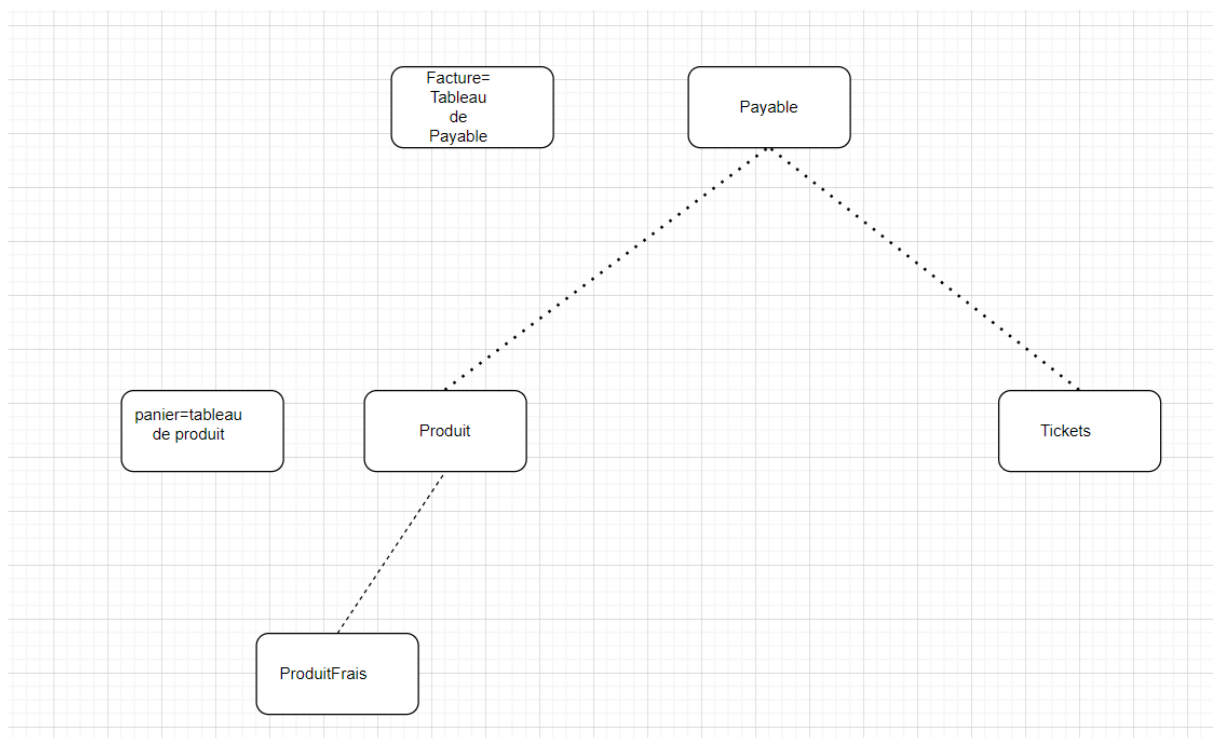


TP6 JAVA AKOBI BANCONLE

Sommaire:

Partie 1 -Articles.....	1
Partie 2: Panier.....	2
Partie 3 - produits frais.....	8
Partie 4:Facture:.....	9

Schéma des héritages:



Partie 1 -Articles

```
package TP6Java;  
class Produit{  
    private final String nom;  
    private final long prix;
```

TP6 JAVA AKOBI BANCONLE

```
public Produit(String nom, long prix){
    this.nom=nom;
    this.prix=prix;
}

public String getNom() {
    return nom;
}

public long getPrix() {
    return prix;
}

public String toString(){
    double prix_euros=prix/100.0;
    return String.format("%s: %.2f", nom, prix_euros);
}
}

class Test {
    public static void main(String[] args) {
        Produit produit = new Produit("cereales", 500);
        System.out.println(produit); // affiche:
cereales: 5.00 €
        Produit Lait = new Produit("lait",403);
        System.out.println(Lait); // affiche: lait: 4.03
€
    }
}
```

Affichage test pour les questions 1 et 2:

```
cereales: 5.00 €
lait: 4.03 €
```

Partie 2: Panier

Question 1: Ici on crée une classe Panier avec la possibilité de retirer , d'ajouter , et de connaître le prix total des produits du panier:

