amestan

Nos produits

Qui sommes-nous?

Contact

salima

Mes commandes

Retour

Référence	Statut	Passée le	Produit(s)	Total	
31052023-64775172eeb17	Paiement accepté	31/05/2023	4	800,18 DH	Voir ma commande
29052023-647454d8ab48a	Paiement accepté	29/05/2023	2	55,18 DH	Voir ma commande

À PROPOS





Alliant style et confort, nos vêtements sont

LIENS RAPIDES

Accueil

Produits

SUIVEZ-NOUS



NOTRE BOUTIQUE

Explorez notre boutique en ligne pour découvrir les dernières tendances de la mode et trouvez les vêtements qui reflètent votre



# Problématique

Imaginons un scénario où nous concevons un système de gestion de panier d'achat pour un site e-commerce. La problématique centrale se pose dans la gestion des opérations variées liées aux commandes du panier, comprenant l'ajout, la suppression, la consultation et la modification des articles. En outre, l'enjeu réside dans la nécessité d'exécuter ces opérations de manière flexible et extensible, tout en veillant à maintenir un faible couplage entre les différentes classes impliquées dans le processus.

Comment pouvons-nous élaborer un mécanisme qui assure une manipulation agile des commandes du panier, en garantissant une extensibilité optimale tout en minimisant les interdépendances entre les composants du système ?

# Solution

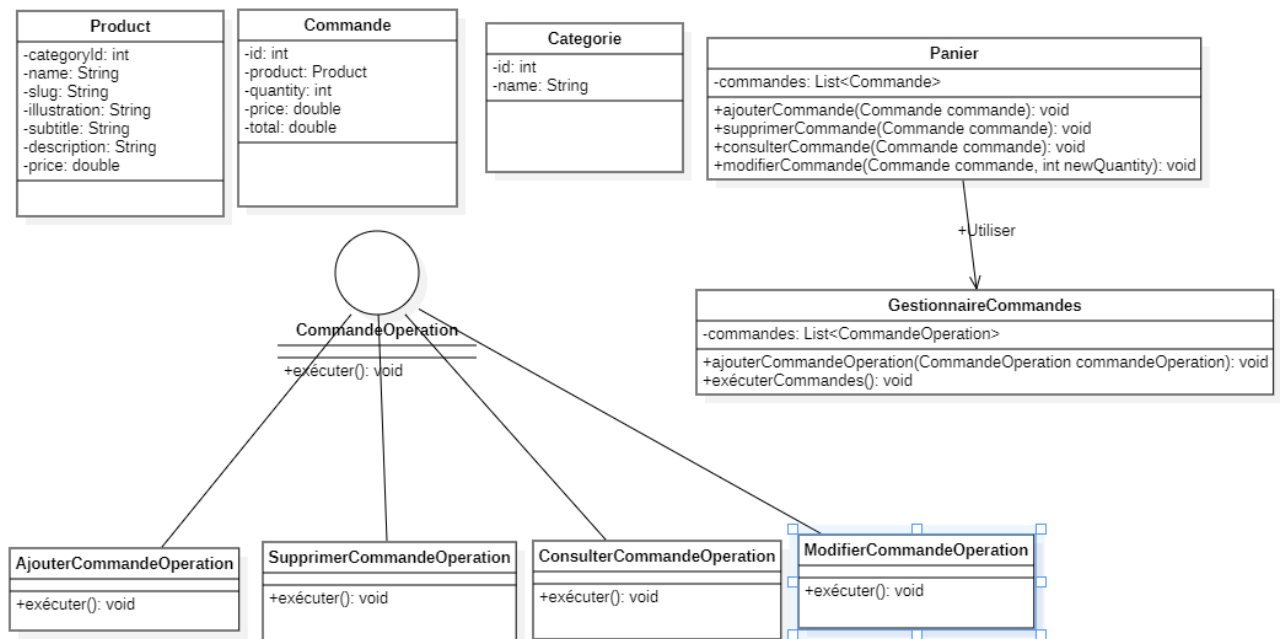
## 1. l'idée de solution

Pour répondre à cette problématique complexe de gestion du panier d'achat dans un contexte e-commerce, l'application du design pattern Command se présente comme une solution élégante et efficace. Le design pattern Command permet d'encapsuler chaque opération du panier (ajout, suppression, consultation et modification d'articles) sous la forme d'objets distincts, offrant ainsi une abstraction qui rend ces opérations indépendantes du contexte dans lequel elles sont exécutées.

