

## Gestion d'un panier pour un site de commerce

- Créer plusieurs variables contenant les noms et prix de chaque livre et les quantités voulues de chaque, et une variable contenant le coût total du panier.

```
public class Main {  
    public static void main(String[] args) {  
        String title1 = "Code Complete";  
        double price1 = 25;  
        int quantity1 = 1;  
        String title2 = "Effective Java";  
        double price2 = 32;  
        int quantity2 = 1;  
        String title3 = "Clean Code";  
        double price3 = 29.5;  
        int quantity3 = 2;  
    }  
}
```

- Ajouter une instruction pour afficher en console le coût du panier, sous le format Le coût du panier est xxx euros.

```
public class Main {  
    public static void main(String[] args) {  
        String title1 = "Code Complete";  
        double price1 = 25;  
        int quantity1 = 1;  
        String title2 = "Effective Java";  
        double price2 = 32;  
        int quantity2 = 1;  
        String title3 = "Clean Code";  
        double price3 = 29.5;  
        int quantity3 = 2;  
        double totalPrice = price1 * quantity1 + price2 * quantity2 + price3 * quantity3;  
        System.out.println("Le coût du panier est " + totalPrice + " euros");  
    }  
}
```

- Rajouter des instructions pour afficher le contenu du panier, sous la forme :

Code Complete, 1 exemplaire(s) à 25.0 euro l'unité, total : 25.0 euros

Effective Java, 1 exemplaire(s) à 25.0 euro l'unité, total : 32.0 euros

Clean Code, 2 exemplaire(s) à 29.5 euro l'unité, total : 59.0 euros

Le coût du panier est 116.0 euros

```
public class Main {
    public static void main(String[] args) {
        String title1 = "Code Complete";
        double price1 = 25;
        int quantity1 = 1;
        String title2 = "Effective Java";
        double price2 = 32;
        int quantity2 = 1;
        String title3 = "Clean Code";
        double price3 = 29.5;
        int quantity3 = 2;
        double totalPrice = price1 * quantity1 + price2 * quantity2 + price3 * quantity3;
        System.out.println("Le coût du panier est " + totalPrice + " euros");
    }
}
```

6. Ajouter la déclaration de titre, puis de la même façon, déclarer un prix unitaire et un nombre d'exemplaires pour les objets de la classe `Book`. Choisissez le type et le nom les plus appropriés dans chaque cas !

```
public class Book {
    String title;
    int quantity;
    double price;
}
```

7. Replacez-vous dans la classe `Main`. Au début de la méthode `main`, déclarer trois variables de type `Book`, une par livre.

```
public class Main {
    public static void main(String[] args) {
        Book codeComplete;
        Book effectiveJava;
        Book cleanCode;
    }
}
```

8. De même, initialiser les trois variables de type `Book`, avec trois objets différents de la classe `Book`.

```
public class Main {
    public static void main(String[] args) {
        Book codeComplete = new Book();
    }
}
```

```

    Book effectiveJava = new Book();
    Book cleanCode = new Book();
}
}

```

9. Essayer d'afficher le titre de `codeComplete`. Que constatez-vous ?

Cela affiche `null` pour le titre de `codeComplete`. La valeur `null` est la valeur par défaut des variables de type `String` qui n'ont pas été initialisées avec une valeur.

10. Utiliser des affectations pour attribuer à chaque livre son titre. Modifier les instructions d'affichage pour qu'elles utilisent la attribut `title` de chaque livre, plutôt que les variables de type `String` définies dans le `main` auparavant.

```

public class Main {
    public static void main(String[] args) {
        Book codeComplete = new Book();
        Book effectiveJava = new Book();
        Book cleanCode = new Book();
        codeComplete.title = "Code Complete";
        double price1 = 25;
        int quantity1 = 1;
        effectiveJava.title = "Effective Java";
        double price2 = 32;
        int quantity2 = 1;
        cleanCode.title = "Clean Code";
        double price3 = 29.5;
        int quantity3 = 2;
        double totalPrice = price1 * quantity1 + price2 * quantity2 + price3 * quantity3;
        System.out.println(codeComplete.title + ", " + quantity1 + " exemplaire(s) à "
            + price1 + " euro l'unité, total : " + quantity1 * price1 + " euros");
        System.out.println(effectiveJava.title + ", " + quantity2 + " exemplaire(s) à "
            + price2 + " euro l'unité, total : " + quantity2 * price2 + " euros");
        System.out.println(cleanCode.title + ", " + quantity3 + " exemplaire(s) à "
            + price3 + " euro l'unité, total : " + quantity3 * price3 + " euros");
        System.out.println("Le coût du panier est " + totalPrice + " euros");
    }
}

```

13. Simplifier la méthode `main`, de sorte qu'il ne reste que les trois variables de type `Book`, la variable du prix du panier, les affectations et les instructions d'affichage du panier.

```

public class Main {
    public static void main(String[] args) {
        Book codeComplete = new Book();

