# Polytech Lille GIS4
# Contrôle de Programmation Par Objets

8 janvier 2019

*Tous documents papiers autorisés. Lire le sujet entièrement avant de répondre aux questions. Répondre en respectant l'ordre des questions et en rappelant leur numéro. Programmer vos solutions en Java (ne pas vous préoccuper des packages). Vous pouvez (devez) utiliser ce qui a été spécifié dans une question même si vous n'y avez pas répondu.*

Les supermarchés offrent en général un site WEB de vente en ligne de leurs articles avec un service de livraison à domicile.

## 1 Articles

### 1.1 Hiérarchie de classes d'articles

On part du diagramme de classes UML de la figure 1 qui décrit la hiérarchie de classes simplifiée des articles avec leurs attributs de base (cette hiérarchie doit rester extensible).

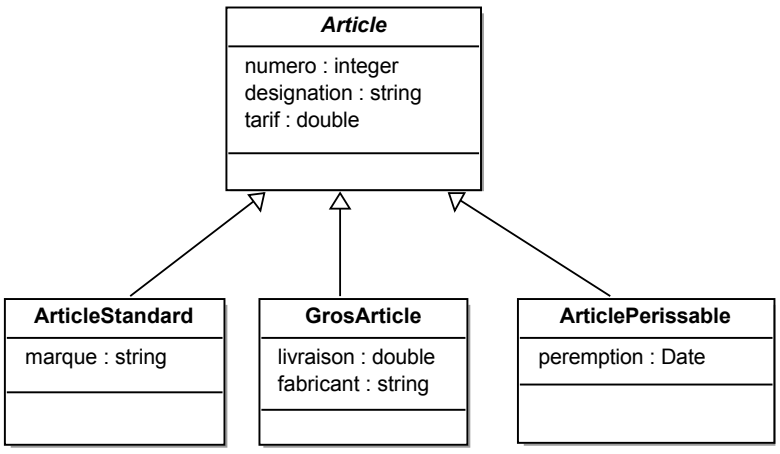


FIGURE 1 – articles

— tout article (classe racine abstraite `Article`) a les attributs suivants :
  — `numero` : son numéro unique en stock
  — `designation` : désignation du produit par exemple `"stylo", "eponge", "lave-linge", "table", "lait", "6 oeufs", ...`

  - `tarif` : son tarif de base.
- les `ArticleStandard` (par exemple `"stylo", "eponge"`) ont en plus un attribut `marque` (la marque du produit).
- les `GrosArticle` (par exemple `"lave-linge", "table"`) ont en plus :
  - un coût de livraison (attribut `livraison`) qui viendra s'ajouter à leur prix de vente
  - un attribut `fabricant` (les coordonnées du fabricant du produit).
- les `ArticlePerissable` (par exemple `"lait", "6 oeufs"`) ont en plus une date de péremption (attribut `peremption`) au delà de laquelle ils ne sont plus consommables donc plus vendus (la classe `Date` est donnée ci-dessous, ne pas la programmer).

**Question (3 points)**

Programmer cette hiérarchie de classes avec leurs attributs.

**Précisions**
- il n'est pas demandé de programmer de méthodes ni de constructeurs à ce niveau.
- la classe `Date` est donnée ci-dessous (ne pas la programmer).

```
public class Date {
  public static Date today() // renvoie la date du jour
  public int diff(Date d) // renvoie (d - this) en nombres de jours
}
```

### 1.2 Prix de vente des articles

Tous les articles ont un prix de vente calculé par leur méthode `'double prix()'` définie comme suit :
- `ArticleStandard` : `tarif` de base
- `GrosArticle` : `tarif` de base + son coût de `livraison`
- `ArticlePerissable` : `tarif` de base avec les réductions suivantes si la date d'achat (= `Date.today()`) est :
  - la veille (1 jour avant) de sa date de péremption : -50%
  - l'avant-veille (2 jours avant) de sa date de péremption : -25%.

**Question (3 points)**

Programmer dans les classes d'articles leur méthode `'double prix()'`.

**Précision** dans cette question faire l'hypothèse que les `ArticlePerissable` ne sont pas périmés (date de péremption > `Date.today()`).