La clé de cette approche réside dans la création d'une interface commune pour toutes les commandes, garantissant ainsi une uniformité dans la manière dont les opérations sont invoquées. Chaque commande spécifique, implémentant cette interface, encapsule la logique associée à une opération particulière. De cette manière, les commandes deviennent des entités autonomes, faciles à ajouter, supprimer ou modifier sans impacter les autres parties du système.

Par ailleurs, l'utilisation du design pattern Command permet une gestion flexible des opérations, puisqu'il devient possible de mettre en file, d'annuler ou de rejouer des commandes, offrant ainsi une manipulation agile du panier d'achat. De plus, la décomposition des opérations en objets distincts favorise une extensibilité optimale du système, permettant l'ajout de nouvelles fonctionnalités sans altérer les classes existantes.

En minimisant les interdépendances entre les différentes classes impliquées, le design pattern Command contribue à maintenir un faible couplage, favorisant ainsi la robustesse et la maintenabilité du système de gestion de panier d'achat. En conclusion, l'adoption du design pattern Command offre une solution structurée et élégante pour relever les défis soulevés par la gestion des commandes du panier dans un contexte e-commerce.



## 2. l'implémentation en Java

### a. Class Product

```

/**
 * and open the template in the editor.
 */
package stagedesignpattern;

/**
 *
 * @author H-R
 */
public class Product {

    private int categoryId;
    private String name;
    private String slug;
    private String illustration;
    private String subtitle;
    private String description;
    private double price;

    public Product(int categoryId, String name, String slug, String illustration, String subtitle, String description,
        this.categoryId = categoryId;
        this.name = name;
        this.slug = slug;
        this.illustration = illustration;
        this.subtitle = subtitle;
        this.description = description;
        this.price = price;
    }
}
  
```

```

[-] public int getCategoryId() {
    return categoryId;
}

[-] public String getName() {
    return name;
}

[-] public String getSlug() {
    return slug;
}

[-] public String getIllustration() {
    return illustration;
}

[-] public String getSubtitle() {
    return subtitle;
}

[-] public String getDescription() {
    return description;
}

[-] public double getPrice() {
    return price;
}

}

```

## b. Class Commande

```

L /**
    package stagedesignpattern;

[-] /**
    *
    * @author H-R
    */
    public class Commande {

        private int id;
        private Product product;
        private int quantity;
        private double price;
        private double total;

[-] public Commande(int id, Product product, int quantity, double price, double total) {
    this.id = id;
    this.product = product;
    this.quantity = quantity;
    this.price = price;
    this.total = total;
}

[-] public int getId() {
    return id;
}

[-] public Product getProduct() {
    return product;
}

```

```

[-] public int getQuantity() {
    return quantity;
}

[-] public double getPrice() {
    return price;
}

[-] public double getTotal() {
    return total;
}

[-] public void setQuantity(int quantity) {
    this.quantity = quantity;
}

[-] public void setTotal(double total) {
    this.total = total;
}
}

```

### c. Class Product

```

: /
package stagedesignpattern;

[-] /**
 *
 * @author H-R
 */
public class Categorie {

    private int id;
    private String name;

[-] public Categorie(int id, String name) {
    this.id = id;
    this.name = name;
}

[-] public int getId() {
    return id;
}

[-] public String getName() {
    return name;
}

}

```

d. Interface Product

```
L  /**
   package stagedesignpattern;

  - /**
    *
    * @author H-R
    */
   public interface CommandeOperation {

       void exécuter();

   }
```

e. Class AjouterCommandeOperation

```
package stagedesignpattern;

] /**
 *
 * @author H-R
- */
public class AjouterCommandeOperation implements CommandeOperation {

    private Panier panier;
    private Commande commande;

] public AjouterCommandeOperation(Panier panier, Commande commande) {
    this.panier = panier;
    this.commande = commande;
- }

] public void exécuter() {
    panier.ajouterCommande(commande);
- }

}
```

#### f. Class ConsulterCommandeOperation

```

    * and open the template in the editor.
    */
package stagedesignpattern;

/**
 *
 * @author H-R
 */
public class ConsulterCommandeOperation implements CommandeOperation {

    private Panier panier;
    private Commande commande;

    public ConsulterCommandeOperation(Panier panier, Commande commande) {
        this.panier = panier;
        this.commande = commande;
    }

    public void exécuter() {
        panier.consulterCommande(commande);
    }

}

