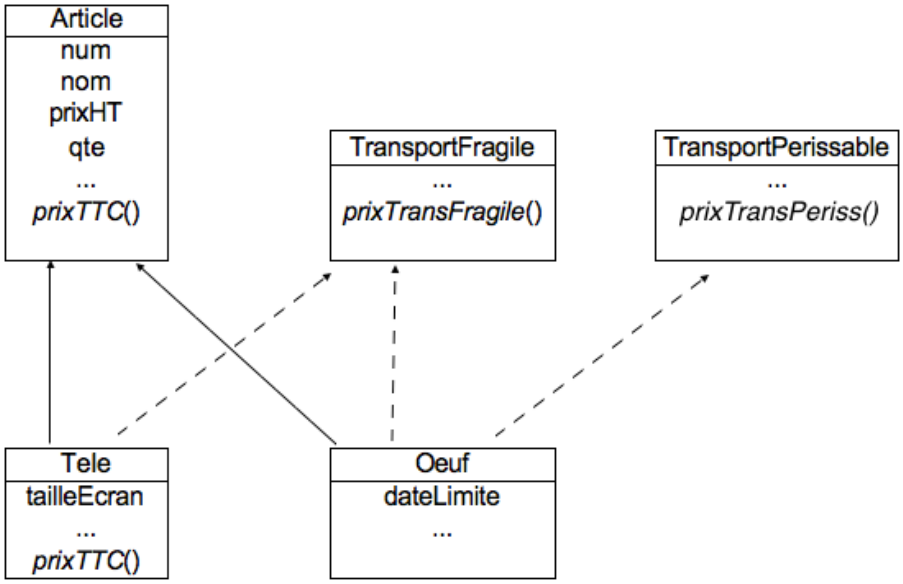
Contrôle de Programmation par objets en Java
Juin 2019, 1A

Najib Tounsi

Correction détaillée.

Question I)

Soit la classe `Article` vue en cours/TP. La conception suivante rajoute des éléments nouveaux pour avoir les prix de transport augmentés de 5% pour les articles périssables, et de 10% pour les fragiles. Le prixTTC des télés a une TVA de 20%.



1) Que suggère pour vous cette conception?
(voir `https://www.emi.ma/ntounsi/COURS/Java/PooJavaPart-3.html#(63)`)

Dans ce schéma il y a une classe Article qui a deux sous-classes (trait plein) Tele et Oeuf. Tele a un champ supplémentaire tailleEcran et Oeuf un champ dateLimite. Tele redéfinit aussi la méthode héritée prixTTC().
Par ailleurs (trait pointillé), la sous classe Oeuf doit en plus définir les méthodes prixTransportFragile() et prixTransportPerissable() "héritée" des "unités" TransportFragile et TransportPerissable. Tout comme la sous classe Tele doit en plus définir la méthode prixTransportFragile().

2) Supposer la classe `Article` déjà existante. Créer en Java les différents éléments de cette conception.

En Java les “unités” mentionnées précédemment seront des interfaces.

```
 interface TransportFragile {
    public double prixTransportFragile();
     // ...
}

interface TransportPerissable {
    public double prixTransportPerissable();
    // ...
}

// La classe Article (supposée déjà définie)
// class Article {/*... */}

// La classe Tele
class Tele extends Article implements TransportFragile {
   // ...
     public double prixTTC(){ return prixHT * 1.2;}
          // prixTTC redéfinie pour Tele

     public double prixTransportFragile() {
        // Calcul effectif du prix transport avec 10%
         return prixTTC()*0.1;
    }
};

// La classe Oeuf
class Oeuf extends Article implements TransportFragile,
TransportPerissable {
    // ...
    public double prixTransportFragile() {
        // Calcul effectif du prix transport avec 10%
          return prixTTC()*0.1;
    }

    public double prixTransportPerissable() {
        // Calcul effectif du prix transport Oeuf avec 5%
            return prixTTC()*0.05;
    }
  // ...
}
```

Question II)

On voudrait faire en Java une API appelée `Utilitaires` avec les opérations suivantes :

i. Imprimer un objet.
ii. Convertir un objet vers une chaîne de caractère imprimable
iii. Transformer un objet vers un format CVS (exemple : ’`valeur; valeur;`…’ pour tous les champs)
iv. Transformer un objet vers un format JSON (structure ‘`{champ:valeur, champ:valeur, …}`‘ pour tous les champs
v. Comparer deux objets entre-eux (`true` pour égaux, `false` sinon)

1) Définir une interface Java pour cette specification.

Voir aussi https://www.emi.ma/ntounsi/COURS/Java/TD/tdJava7-bis.html

```
interface Utilitaires {

    public void print();
            // Affiche sur l'unité standard de sortie

    public String toString();
            // Formate un objet vers un texte imprimable

    public String toCSV();
            // Formate un objet vers CSV

    public String tpJSON();
            // Formate un objet vers un texte imprimable

   public Boolean compare (object o);
            // compare this avec o et  retourne true ou false selon le cas
}
```

2) Implémenter cette interface avec la classe `Article` précédente (rajouter juste ce qu'il faut en plus).

```
class Article implements Utilitaires{

    protected int numero = 0;
    protected String nom = new String("spécimen");
    protected double prixHT = 1.;
    protected int qte;
// ... méthodes déjà existantes

    public void print(){
        System.out.println(this.toString());
    }
    public String toString(){
        return numero+" "+nom+" "+prixHT+" "+qte;
    }
    public String toCSV(){
        return numero+";"+nom+";"+prixHT+";"+qte;
    }

    public String toJSON(){
        return "{"+"numero:"+numero+", nom:"+nom +
        ", prixHT:"+prixHT+", qte:"+qte+"}" ;
    }

   public Boolean compare (Object o){
      // on choisit de comparer les numéros
        if (numero == ((Article)o).numero)
            return true;
        else return false;
   }
};
```

Question III)

1) Quel est le résultat du programme suivant :

```
class Exam {
    public static void main(String[] args) {
        String a = "Bonjour Java !";
        StringBuffer b = new StringBuffer();
        int i = 0;
        char c;
        do {
            c = a.charAt(i++);
            b.append(c);
        } while (c != 'J');
        System.out.println(b);
```

```
        }
    }
```

Ce programme affiche la chaîne **b** qui contient le le texte « **Bonjour J** ». En effet, La chaîne **b** est construite à partir de **a** jusqu'au caractère « **J** » (majuscule) inclus.

2) Pourquoi la variable *a* est `String` et *b* `StringBuffer` ?

La variable b est *à modifier* par adjonction successives de lettres. Elle doit être de type `StringBuffer`, ou `StringBuilder`.