INF1010

**Programmation Orientée-Objet**

**Hiver 2019**

Travail pratique #3
Héritage

---

| | |
|---|---|
| **Objectifs :** | Permettre à l'étudiant de se familiariser avec l'héritage et la conversion de type. |
| **Remise du travail :** | Lundi 25 février 2019, 8h (AM) |
| **Références :** | Notes de cours sur Moodle & Chapitre 14 du livre Big C++ 2e éd. |
| **Documents à remettre :** | Tous les fichiers .cpp et .h exclusivement complétés et réunis sous la forme d'une archive au format .zip. |
| **Directives :** | Directives de remise des Travaux pratiques sur Moodle<br>Les en-têtes (fichiers, fonctions) et les commentaires sont obligatoires.<br>Les travaux dirigés s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe. **Pas de remise possible sans être dans un groupe.**<br>Veuillez suivre le guide de codage |

## Travail à réaliser

Le travail consiste à continuer l'application PolyFood commencée aux TPs précédents en y intégrant les notions d'héritage et de conversion de type.
Afin de diversifier leurs offres, le restaurant PolyFood ajoute à son menu différents types de plats avec l'introduction de plat biologique et la possibilité de personnalisé un certain nombre d'ingrédients dans un plat. Un programme de fidélité est également créé pour les clients : ils sont donc répartis en trois catégories qui donnent accès à différents avantages : **les clients occasionnels, les clients réguliers et les clients prestiges.** Les clients seront désormais représentés par des objets plutôt que par des nombres. Un programme de livraison est aussi en cours de test dans le restaurant et accessible une catégorie restreinte de client.
Les nouveaux types de plats vont donc dériver de la classe Plat qui existe déjà, tout comme les nouvelles catégories de clients qui dériveront de la classe Client.
L'héritage permet de créer une classe de base possédant ses propres caractéristiques et de l'utiliser pour créer de nouvelles classes. Les nouvelles (dérivées) classes ont avec la classe de base, la relation « est un » et partagent avec la classe de base certaines caractéristiques. Pour réussir ce TP il faudra vous familiariser avec la fonction C++ **static_cast<T>(Objet).** Cet opérateur permet de convertir à la compilation un objet d'une classe en objet d'une autre classe. Dans ce TP, il vous permettra par exemple de convertir objet de type **Plat** en objet de type **PlatBio**.
**La structure du fichier polyfood.txt a changé, et les méthodes de lecture ont déjà été modifiées en conséquence.**
Pour vous aider, les fichiers corrigés du TP précédent vous sont fournis. Vous n'avez qu'à implémenter les nouvelles méthodes décrites plus bas et les méthodes à modifier vous ont été indiquées.
**ATTENTION :**

- Pensez à modifier les méthodes d'affichage déjà écrites pour qu'elles s'adaptent aux nouvelles classes.
- Tout au long du TP, assurez-vous d'utiliser les opérateurs sur les objets et non sur leurs pointeurs ! Vous devez donc déréférencer les pointeurs si nécessaires.
- Vous serez pénalisés pour les utilisations inutiles du mot-clé this. Utilisez-le seulement où nécessaire.
- Il faut utiliser les fichiers fournis, plutôt que de continuer avec vos fichiers du TP2.
- Écrire les destructeurs lorsque c'est nécessaire.

**Remarque : Pour plus de précision sur le travail à faire et les changements à effectuer, veuillez-vous référer aux fichiers .h et à l'énoncé.**

## Classe Plat

Elle contient le nouvel attribut type_ qui représente le type du plat. Il n'y a aucune modification à faire dans cette classe.

## Classe PlatBio

Il s'agit d'un plat biologique qui est ajouté au menu du restaurant et qui dérive de la classe de base Plat. En plus de son prix, le restaurant perçoit un montant supplémentaire qui représente un pourcentage (taux) sur son prix.

**Elle contient l'attribut privé suivant** :

- ecotaxe_ : taux qui s'ajoute au prix.