```

#### g. Class ModifierCommandeOperation

```

    */
package stagedesignpattern;

/**
 *
 * @author H-R
 */
public class ModifierCommandeOperation implements CommandeOperation {

    private Panier panier;
    private Commande commande;
    private int newQuantity;

    public ModifierCommandeOperation(Panier panier, Commande commande, int newQuantity) {
        this.panier = panier;
        this.commande = commande;
        this.newQuantity = newQuantity;
    }

    public void exécuter() {
        panier.modifierCommande(commande, newQuantity);
    }

}

```

#### h. Class SupprimerCommandeOperation

```

L  */
   package stagedesignpattern;

   /**
    *
    * @author H-R
    */
   public class SupprimerCommandeOperation implements CommandeOperation {

       private Panier panier;
       private Commande commande;

       public SupprimerCommandeOperation(Panier panier, Commande commande) {
           this.panier = panier;
           this.commande = commande;
       }

       public void exécuter() {
           panier.supprimerCommande(commande);
       }

   }

```

#### i. Class AjouterCommandeOperation

```

   import java.util.ArrayList;
   import java.util.List;
   public class Panier {

       private List<Commande> commandes = new ArrayList<>();

       public void ajouterCommande(Commande commande) {
           commandes.add(commande);
           System.out.println("Commande ajoutée au panier.");
       }

       public void supprimerCommande(Commande commande) {
           commandes.remove(commande);
           System.out.println("Commande supprimée du panier.");
       }

   }

```



```

public void consulterCommande(Commande commande) {
    System.out.println("Détails de la commande :");
    System.out.println("ID : " + commande.getId());
    System.out.println("Produit : " + commande.getProduct().getName());
    System.out.println("Quantité : " + commande.getQuantity());
    System.out.println("Prix unitaire : " + commande.getPrice());
    System.out.println("Total : " + commande.getTotal());
}

public void modifierCommande(Commande commande, int newQuantity) {
    commande.setQuantity(newQuantity);
    commande.setTotal(newQuantity * commande.getPrice());
    System.out.println("Commande modifiée avec succès.");
}
}

```

#### j. Class AjouterCommandeOperation

```

package stagedesignpattern;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author H-R
 */
public class GestionnaireCommandes {

    private List<CommandeOperation> commandes = new ArrayList<>();

    public void ajouterCommandeOperation(CommandeOperation commandeOperation) {
        commandes.add(commandeOperation);
    }

    public void executerCommandes() {
        for (CommandeOperation commandeOperation : commandes) {
            commandeOperation.executer();
        }
        commandes.clear();
    }
}

```

## k. Class Main

```
- * @author H-R
- */
public class Main {
3   public static void main(String[] args) {
      |
      |
      |   Panier panier = new Panier();
      |   Product product1 = new Product(1, "Capuche", "Capuche", "Capuche.jpg", "Une Capuche styl  ", "Description du Capuch
      |   Commande command1 = new Commande(1, product1, 2, product1.getPrice(), 2 * product1.getPrice());
      |
      |   GestionnaireCommandes gestionnaireCommandes = new GestionnaireCommandes();
      |
      |   // Utilisation des commandes
      |   CommandeOperation ajouterCommandeOperation = new AjouterCommandeOperation(panier, command1);
      |   CommandeOperation consulterCommandeOperation = new ConsulterCommandeOperation(panier, command1);
      |   CommandeOperation modifierCommandeOperation = new ModifierCommandeOperation(panier, command1, 3);
      |
      |   gestionnaireCommandes.ajouterCommandeOperation(ajouterCommandeOperation);
      |   gestionnaireCommandes.ajouterCommandeOperation(consulterCommandeOperation);
      |   gestionnaireCommandes.ajouterCommandeOperation(modifierCommandeOperation);
      |
      |   // Ex  cution des commandes
      |   gestionnaireCommandes.ex  cuterCommandes();
-   }
}
```

# Conclusion

En conclusion, l'adoption du design pattern Command pour la gestion du panier d'achat a démontré son efficacité en offrant une structure modulaire, flexible, et extensible. En minimisant les interdépendances entre les différentes opérations, ce design pattern favorise une manipulation agile du panier tout en facilitant la maintenance du code. La mise en œuvre réussie de cette approche renforce la robustesse du système et prépare le terrain pour une évolution harmonieuse, répondant ainsi de manière optimale aux exigences changeantes d'un site e-commerce.