## 2 Supermarché en ligne

On complète le diagramme de classes de la figure 1 par la figure 2. Un supermarché en ligne est modélisé par la classe `Supermarche` et le stock des articles qu'il met en vente est modélisé par l'association de rôle `stock` vers la classe `Article`.

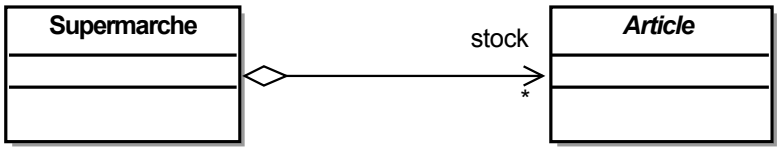


FIGURE 2 – stock

Pour faciliter l'accès aux articles de même désignation, on représente ce stock par une `TreeMap` qui associe à chaque désignation (`String`) la liste des articles correspondants :

```
class Supermarche {
  TreeMap<String, List<Article» stock;
  ...
}
```

### Question (2,5 points)

Dans la classe `Supermarche` écrire une méthode `'List<Article> getArticles(String designation)'` qui retourne la liste des articles associée à la désignation passée en paramètre ou provoque une exception `'DesignationInconnueException'` (à programmer) si la désignation n'existe pas.

## 3 Gestion journalière du stock

Tous les jours d'ouverture le supermarché doit effectuer des traitements sur son stock tels que :

1. la suppression des articles périssables qui sont périmés (question 3.1)
2. son réapprovisionnement suite aux arrivages d'articles qu'il a commandés auprès de ses fournisseurs (question 3.2)

### 3.1 Suppression des articles périmés

Pour optimiser cela le supermarché gère une liste `'perissables'` des désignations correspondant aux articles périssables :

```
class Supermarche {
  List<String> perissables;
  ...
}
```

### Question (3 points)

Programmer dans la classe `Supermarche` une méthode `'void supprimerPerimes()'` qui supprime du stock les articles périssables qui sont périmés (date de péremption $\leq$ `Date.today()`).

**Indications**

- c'est le supermarché qui gère sa liste `'perissables'` de désignations d'articles périssables et donc on peut considérer que toutes ces désignations existent bien dans sa `TreeMap stock`.
- il peut être nécessaire de programmer d'autres méthodes dans les classes concernées.
- pour rappel : la méthode `'remove(int i)'` des `List` permet de supprimer leur élément d'indice `i`.

### 3.2 Réapprovisonnement

#### Question (3,5 points)

Programmer dans la classe `Supermarche` une méthode `'void reapprovisionner(List<Article> arrivage)'` qui :

1. ajoute les articles de la liste `arrivage` dans son stock (dans les listes d'articles correspondant à leur désignation).
2. pour chaque désignation dans la `TreeMap stock`, trie la liste des articles correspondants par prix de vente croissant.

**Indications**

— considérer que toutes les désignations des articles de la liste `arrivage` existent dans le stock du supermarché (c'est lui qui les a commandés auprès de ses fournisseurs).
— considérer que la suppression des articles périmés (3.1) a été effectuée avant.
— il peut être nécessaire de compléter d'autres classes concernées.

## 4 Vente d'articles

Les traitements de gestion journalière du stock (Partie 3) ayant été effectués par le supermarché, on s'intéresse finalement à la vente d'articles aux clients (à la date du jour `Date.today()`).

### 4.1 Question (1,5 points)

Programmer dans la classe `Supermarche` une méthode `'Article fournir(String designation)'` qui retourne l'article le moins cher correspondant à la désignation passée en paramètre ou provoque :

— l'exception `DesignationInconnueException` si la désignation n'existe pas
— une exception `NonDisponibleException` si le stock est vide pour cette désignation.

### 4.2 Question (3,5 points)

Programmer dans la classe `Supermarche` une méthode `'Facture commander(List<String> commande)'` définie comme suit :

— le paramètre `commande` est une liste de désignations d'articles à fournir au client en utilisant la méthode `'Article fournir(String designation)'` précédente (4.1).
— retourner un objet `Facture` (classe à programmer) constitué de :
  — la liste des articles fournis.
  — la liste des désignations d'articles qui ne peuvent être fournis, soit parce que la désignation est inconnue (`DesignationInconnueException`) ou qu'il n'y a pas d'article en stock correspondant à cette désignation (`NonDisponibleException`).
  — le coût total (des articles fournis).