L2 TP C++

Héritage Polymorphisme Collections

SOUPE AUX LEGUMES

L'objectif est d'écrire quelques classes simples permettant de représenter des légumes, avec lesquels on peut faire de la soupe.

Les lignes de code fournies sont à écrire dans des fonctions externes, telle que la fonction principale ***main***.

PARTIE I : Pommes de terre, carottes

1) Dans un espace de noms **soup**, écrire une **classe Carrot** représentant une carotte avec son poids en kilogrammes et sa longueur en centimètres de sorte que :

les lignes de code ci-dessous	**produisent l'affichage suivant:**
Carrot c = Carrot(0.25, 30) cout << c << endl;	Carotte: [0.25 kg - 30 cm]

2) Dans le même espace de noms **soup**, écrire une **classe Potatoe** représentant une pomme de terre avec son poids en kilogrammes et un nombre de germes (les fameux "yeux") de sorte que :

les lignes de code ci-dessous	**produisent l'affichage suivant:**
Potatoe pdt = Potatoe(0.3, 10); cout << pdt << endl;	Patate: [0.3 kg - 10 yeux]

3) La pomme de terre ci-dessus a beaucoup de germes (10 pour un poids de 300 grammes).

   En règle générale, les pommes de terre ont par défaut un germe par tranche de 100 grammes.

   Ajoutez à la **classe Potatoe** ce qu'il faut pour créer des pommes de terre qui respectent cette règle générale sans avoir à spécifier explicitement le nombre de germes de sorte que :

les lignes de code ci-dessous	**Produisent l'affichage suivant :**
Potatoe pdt1 = Potatoe(0.3, 10); Potatoe pdt2 = Potatoe(0.3); Potatoe pdt3 = Potatoe(0.75); cout << pdt1 << endl; cout << pdt2 << endl; cout << pdt3 << endl;	Patate: [0.3 kg - 10 yeux] Patate: [0.3 kg - 3 yeux] Patate: [0.75 kg - 7 yeux]

PARTIE II : Soupe de légumes

1) Comme on risque d'avoir plein de légumes (carottes, pommes de terre, poireaux...) mais que tous auront un poids, on souhaite créer une **classe Vegetable** pour "**factoriser**" cet attribut, de sorte que le code suivant puisse compiler et que Pourquoi les 2 lignes suivantes ne compilent pas ?

```
Vegetable pdt = Potatoe(0.3, 10);
Vegetable c = Carrot(0.25, 30);
```

les lignes de code ci-dessous	**produisent l'affichage suivant:**
Vegetable * pdt = new Potatoe(0.3, 10); Vegetable * c = new Carrot(0.25, 30); cout << *pdt << endl; cout << *c << endl;	Patate: [0.3 kg - 10 yeux] Carotte: [0.25 kg - 30 cm]

2) On veut maintenant définir une **classe Soup** représentant une soupe de légumes par la liste de ses ingrédients (légumes de **type Vegetable**).
Vous pouvez utiliser une **classe container de la STL**, ou par défaut un tableau.
Une fois qu'une soupe est créée, on lui ajoute chaque légume un par un, grâce à la méthode **add()** oomme illustré ci dessous. A tout moment, on peut demander l'affichage de la soupe.
Écrire la **classe Soup** de sorte que le code suivant s'exécute :

```
Soup s = Soup();
s.add(new Potatoe(0.3, 10));
s.add(new Carrot(0.25, 30));
cout << s << endl;
s.add(new Potatoe(0.500));
s.add(new Potatoe(0.150));
s.add(new Carrot(0.20, 25));
s.add(new Potatoe(0.450, 2));
cout << s << endl;
```

3) Toujours dans la **classe Soup**, on souhaite avoir une **méthode getPeelingWeight()** qui donne le poids des épluchures générées lors de la préparation de cette soupe.
Par défaut, le poids d'épluchure des légumes est de 10% de leur poids.
Néanmoins, dans le cas particulier des pommes de terre, le poids d'épluchure est ce pourcentage (10%) auquel il faut ajouter 10 grammes par germe (oeil).
Modifiez les classes nécessaires pour réaliser ces fonctionnalités et pouvoir exécuter le code suivant :

```
Soup s = Soup();
s.add(new Potatoe(0.3, 10));
s.add(new Carrot(0.25, 30));
cout << s.getPeelingWeight() << endl;
```

4) On souhaite maintenant que le pourcentage du poids du légume perdu en épluchure (10% par défaut) puisse être paramétré globalement, pour tous les légumes.
Ainsi, par exemple, sur la base du code précédent, on souhaite voir la différence de poids d'épluchure d'une soupe entre le paramétrage par défaut (10%) et un paramétrage à 5%...

Modifiez les classes nécessaires pour réaliser ces fonctionnalités et pouvoir exécuter le code suivant :

```
Soup s = Soup();
s.add(new Potatoe(0.3, 10));
s.add(new Carrot(0.25, 30));
cout << s.getPeelingWeight() << endl;
Vegetable::setPeelingWeight(0.05);
cout << s.getPeelingWeight() << endl;
```

PARTIE III : On surveille ses calories

La plupart des aliments, qu'ils soient crus, cuits, nature ou transformés, contiennent des calories.
On souhaite maintenant disposer d'une méthode indiquant le nombre de calories d'un aliment donné.

On suppose que:

- les carottes contiennent 40 calories pour 100 grammes;
- les pommes de terre contiennent 80 calories pour 100, moins 5 calories par germe sur la patate;
- le calcul du nombre de calories se fait sur chaque légume sans tenir compte du poids des épluchures, et produit un nombre entier (pas un nombre flottant);
- on ne peut pas définir le nombre de calories d'un légume sans savoir précisément de quel légume il s'agit;
- une soupe contient autant de calories que les légumes qui entrent dans sa composition.

Modifiez les classes nécessaires pour réaliser ces fonctionnalités et pouvoir exécuter le code suivant (attention: toutes les fonctionnalités décrites dans les deux exercices précédents doivent être maintenues) :

```
Potatoe p = Potatoe(0.3, 10);
cout << p.getCalories() << endl;
Carrot c = Carrot(0.25, 30);
cout << c.getCalories() << endl;
Soup s = Soup();
s.add(&p);
s.add(&c);
cout << s.getCalories() << endl;
```