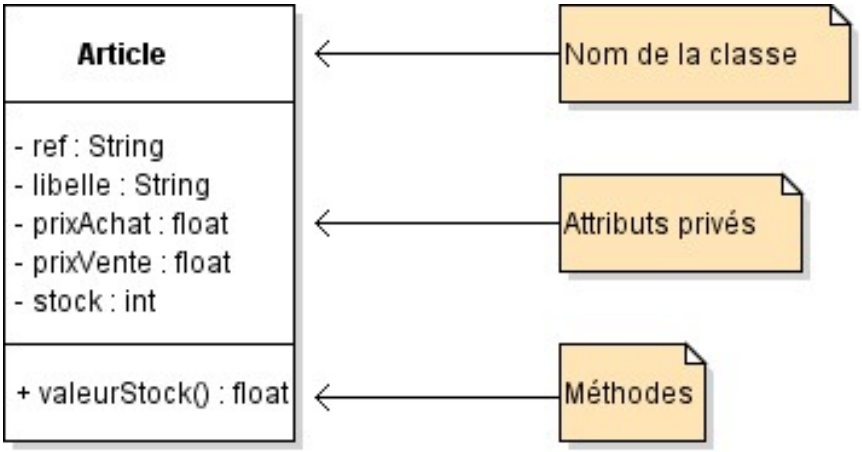
2) L'encapsulation

Voilà un diagramme de classes UML (Unified Modeling Language) :

UML est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique.



Le terme "**objet**" désigne la valorisation des attributs d'une classe, comme un gâteau qui est créé à partir d'ingrédients mis dans un moule.

- L'**objet** correspondrait au gâteau,
- les **attributs** correspondraient aux ingrédients,
- la **classe** correspondrait au moule.

Le terme "**encapsulation**" signifie donc de mettre au sein d'une même capsule (l'objet) des attributs et des méthodes. Les attributs servent à décrire un objet alors que les méthodes permettent de manipuler l'objet.

Quelques remarques :

- Un nom de classe commence par une **Majuscule** (par convention de nommage UpperCamelCase),
- le signe – signifie **privé,**
- le signe + signifie **public,**
- une méthode avec type de retour void (vide) est une **procédure,**
- une méthode avec un type de retour autre que void est une **fonction.**

Traduction Java

Ici je suis en conception de classe !

```
public class Article{
    //Attributs privés
    private String ref;
    private String libelle;
    private float prixAchat;
    private float prixVente;
    private int stock;

    //Méthode publique
    public float valeurStock(){
        return this.prixVente * this.stock;
    }
}
```

Remarque : le mot clé **this** en java signifie que l'on manipule un attribut ou une méthode **de l'objet courant**.

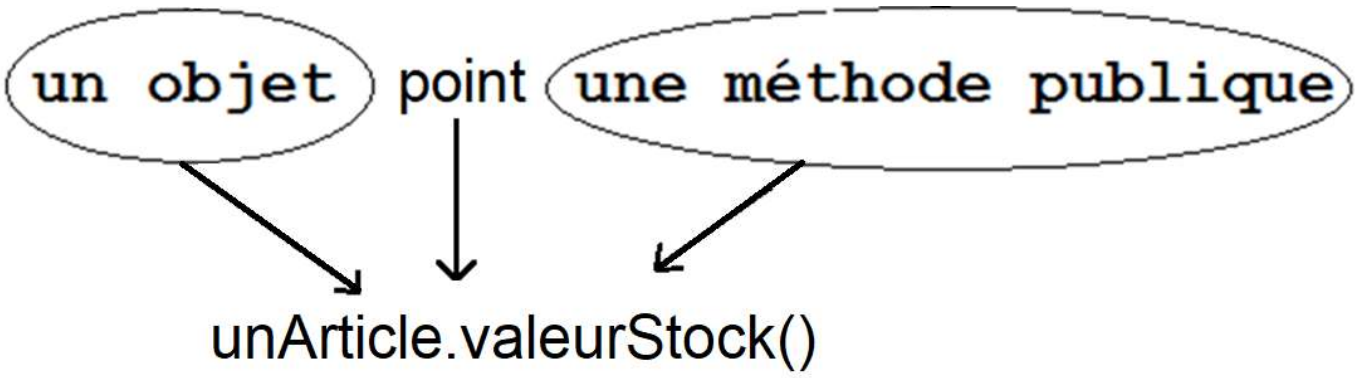
Utilisons maintenant la classe Article dans un programme de test :

Ici je suis en utilisation d'objet !

```
public class TestArticle{
    public static void main(String[] args){
        Article unArticle;                  //Déclaration de la variable objet
        unArticle = new Article();          //Instanciation

        //Affichage de la valeur du stock en appelant la méthode valeurStock
        System.out.println("La valeur du stock est de " + unArticle.valeurStock() + "€");
    }
}
```

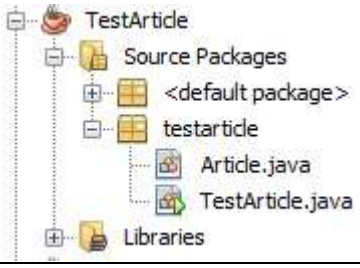
Considérons la partie de code suivante qui permet d'utiliser un objet :



Le principe est toujours d'appliquer à un **objet** une **méthode publique** préfixée d'un **point**.

Exercice 2 :

Créer un projet TestArticle et coder la classe Article et le programme principal TestArticle selon l'architecture suivante :



A l'exécution nous obtenons ceci :

La valeur du stock est de 0.0 €

Ce résultat est dû au fait que nous avons instancié un objet Article sans valoriser ses attributs. Pour instancier un objet et valoriser ses attributs, nous allons créer et utiliser un **constructeur**.

Si on reprend notre métaphore du gâteau :

- la **classe** correspondrait au moule,
- les **attributs** correspondraient aux ingrédients,
- le **constructeur** correspondrait à la recette,
- et le gâteau confectionné serait l'**objet**