**Les méthodes suivantes doivent être implémentées:**

- Les méthodes d'accès et de modification de l'écotaxe.
- Le constructeur par paramètres initialise les attributs aux valeurs correspondantes
- L'opérateur qui affiche les caractéristiques du plat (voir l'affichage à la fin de l'énoncé).

## Classe PlatCustom

Cette classe dérive de la classe Plat. Il s'agit d'un plat auquel le client a choisi d'ajouter des ingrédients supplémentaires. Il ne figure pas au menu du restaurant et ne sera ajouté que dans la commande de la table concernée. En plus du prix du plat, on ajoute au plat, un supplément égal au nombre d'ingrédients multiplié par la constante **FRAIS_CUSTOMISATION**.

**Elle contient les attributs privés suivants** :

- supplement_ : supplément à ajouter au prix du plat.
- nbIngredients_ : nombre d'ingrédients supplémentaires.

**Les méthodes suivantes doivent être implémentées:**

- Le constructeur par paramètres initialise les attributs aux valeurs correspondantes et calcule le supplément.
- Les méthodes d'accès et de modification des attributs dont la signature est dans le fichier PlatCustom.h
- La méthode calculerSupplement permet de calculer le montant du supplément.
- L'opérateur << qui affiche les caractéristiques du plat (voir l'affichage à la fin de l'énoncé).

## Classe Client

Elle représente les clients occasionnels du restaurant qui ne participent pas au programme de fidélité du restaurant. Ils ne bénéficient pas des livraisons et des réductions.

**Elle contient les attributs privés suivants** :

- nom_ : nom du client.
- prénom_ : prénom du client.
- tailleGroupe_ : le nombre de personnes qui accompagnent le client au restaurant.
- statut_ : le statut du client dans le programme de fidélité ( selon le type énuméré).

**Les méthodes suivantes doivent être implémentées:**

- Le constructeur par paramètres initialise les attributs aux valeurs correspondantes :
- Le constructeur par défaut initialise les attributs aux valeurs par défaut (nom et prénom à inconnu et statut à occasionnel).
- Les méthodes d'accès et de modification des attributs dont la signature est dans le fichier Client.h
- La méthode ConvertirStatutString() permet d'obtenir la chaine de caractères correspondant à la valeur énumérée de statut_
- L'opérateur << qui affiche les caractéristiques du client (voir l'affichage à la fin de l'énoncé).

## Classe ClientRegulier

Elle représente les clients réguliers du restaurant qui participent au programme de fidélité et dérive de la classe Client. Les clients ne sont pas admissibles aux livraisons, mais bénéficient d'un taux de réduction TAUX_REDUC_REG à partir d'un certain nombre de points.

**Elle contient l'attribut protégé suivant** :

- nbPoints_ : nombre de points de fidélité.

**Les méthodes suivantes doivent être implémentées:**

- Le constructeur par paramètres initialise les attributs aux valeurs correspondantes.
- Le constructeur par défaut initialise les attributs aux valeurs par défaut. Le nombre de points est nul.
- Les méthodes d'accès et de modification des attributs dont la signature est dans le fichier ClientRegulier.h.
- La méthode augmenterNbPoints() prend en paramètre le nombre points à ajouter au solde actuel de points.
- L'opérateur << qui affiche leurs caractéristiques (voir l'affichage à la fin de l'énoncé).

## Classe ClientPrestige

Elle représente les clients les plus assidus du restaurant qui participent au programme de fidélité et dérive de la classe ClientRegulier. Les clients sont admissibles aux livraisons et bénéficient d'un taux de réduction TAUX_REDUC_PRESTIGE. Les plus assidus parmi eux, dont le solde de point dépasse le seuil de livraison gratuite ne paient pas les livraisons.

**Elle contient l'attribut privé suivant** :

- addresse_ : la valeur du type énuméré AddressCode et réprésente la zone d'habitation du client

**Les méthodes suivantes doivent être implémentées:**

- Le constructeur par paramètres initialise les attributs aux valeurs correspondantes.
- Le constructeur par défaut initialise les attributs aux valeurs par défaut. Son adresse par défaut est la zone3.
- Les méthodes d'accès et de modification des attributs dont les signatures sont dans le fichier ClientPrestige.h
- La méthode getAddressCodeString() permet d'obtenir la chaine de caractères correspondant à la valeur énumérée de addresse_.
- L'opérateur << qui affiche les caractéristiques du client (voir l'affichage à la fin de l'énoncé).

