Les exercices de la première section sont à terminer pendant la séance, celui de la seconde section pourra être terminé à la maison, en exercice.
Notez que vous aurez bientôt une première épreuve de contrôle en TD pour évaluer votre compréhension des notions *public/private*, $final$, et des notations $A.x$,$A.f()$, $a.y$, $a.g()$ indispensables pour la suite.

# 1 Manipuler des confitures

## 1.1 Définition des confitures

On définit une classe `Confiture` qui aura comme attributs **privés**
- un attribut `fruit` de type chaîne de caractères ;
- un attribut `proportion` de type entier qui correspondra au pourcentage de fruit dans la confiture ;
- un attribut `cal` de type entier qui correspondra au nombre de calories par 100 grammes de la confiture.

1. Écrivez la classe `Confiture` avec un constructeur public adapté.
2. Écrivez un deuxième constructeur qui ne prend en argument que la nature du fruit et le nombre de calories ; la proportion sera initialisée à 50. (Pensez que vous pouvez réutiliser le premier constructeur en fixant un paramètre)
3. Écrivez une méthode publique **d'objet** (c.à.d non statique) `description()` et qui renvoie une chaîne de caractères le décrivant (par exemple : "Confiture de fraise, 50% de fruit, 120 calories aux 100 grammes").
4. Dans une méthode `main` située dans une nouvelle classe `Test`, créez un objet de type `Confiture` et affichez sa description.
5. Dans la classe `Confiture`, écrivez une méthode publique d'objet qui prend en argument une quantité en grammes, et donne le nombre de calories correspondant à cette quantité pour cette confiture. (Il faut simplement faire un calcul qui respecte les proportions)
6. Écrivez une méthode de signature `public boolean egal(Confiture c)` qui s'adresse à une couverture *courante* et qui regarde si oui ou non elle a les mêmes attributs que la confiture `c`. (pour savoir si deux objets `s1` et `s2` de type `String` sont égaux, utilisez l'expression `s1.equals(s2)` qui est fournie par `java` et retourne un booléen)
7. On écrit le bout de code suivant situé dans la méthode `main` de la classe `Test`.
   Quelles lignes ne compilent pas, que produisent les autres ?

```
Confiture c1 = new Confiture("fraise", 50, 120);
Confiture c2 = new Confiture("fraise", 50, 120);
System.out.println(c1.egal(c2));
System.out.println(c1==c2);
System.out.println(c1.fruit);
```

8. On voudrait que l'attribut `fruit` ne puisse pas être modifié, même par une méthode de la classe `Confiture` ; comment faire ?

9. Écrivez une méthode qui retourne la valeur de l'attribut `fruit`. Écrivez-en une qui permet de modifier l'attribut `cal`. De quelles familles sont ces méthodes ?
10. En fait la valeur calorique dépend principalement de la quantité de sucre, qui est de 387 Kca pour 100 g, la valeur calorique du fruit est négligeable. Stockez cette valeur dans une variable adéquate.
11. Écrivez un modifieur de proportion en précisant son domaine d'utilisation.

### 1.2 Mettre les confitures en pots

On définit une classe `Pot` qui représente des pots de confiture. Pour chaque pot, on précisera la confiture qu'il contient et sa quantité en grammes.

1. Écrivez la classe `Pot` avec un constructeur public adapté.
2. Écrivez une méthode publique `description` qui renvoie une chaîne de caractères le décrivant. Remarquez qu'il n'y a pas d'ambiguïté avec la méthode `description` de `Confiture`, et que vous pouvez l'utiliser pour définir celle ci.
3. On veut numéroter les pots de confitures, à partir de 1, dans l'ordre de leur création. Mettez en place ce mécanisme.
4. Écrivez une méthode statique qui retourne le dernier numéro attribué.
5. Dans votre classe `Test` créez plusieurs instances de `Pot`, affichez leurs descriptions, puis affichez le dernier numéro attribué.

## 2 Température

Le but de cet exercice est d'écrire une classe représentant la température dans l'une ou l'autre des trois unités possibles "Kelvin", "Celsius" ou "Fahrenheit".
Les méthodes écrites devront toutes être des `méthodes d'objet.`

1. Définissez une classe `Temperature`, décrite par un `double` représentant la température, et un `String` représentant l'unité.
2. Définissez un constructeur sans arguments qui lorsqu'il est utilisé produit un objet `Temperature` à zéro Kelvin.
3. Définissez un deuxième constructeur prenant en argument un `double` et un `String` (Si l'unité n'est pas reconnue elle sera interprétée en Kelvin).
4. Définissez un troisième constructeur prenant en argument une `Temperature` et réalisant une copie de celle-ci.
5. Définissez des méthodes permettant d'afficher et de modifier chaque élément d'une `Temperature` (sans vous poser de questions de conversions, c'est abordé dans la suite).
6. Définissez une méthode privée `conversionKC`, non statique, produisant un nouvel objet `Temperature`. Lorsque `this` est bien en Kelvin le résultat sera sa conversion en Celsius, sinon elle ne produira pas de nouvel objet. On rappelle la formule $T_C = T_K - 273.15$ .
7. Supposons que l'on ait écrit suffisamment de méthodes de conversions sur le modèle de la précédente (on rappelle par exemple la formule $T_F = 9/5 * T_C + 32$). Écrivez une méthode `get` qui prend en argument une unité et renvoie la valeur numérique d'un objet `Temperature` dans l'unité spécifiée en argument.
8. Comment tester l'égalité de deux objets `Temperature`s (même valeur et même unité) ?
9. Définir une méthode `plusBasseQue` permettant de comparer deux `Temperature`s.