TP6 JAVA AKOBI BANCONLE

Code:

```
class Panier{
    ArrayList<Produit> panier;

    public Panier() {
        panier=new ArrayList<>();
    }

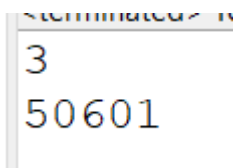
    public void ajoutProduit(Produit nouveau_produit) {
        panier.add(nouveau_produit);
    }

    public void SupprimerProduit(Produit produit) {
        panier.remove(produit);
    }

    public int nombreProduit() {
        return panier.size();
    }

    // Complexité d'ordre O(n)
    public long prixTotal(){
        long prixtotal=0;
        for(Produit p:panier) {
            prixtotal+=p.getPrix();
        }
        return prixtotal;
    }
}
```

Affichage question 1:



```
terminated
3
50601
```

TP6 JAVA AKOBI BANCONLE

*Question 2. Ici on va surcharger la classe Produit avec la méthode equals pour qu'il puisse comparer le contenu lors de la suppression:
ainsi on ajoute ceci à produit:*

```
public String toString(){
    double prix_euros=prix/100.0;
    return String.format("%s: %.2f €", nom, prix_euros);
}
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null || getClass() != obj.getClass()) {
        return false;
    }
    Produit other = (Produit) obj;
    return this.nom.equals(other.nom) && this.prix ==
other.prix;
}
```

Et on voit que l'affichage est bien 0:

A screenshot of a Java Swing window with a light gray border. Inside the window, the number '0' is displayed in a blue, monospaced font.

Question 3:

Ici on ajoutera une condition à la méthode d'ajout:

```
public void ajoutProduit(Produit nouveau_produit) {
    if(nouveau_produit.poids>10000) {
        System.out.println("Votre produit dépasse les
10 kg->IMPOSSIBLE A AJOUTER!!!");
    }
    else {
        panier.add(nouveau_produit);
    }
}
```

TP6 JAVA AKOBI BANCONLE

Question 4 et 5 : On ajoutera ici un ID à panier puis une méthode toString le panier :

```
class Panier{
    ArrayList<Produit> panier;
    private static int Id_panier=1;
    private final int Id;

    public Panier() {
        this.Id=Id_panier++;
        panier=new ArrayList<>();
    }

    public void ajoutProduit(Produit nouveau_produit) {
        if(nouveau_produit.poids>10000) {
            System.out.println("Votre produit dépasse les
10 kg->IMPOSSIBLE A AJOUTER!!!");
        }
        else {
            panier.add(nouveau_produit);
        }
    }

    public boolean supprimerProduit(Produit produit) {
        if(panier.contains(produit)) {
            panier.remove(produit);
            return true;
        }
        else {
            return false;
        }
    }

    public int nombreProduit() {
        return panier.size();
    }

    // Complexité d'ordre 0(n)
    public long prixTotal(){
        long prixtotal=0;
    }
}
```

TP6 JAVA AKOBI BANCONLE

```
        for(Produit p:panier) {
            prixtotal+=p.getPrix();
        }
        return prixtotal;
    }

    int getId() {
        return Id;
    }
}

public String toString() {
    StringBuilder resultat = new StringBuilder();

    resultat.append("Panier
").append(Id).append("[").append(nombreProduit()).append("
article(s)]");
    for (Produit p : panier) {
        resultat.append("\n").append(p.toString());
    }
    return resultat.toString();
}
```

Et on constate que l'affichage se passe comme prévu:

TP6 JAVA AKOBI BANCONLE

```
terminated: test (/path/application/erp/program...)
1
1
2
3
Panier 1[2 article(s)]
cereales: 5.01 €
caviar: 500.00 €
Panier 2[0 article(s)]
Panier 3[1 article(s)]
eau: 5.00 €
|
```

Partie 3 - produits frais

On ajoute dans cette partie une classe produitfrais qui est une sorte de produit en ajoutant juste une date limite de consommation:

```
class ProduitFrais extends Produit{
    String DateLimiteConso;
    ProduitFrais(String nom, long prix, int
poids,String DateLimiteConso){
        super(nom, prix, poids);
        this.DateLimiteConso=DateLimiteConso;
    }

    public String toString() {
        double prix_euros=super.getPrix()/100.0;
```

TP6 JAVA AKOBI BANCONLE

```
        return String.format("B:%s %s: %.2f  
€", DateLimiteConso, super.getNom(), prix_euros);  
    }  
}
```

Affichage réussi pour la question 1:

```
cereales: 5.00 €  
B:01-12-2022 Saumon: 14.50 €
```

Affiche correct pour la vérification également:

```
<terminated> test (7) java Application C:\Program Files\Java\jre-  
Panier 1[1 article(s)]  
B:01-12-2022 sardine: 5.00 €
```

Partie 4:Facture:

Question 1: Création de la classe Ticket:

```
class Ticket implements Payable {  
    private final String reference;  
    private final long prix;  
    public Ticket(String reference, long prix) {  
        this.reference = reference;  
    }  
}
```