## Classe Table

Cette classe caractérise une table du restaurant, elle a un identifiant, une liste de plats (la commande de la table), et un nombre de places. Une nouvelle table virtuelle a été ajouté au fichier texte pour effectuer les commandes de livraison et pouvoir garder un traitement similaire celui des tables physiques. Son index est désigné par INDEX_TABLE_LIVRAISON dans le fichier Restaurant.h.

**Elle contient les nouveaux attributs privés suivants** :

- clientPrincipal_ : un pointeur sur la classe client qui représente le client principal de cette table. C'est ce client qui passe les commandes et son statut sera pris en compte pour appliquer les réductions.

**Les méthodes suivantes doivent être modifiées:**

- La méthode getChiffreAffaire() qui doit désormais prendre en compte le type de plat dans la commande dans le calcul du chiffre d'affaire selon les modalités indiquées plus haut.
- L'opérateur << qui affiche les caractéristiques de la table (voir l'affichage à la fin de l'énoncé).

**Les méthodes suivantes doivent être implémentées:**

- Les méthodes d'accès et de modification des attributs dont les signatures est dans le fichier Table.h

## Classe Menu

Cette classe caractérise un menu, ainsi il contient un tableau dynamique contenant des pointeurs vers des plats ainsi qu'un attribut issu d'un type énuméré (fourni dans le fichier header) permettant de déterminer le type de menu (Matin, Midi ou Soir).

- Son attribut listePlat_ devra désormais contenir des objets Plat ou PlatBio en fonction du fichier polyfood.txt.

**Les méthodes suivantes doivent être modifiées :**

- L'opérateur << qui affiche les caractéristiques du menu (voir l'affichage à la fin de l'énoncé).
- Un constructeur par copie.
- L'opérateur = qui écrase les attributs du menu par les attributs du menu passé en paramètre et qui renvoie ensuite une référence au menu.

**Les méthodes suivantes doivent être implémentées :**

- L'opérateur += qui prend en paramètre un objet de la classe PlatBio.

## Classe Restaurant

Cette classe caractérise le restaurant ayant des tables et des clients qui commandent des plats.

**Elle contient le nouvel attribut privé suivant :**

- fraisTransport_ qui contient le montant des différents frais de livraison pour les trois zones de livraisons existantes. Sa lecture a déjà été effectué par la méthode lireAdresses().

**Les méthodes suivantes doivent être implémentées :**

- livrerClient() qui prend en paramètres un client et sa liste de plats. La méthode vérifie si le client est admissible à la livraison et si tel est le cas, on effectue la livraison. Le client prestige a droit à une livraison. On suppose que la table pour la livraison correspond à l'indice INDEX_TABLE_LIVRAISON du tableau tables_. On place un seul client à cette table. On fait appel à la méthode commanderPlat() sur la liste de plats reçu en paramètre. La livraison se fait de manière similaire au placement des clients sur place à la différence que seul le client concerné y est placé et il y a des frais de livraison qui s'applique selon sa zone d'habitation et son statut.
- La libération de cette table de livraison doit être systématique (appel à la méthode libérerTable())_
- calculerReduction() qui vérifie si le client est admissible aux réductions.
    - **Client régulier**, si le nombre de points accumulés dépasse la constante SEUIL_DEBUT_REDUCTION, alors la réduction calculée est égale à - montant x TAUX_REDUC_REGULIER.
    - **Client Prestige**, la réduction calculée est égale à - montant x TAUX_REDUC_PRESTIGE et si le nombre de points accumulés est plus petit que la constante SEUIL_LIVRAISON_GRATUITE, alors on ajoute à la réduction, la livraison des frais de transport selon la zone.