```

```

Book effectiveJava = new Book();
Book cleanCode = new Book();
codeComplete.title = "Code Complete";
codeComplete.price = 25;
codeComplete.quantity = 1;
effectiveJava.title = "Effective Java";
effectiveJava.price = 32;
effectiveJava.quantity = 1;
cleanCode.title = "Clean Code";
cleanCode.price = 29.5;
cleanCode.quantity = 2;
double totalPrice = codeComplete.price * codeComplete.quantity
    + effectiveJava.price * effectiveJava.quantity
    + cleanCode.price * cleanCode.quantity;
System.out.println(codeComplete.title + ", " + codeComplete.quantity
    + " exemplaire(s) à " + codeComplete.price
    + " euro l'unité, total : "
    + codeComplete.quantity * codeComplete.price + " euros");
System.out.println(effectiveJava.title + ", " + effectiveJava.quantity
    + " exemplaire(s) à " + codeComplete.price
    + " euro l'unité, total : "
    + effectiveJava.quantity * effectiveJava.price + " euros");
System.out.println(cleanCode.title + ", " + cleanCode.quantity
    + " exemplaire(s) à " + cleanCode.price
    + " euro l'unité, total : "
    + cleanCode.quantity * cleanCode.price + " euros");
System.out.println("Le coût du panier est " + totalPrice + " euros");
}
}

```

14. Écrire la méthode `getTotalPrice` dans la classe `Book` de telle sorte qu'elle renvoie le produit du prix du livre par son nombre d'exemplaire.

```

public class Book {
    String title;
    int quantity;
    double price;

    double getTotalPrice(){
        return price * quantity;
    }
}

```

15. Réécrire le calcul du coût du panier avec trois appels de méthodes et des additions.

```
double totalPrice = codeComplete.getTotalPrice()
    + effectiveJava.getTotalPrice()
    + cleanCode.getTotalPrice();
```

16. Écrire la déclaration de la méthode `display` (laisser les instructions vides).

```
public class Book {
    void display(){

    }
}
```

17. Pour les instructions, commencer par récupérer une des instructions d’affichage actuellement dans `main`, et la couper-coller vers `display`. Corriger l’instruction d’affichage pour la rendre correcte.

```
public class Book {
    void display(){
        System.out.println(title + ", " + quantity
            + " exemplaire(s) à " + price
            + " euro l'unité, total : "
            + getTotalPrice() + " euros");
    }
}
```

18. Dans la classe `Main`, remplacer les instructions d’affichage des livres par des appels à la méthode `display`.

```
public class Main {
    public static void main(String[] args) {
        Book codeComplete = new Book();
        Book effectiveJava = new Book();
        Book cleanCode = new Book();
        codeComplete.title = "Code Complete";
        codeComplete.price = 25;
        codeComplete.quantity = 1;
        effectiveJava.title = "Effective Java";
        effectiveJava.price = 32;
        effectiveJava.quantity = 1;
        cleanCode.title = "Clean Code";
        cleanCode.price = 29.5;
        cleanCode.quantity = 2;
        double totalPrice = codeComplete.getTotalPrice()
            + effectiveJava.getTotalPrice()
            + cleanCode.getTotalPrice();
        codeComplete.display();
    }
}
```

```

    effectiveJava.display();
    cleanCode.display();
    System.out.println("Le coût du panier est " + totalPrice + " euros");
}
}

```

19. Créer une déclaration d'un constructeur dans la classe `Book`. Ce constructeur a 3 paramètres : `initTitle` de type `String`, `initPrice` de type `double` et `initQuantity` de type `int`. Pour l'instant, laisser le bloc d'instructions vide.

```

public Book(String initTitle, double initPrice, int initQuantity) {

}

```

20. Corriger les trois instructions `new` de la méthode `main`.

```

public class Main {
    public static void main(String[] args) {
        Book codeComplete = new Book("Code Complete", 25, 1);
        Book effectiveJava = new Book("Effective Java", 32, 1);
        Book cleanCode = new Book("Clean Code", 29.5, 2);
        double totalPrice = codeComplete.getTotalPrice()
            + effectiveJava.getTotalPrice()
            + cleanCode.getTotalPrice();
        codeComplete.display();
        effectiveJava.display();
        cleanCode.display();
        System.out.println("Le coût du panier est " + totalPrice + " euros");
    }
}

```

21. Écrire trois instructions, chacune initialisant une attribut du livre avec la valeur d'un des paramètres du constructeur.

```

public class Book {
    public Book(String initTitle, double initPrice, int initQuantity) {
        title = initTitle;
        quantity = initQuantity;
        price = initPrice;
    }
}

