Faculté des Sciences
Département de Physique

Travaux pratique 4 - Mini Projet encadré Programmation Orientée Objet en Java Licence Universitaire Spécialisée

L'objectif de ce TP est d'utiliser les concepts de l'héritage, du polymorphisme, des méthodes abstraites et des interfaces.

Exercice : Gestion des stocks d'un magasin

Nous voulons mettre en place un système qui gère les stocks d'un magasin. D'abord il faut comprendre la hiérarchie des classes et des interfaces comme suit. Cela est nécessaire pour voir clairement les paramètres de retour des méthodes ainsi que la liste de leurs paramètres formels. Voici une spécification du programme en termes de classes et des interfaces :

Il existe trois interfaces :

- Vendable par pièce: l'interface pour les produits qui se vendent par pièces
  - La méthode **vendre** reçoit la quantité vendue du produit et retourne le revenu du magasin et modifie le stock.

- Vendable par kilogramme: l'interface pour les produits qui se vendent par kilogramme
  - La méthode **vendre** reçoit la quantité vendue du produit et retourne le revenu du magasin et modifie le stock

- Susceptible d'être vendu en solde
  - Les méthodes: **lancer_le_solde()** baisse le prix du produit par le pourcentage donné et **terminer_le_solde()** augmente le prix du produit par le pourcentage donné.

Une classe générale des Articles

- Propriétés:
  - prix d'achat: le prix pour lequel le supermarché achète le produit
  - prix de vente: le prix pour lequel le supermarché vend le produit
  - nom: le nom du produit
  - fournisseur: le nom du fournisseur du produit

- Méthodes (autre que le constructeur):
  - Méthode renvoyant le taux de rendement rendement = prix_achat - prix_vente/prix_achat
  - description des caractéristiques du produit (méthode info()) (les prix, le nom, le fournisseur; rendement)
  - Cette classe n'implémente aucune interface.

Deux classes dérivées des Articles

- Chaque classe dérivée des articles respecte la règle suivante: au moment de la construction de l'objet, le stock est vide.

- La classe des **Article_elec** (électroménagers) avec la propriété supplémentaire: nombre de pièces en stock et les méthodes supplémentaires (autre que le constructeur): remplir_le_stock(), description des caractéristiques du produit (méthode **info()**) (les prix, le nom, le fournisseur; rendement; stock). De plus, Il faut implémenter les interfaces correspondantes à cette classe.

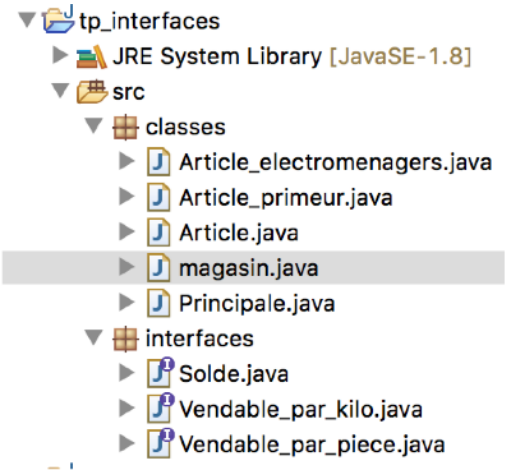
- La classe **Articles_primeurs (qui se vendent par kilogramme)** avec la propriété supplémentaire: quantité en stock et les méthodes supplémentaires: remplir le stock, description des caractéristiques du produit (méthode **info()**) (les prix, le nom, le fournisseur; rendement; stock). De plus Il faut implémenter les interfaces correspondantes à cette classe, sachant que les primeurs ne peuvent pas être vendues en solde.

Une classe pour les magasins

- Propriétés déclarées public :
  - Dépenses: le coût d'achat des produits
  - Revenus: les revenus après la vente des produits
  - Produits: deux tableaux de **deux articles** (électroménagers et primeurs)
- Méthodes (autre que le constructeur):
  - Méthode info () : décrit l'état du magasin (dépense, revenu et rendement + affichage des informations des articles appartenant au magasin).
  - Méthode renvoyant le taux de rendement ((Revenu - Depense)/Depense)

Dans cette classe, l'objectif est de modéliser un magasin avec sa marchandise, ses revenus et ses dépenses. A la création d'un magasin, celui ci possède 2 articles électroménagers et deux articles primeurs mis dans deux tableaux.

L'architecture de votre projet dans votre environnement de développement devra ressembler à cela :



Questions:

1. Coder les interfaces et des classes.
2. Dans la méthode main de votre classe Principale, créez un magasin puis définissez les articles à vendre. Ensuite remplissez les stocks et simulez les achats et les ventes.

```
package classes;

public class Principale {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		magasin mon_magasin = new magasin(); // Création d'un magasin

	  // Définir les produits à vendre dans le magasin


	  //remplir les stocks affichage de la description du magasin. Attention, il
	  //faut que les dépenses soient modifiées…



	  //vendre article_electro n°1 et affichage de la description du magasin
	  System.out.println("\n------------------Après vente Article
	                      Elec n°1-------");

	  ...
	   mon_magasin.info(); //afficher la description du magasin

	  //vendre article_electro n°2 et affichage de la description du magasin
	 System.out.println("\n------------------Après vente Article
	                      Elec n°2——");
	  ...

	//tester les soldes et proposez d'autres simulations

	}

}
```