**Les méthodes suivantes doivent être modifiées :**

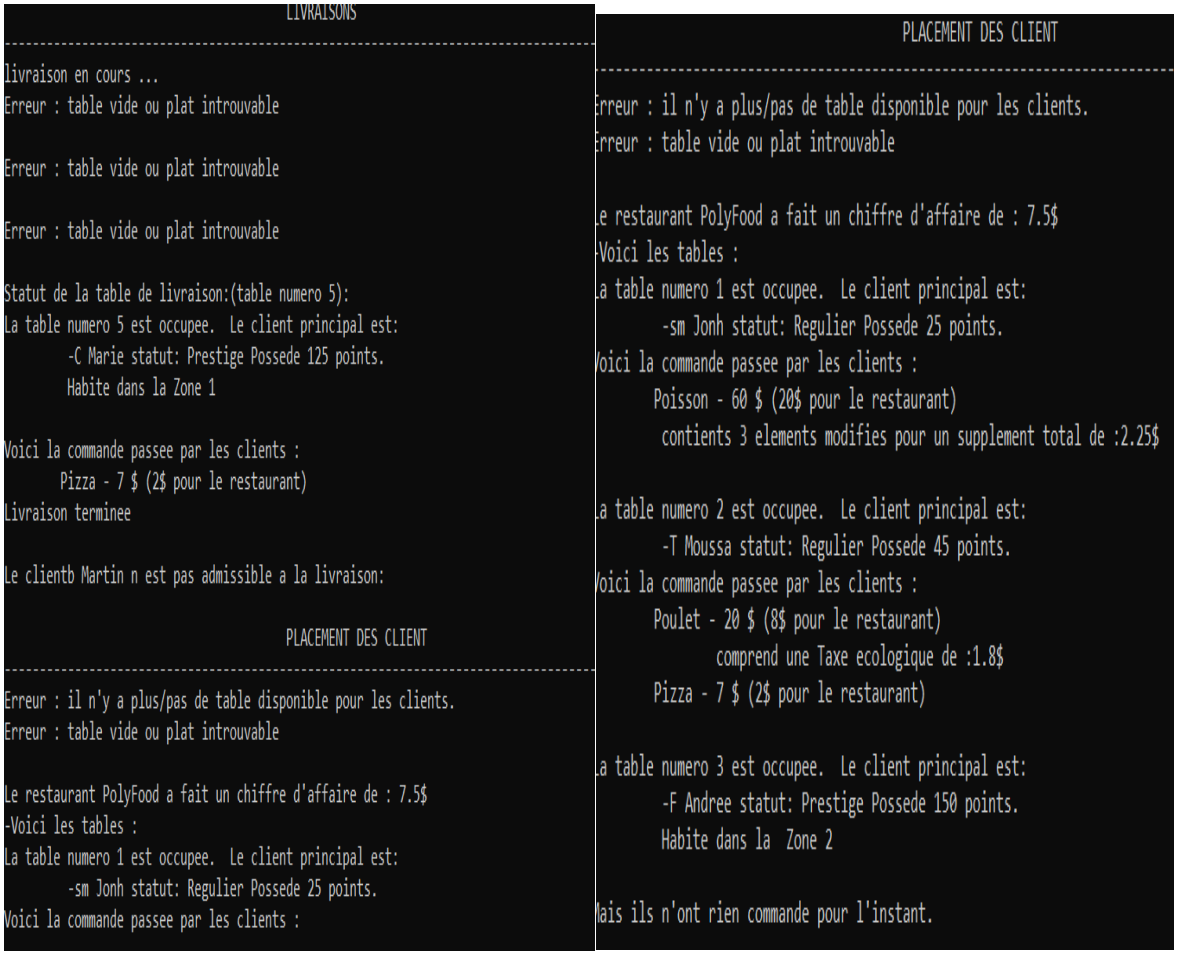
- L'opérateur << (remplace la méthode d'affichage), qui affiche les caractéristiques du restaurant.
- libererTable() qui prend en compte les différents avantages (réduction) du client principal.
- placerClients() qui s'adapte à la nouvelle structure du client ( Indice : la taille du groupe).
- commanderPlat() : qui s'adapte à l'existence des plat customisés. Elle prend comme paramètre facultatif le type de plat et la quantité d'ingrédients à modifier dans le cas des plats personnalisés.