```

22. Modifier la méthode `display()` de `Book` afin d'obtenir l'affichage suivant :

Code Complete, 1 exemplaire à 25.0 euro l'unité, total : 25.0 euros

Effective Java, 1 exemplaire à 25.0 euro l'unité, total : 32.0 euros

Clean Code, 2 exemplaires à 29.5 euro l'unité, total : 59.0 euros

Le coût du panier est 116.0 euros

```
public class Book {
    String title;
    int quantity;
    double price;

    double getTotalPrice(){
        return price * quantity;
    }

    public Book(String initTitle, double initPrice, int initQuantity) {
        title = initTitle;
        quantity = initQuantity;
        price = initPrice;
    }

    void display(){
        String quantityDescription = "exemplaire";
        if(quantity > 1){
            quantityDescription += "s";
        }
        System.out.println(title + ", " + quantity
            + " " + quantityDescription + " à " + price
            + " euro l'unité, total : "
            + getTotalPrice() + " euros");
    }
}
```

## Gestion du panier

1. Créer une classe `Cart`.

```
public class Cart {
}
```

2. Sur la première ligne de la classe `Cart` (donc avant la déclaration de la classe), ajouter `import java.util.List;`. Sur la ligne d'après, procéder de même pour le terme `java.util.ArrayList`.

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

public class Cart {
    List<Book> listOfBooks = new ArrayList<>();
}
```

```
}
```

3. Ajouter une méthode `addBook` dans la classe `Cart`.

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

public class Cart {
    List<Book> listOfBooks = new ArrayList<>();

    void addBook(Book book){

    }
}
```

4. Terminer l'instruction d'ajout du livre dans la liste, en fournissant à la méthode `add` son argument qui est le livre à ajouter.

```
public class Cart {
    void addBook(Book book){
        listOfBooks.add(book);
    }
}
```

5. Recopier la boucle, et mettre une instruction provoquant l'affichage du livre.

```
public class Cart {
    void display(){
        for(Book oneBook : listOfBooks){
            oneBook.display();
        }
    }
}
```

6. Dans la classe `Main`, créer une nouvelle variable `cart` de type `Cart`, initialisée avec un nouvel objet.

```
public class Main {
    public static void main(String[] args) {
        Cart cart = new Cart();
    }
}
```

7. Utiliser la méthode `addBook` pour ajouter les trois livres dans le panier.

```

public class Main {
    public static void main(String[] args) {
        Cart cart = new Cart();
        Book codeComplete = new Book("Code Complete", 25, 1);
        cart.addBook(codeComplete);
        Book effectiveJava = new Book("Effective Java", 32, 1);
        cart.addBook(effectiveJava);
        Book cleanCode = new Book("Clean Code", 29.5, 2);
        cart.addBook(cleanCode);
        codeComplete.display();
        effectiveJava.display();
        cleanCode.display();
        double totalPrice = codeComplete.getTotalPrice()
            + effectiveJava.getTotalPrice()
            + cleanCode.getTotalPrice();
        System.out.println("Le coût du panier est " + totalPrice + " euros");
    }
}

```

8. Remplacer l’affichage de chaque livre par un seul appel de méthode à l’objet `cart`.

```

public class Main {
    public static void main(String[] args) {
        Cart cart = new Cart();
        Book codeComplete = new Book("Code Complete", 25, 1);
        cart.addBook(codeComplete);
        Book effectiveJava = new Book("Effective Java", 32, 1);
        cart.addBook(effectiveJava);
        Book cleanCode = new Book("Clean Code", 29.5, 2);
        cart.addBook(cleanCode);
        cart.display();
        double totalPrice = codeComplete.getTotalPrice()
            + effectiveJava.getTotalPrice()
            + cleanCode.getTotalPrice();
        System.out.println("Le coût du panier est " + totalPrice + " euros");
    }
}

```

9. Ajouter une méthode `getCost` à la classe `Cart`, permettant de calculer et retourner le coût du panier.

```

public class Cart {
    double getCost(){
        int sum = 0;
        for(Book oneBook : listOfBooks){
            sum += oneBook.getTotalPrice();
        }
    }
}

```

```

    }
    return sum;
}
}

```

10. Modifier la méthode `display` de la classe `Cart`, pour qu'elle affiche d'abord le coût total du panier, puis les livres.

```

public class Cart {
    List<Book> listOfBooks = new ArrayList<>();

    void addBook(Book book){
        listOfBooks.add(book);
    }
    void display(){
        for(Book oneBook : listOfBooks){
            oneBook.display();
        }
        System.out.println("Le coût du panier est " + getCost() + " euros");
    }
    double getCost(){
        int sum = 0;
        for(Book oneBook : listOfBooks){
            sum += oneBook.getTotalPrice();
        }
        return sum;
    }
}

```

11. Modifier aussi la méthode `main` de la classe `Main`, pour que l'affichage reste le même qu'avant.

```

public class Main3 {
    public static void main(String[] args) {
        Cart cart = new Cart();
        Book codeComplete = new Book("Code Complete", 25, 1);
        cart.addBook(codeComplete);
        Book effectiveJava = new Book("Effective Java", 32, 1);
        cart.addBook(effectiveJava);
        Book cleanCode = new Book("Clean Code", 29.5, 2);
        cart.addBook(cleanCode);
        cart.display();
    }
}

```