Seul document autorisé : une feuille A4 recto-verso manuscrite (non photocopiée). Aucune machine.
Le barème est indicatif. Une question peut toujours être traitée en utilisant les précédentes (traitées ou non).
Les morceaux de code Java devront être clairement présentés, indentés et commentés.
L'utilisation de collections de l'API Java (en particulier `LinkedList` et `ArrayList`) est interdite, sauf mention explicite contraire.

- Tous les attributs d'instance de toutes les classes introduites doivent être privés.
- Il est possible d'écrire des méthodes non spécifiquement demandées si c'est plus pratique, en spécifiant pour chaque méthode dans quelle classe elle se trouve.

Le gérant du "gland de chêne" étant satisfait de certaines solutions proposées pour la gestion informatisée des commandes et des préparations de sandwichs, il voudrait maintenant améliorer la comptabilité de son fast-food. Le but de ce sujet est de pouvoir calculer automatiquement le prix et le bénéfice associés à un sandwich.

Ingrédient

Un ingrédient est représenté par son nom, son prix d'achat (par le gland de chêne à un grossiste) et son prix de revente (par le gland de chêne à ses clients) ; ces deux prix sont représentés par des entiers qui expriment le prix d'une portion en centimes. Aucun ingrédient ne peut disparaître de la carte : une fois un ingrédient créé, il pourra toujours être utilisé. Le *bénéfice* associé à un ingrédient est la différence entre son prix de revente et son prix d'achat.

La classe `Ingredient` contient donc au minimum un attribut de type chaîne de caractères et deux attributs entiers représentant le prix d'achat et le prix de revente.

Cette classe possède en outre deux constructeurs :

- l'un prend en argument le nom d'un ingrédient, son prix d'achat et son prix de revente ;
- l'autre prend en argument le nom de l'ingrédient et son prix d'achat ; le prix de revente est alors calculé en multipliant le prix d'achat par un facteur qui ne dépend pas de l'ingrédient.

Son interface est la suivante (le caractère statique ou non des méthodes n'est pas spécifié, ce sera à vous de le donner, en justifiant votre choix) :

```
/** renvoie le bénéfice associé à un ingrédient */
int benefice();

/** renvoie l'ingrédient dont le bénéfice est maximal */
Ingredient quiRapporteLePlus();

/** fixe à f le facteur multiplicatif qui est utilisé par le deuxième constructeur */
void facteurDeRevente(int f);
```

Exercice 1. (3 points) Écrire la classe `Ingredient` (attributs, constructeurs et méthodes). N'oubliez pas d'expliquer ce qui doit être statique.

Recettes

Une recette est une suite ordonnée d'ingrédients. On peut voir apparaître plusieurs fois le même ingrédient dans une recette (ex : dans un double cheeseburger, il y a deux steaks hachés séparés par une tranche de fromage). Pour simplifier, on supposera qu'une apparition d'un ingrédient dans une recette correspond à une unité de cet ingrédient.

L'ensemble des recettes est représenté dans un arbre binaire. L'idée étant qu'en suivant un chemin depuis la racine de l'arbre jusqu'à un nœud reconnu comme étant un sandwich (mais qui n'est pas nécessairement une feuille), on a la liste des ingrédients *dans l'ordre* pour ce sandwich. Chaque nœud représente une option sur un ingrédient : s'il est intégré au sandwich la recette se poursuit en explorant le fils gauche ; et s'il n'est pas intégré, une autre option d'ingrédient est envisagée : celle du fils droit.

Pour celà, on crée les classes suivantes :

- **`Recette`** : référence la racine de l'arbre ;
- **`Creation`** : référence un nœud de l'arbre et contient :
  - une référence vers un ingrédient (éventuellement nulle),
  - deux références vers des nœuds fils,
  - une référence vers le nœud père dans l'arbre (nulle dans le cas de la racine de l'arbre),
  - une référence vers un sandwich, non nulle si et seulement si le nœud correspond à un sandwich ;
- **`Sandwich`** : référence un sandwich et contient :
  - un attribut de type chaîne de caractères contenant le nom du sandwich,
  - une référence vers le nœud de l'arbre qui correspond à ce sandwich (c'est-à-dire celui qui contient l'instance courante de **`Sandwich`** comme attribut).

Le *prix* (resp. *bénéfice*) associé à un sandwich est la somme des prix de revente (resp. des bénéfices) associés à ses ingrédients (en tenant compte des multiplicités d'apparition).

Exercice 2. (1 point) Écrire les déclarations des attributs des classes **`Recette`**, **`Creation`** et **`Sandwich`**.

Exercice 3. (1 point) Dessiner une instance de **`Recette`** qui contient les recettes des trois sandwichs suivants (les ingrédients sont notés dans l'ordre) :

- le *matelot* : beurre, saumon ;
- le *parigot* : beurre, jambon ;
- le *banlieusard* : beurre, jambon, cornichon ;
- le *grassouillet* : saumon, mayonnaise.

L'interface de la classe **`Sandwich`** est la suivante :

```
/** calcule le prix d'un sandwich en centimes */
int prix();
```

L'interface de la classe **`Recette`** est la suivante :

```
/** renvoie le sandwich associé au bénéfice maximal */
Sandwich quiRapporteLePlus();

/** affiche la liste des sandwichs dont le prix est inférieur ou égal  à l'argument */
void jePeuxLAcheter(int sous);

/** supprime les sandwichs peu bénéfiques */
void supprime(int tropPeu);
```

Exercice 4. (2 points) Écrire la méthode **`prix`** de la classe **`Sandwich`**.

Exercice 5. (3 points) Écrire la méthode **`quiRapporteLePlus`** de la classe **`Recette`**.

Exercice 6. (3 points) Écrire la méthode **`jePeuxLAcheter`** de la classe **`Recette`**.

Exercice 7. (5 points) Le but de la méthode **`supprime`** de la classe **`Recette`** est de supprimer les sandwichs dont le bénéfice est strictement inférieur à l'argument. En écrire deux versions : dans la première on se contentera de mettre à **`null`** les références vers les sandwichs à supprimer, dans la deuxième on supprimer les nœuds de l'arbre devenus inutiles (par l'absence de présence d'un sandwich dans leur descendance). Pour simplifier cette deuxième version, on supposera (sans avoir à le vérifier) qu'au moment de son invocation l'arbre ne contient pas de nœud inutile. Ces deux versions peuvent-elles porter le même nom ? (Justifiez la réponse.)

Le gérant du gland de chêne décide de mettre en place les sandwichs à la demande : un tel sandwich est fabriqué à partir d'une liste d'ingrédients fournis par le client. Son prix est soit le prix du sandwich s'il s'avère exister dans la liste des recettes proposées par le fast food, soit la somme des prix de revente de ses ingrédients plus 1€ (pour rappel 1€ vaut 100 centimes).

La méthode suivante est ajoutée à l'interface de la classe **`Recette`** :

```
/** calcule le prix d'un sandwich à la demande */
int prixSandwichLibre(java.util.ArrayList<Ingredient> liste);
```

Exercice 8. (2 points) Écrire la méthode **`prixSandwichLibre`** de la classe **`Recette`**.