TP6 JAVA AKOBI BANCONLE

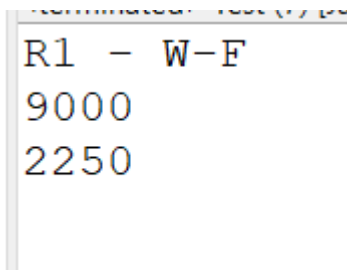
```
        this.prix = prix;
    }
    public String label() {
        return reference;
    }
    public long cout() {
        return prix;
    }
    public long taxe() {

        return (long) (prix * 0.25);
    }
}
```

Question 2 : Créons une interface Payable pour que ticket soit une sorte de payable:

```
interface Payable {
    String label();
    long cout();
    long taxe();
}
```

Affichage du test:



A screenshot of a terminal window with a dark background. The text displayed is as follows:

```
terminated: test 0/1 ps
R1 - W-F
9000
2250
```

Question 3 et 4 et 5 : On crée ici une classe Facture où on stockera pour tout les produits , les différents payables(tickets , produits) donc il faut ajouter implements à produit pour qu'il soit une sorte de payable (et comme produit frais est une sorte de produit , donc il sera aussi une sorte de payable).

Après cela on respecte les taxes selon le sujet pour ajouter les implémentations des methodes taxes, labels et cout à produit.

TP6 JAVA AKOBI BANCONLE

Donc pour la classe facture on a ceci:

```
class Facture {
private final ArrayList<Payable> Apayer;
private long coutTotal;
private long taxeTotale;
public Facture() {
    this.Apayer = new ArrayList<>();
    this.coutTotal = 0;
    this.taxeTotale = 0;
}
public void ajout(Payable p) {
    Apayer.add(p);
    coutTotal += p.cout();
    taxeTotale += p.taxe();
}
}
```

Pour les classes produits et produits classe on maintenant ceci:

```
class Produit implements Payable{
    private final String nom;
    private final long prix;
    int poids;

    public Produit(String nom, long prix, int poids){
        this.nom=nom;
        this.prix=prix;
        this.poids=poids;
    }

    public String getNom() {
        return nom;
    }

    public long getPrix() {
        return prix;
    }
    public long getPoids() {
        return poids;
    }
}
```

TP6 JAVA AKOBI BANCONLE

```
public String toString(){
    double prix_euros=prix/100.0;
    return String.format("%s: %.2f €", nom, prix_euros);
}

public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null || getClass() != obj.getClass()) {
        return false;
    }
    Produit other = (Produit) obj;
    return this.nom.equals(other.nom) && this.prix ==
other.prix;
}

    public String label() {
        return getNom();
    }
    public long cout() {
        return getPrix();
    }
    public long taxe() {
        return (long) (getPrix() * 0.10);
    }
}

class ProduitFrais extends Produit{
    String DateLimiteConso;
    ProduitFrais(String nom, long prix, int poids,String
DateLimiteConso){
        super(nom, prix, poids);
        this.DateLimiteConso=DateLimiteConso;
    }

    public String toString() {
        double prix_euros=super.getPrix()/100.0;
        return String.format("B:%s %s: %.2f
€",DateLimiteConso, super.getNom(),prix_euros);
    }

    public long taxe() {
```

TP6 JAVA AKOBI BANCONLE

```
        double reduction = getPoids() * 0.001;
        return (long) (super.taxe() - reduction);
    }
}
```

Et les affichages pour chaque test sont:

-Pour les questions 1 et 2:

```
R1 - W-F
9000
2250
```

-pour les question 3 et 4:

Tout s'est bien ajouté , et il n'y a pas d'affichage

-l'affichage pour la question 5 est aussi bonne(juste qu'il y a une erreur dans le sujet):

```
50
49
148
```

Question 6:

Ici on ajoute les methodes montantTotal() et taxeTotal() à facture:

```
class Facture {
    private final ArrayList<Payable> Apayer;
    private long coutTotal;
    private long taxeTotale;
    public Facture() {
        this.Apayer = new ArrayList<>();
    }
}
```

TP6 JAVA AKOBI BANCONLE

```
        this.coutTotal = 0;
        this.taxeTotale = 0;
    }
    public void ajout(Payable p) {
        Apayer.add(p);
        coutTotal += p.cout();
        taxeTotale += p.taxe();
    }
    // Méthodes montantTotal() et taxeTotale() sans parcourir la
    // liste à nouveau
    public long montantTotal() {
        return coutTotal;
    }
    public long taxeTotale() {
        return taxeTotale;
    }
}
```

Et on va que l'affichage a bien réussi

```
50
49
148
2500
247
```

Ci-joint le fichier Test.Java final