## Main.cpp

Des directives vous sont fournies dans le fichier main.cpp et il vous est demandé de les suivre.

Votre affichage devrait avoir une apparence semblable à celle ci-dessous.

### 1er affichage PolyFood :



```
ais ils n'ont rien commande pour l'instant.

a table numero 4 est occupee.  Le client principal est:
        -b Martin statut: OccasionnelVoici la commande passee par les c
       Poulet - 20 $ (8$ pour le restaurant)
              comprend une Taxe ecologique de :1.8$
       Muffin - 5 $ (2$ pour le restaurant)

a table numero 5 est vide.


Voici son menu :
atin :
       Soupe - 100 $ (50$ pour le restaurant)
              comprend une Taxe ecologique de :1.5$
       Oeuf - 12 $ (4.5$ pour le restaurant)
              comprend une Taxe ecologique de :4.5$
       Pain - 5 $ (2$ pour le restaurant)
       Crepes - 6 $ (2$ pour le restaurant)
       Pancakes - 7 $ (2$ pour le restaurant)

e plat le moins cher est : Pain - 5 $ (2$ pour le restaurant)

idi :
       Poulet - 20 $ (6$ pour le restaurant)
       Frites - 5 $ (1$ pour le restaurant)
       Burrito - 8 $ (2$ pour le restaurant)
              comprend une Taxe ecologique de :2$
       Quesadillas - 9 $ (4$ pour le restaurant)
       Ratatouille - 8 $ (5$ pour le restaurant)
              comprend une Taxe ecologique de :2$

e plat le moins cher est : Frites - 5 $ (1$ pour le restaurant)

oir :
       Pates - 30 $ (9$ pour le restaurant)
       Poisson - 60 $ (20$ pour le restaurant)
```

```
Le restaurant PolyFood a fait un chiffre d'affaire de : 103.12$
-Voici les tables :
La table numero 1 est vide.

La table numero 2 est vide.

La table numero 3 est vide.

La table numero 4 est vide.

La table numero 5 est vide.


-Voici son menu :
Matin :
       Soupe - 100 $ (50$ pour le restaurant)
              comprend une Taxe ecologique de :1.5$
       Oeuf - 12 $ (4.5$ pour le restaurant)
              comprend une Taxe ecologique de :4.5$
       Pain - 5 $ (2$ pour le restaurant)
       Crepes - 6 $ (2$ pour le restaurant)
       Pancakes - 7 $ (2$ pour le restaurant)

Le plat le moins cher est : Pain - 5 $ (2$ pour le restaurant)

Midi :
       Poulet - 20 $ (6$ pour le restaurant)
       Frites - 5 $ (1$ pour le restaurant)
```

```
Midi :
       Poulet - 20 $ (6$ pour le restaurant)
       Frites - 5 $ (1$ pour le restaurant)
       Burrito - 8 $ (2$ pour le restaurant)
              comprend une Taxe ecologique de :2$
       Quesadillas - 9 $ (4$ pour le restaurant)
       Ratatouille - 8 $ (5$ pour le restaurant)
              comprend une Taxe ecologique de :2$

Le plat le moins cher est : Frites - 5 $ (1$ pour le restaurant)

Soir :
       Pates - 30 $ (9$ pour le restaurant)
       Poisson - 60 $ (20$ pour le restaurant)
              comprend une Taxe ecologique de :20$
       Poulet - 20 $ (8$ pour le restaurant)
              comprend une Taxe ecologique de :1.8$
       Muffin - 5 $ (2$ pour le restaurant)
       Pizza - 7 $ (2$ pour le restaurant)

Le plat le moins cher est : Muffin - 5 $ (2$ pour le restaurant)
```

## Spécifications générales

- Ajouter un destructeur pour chaque classe chaque fois que cela vous semble pertinent.
- Utilisez la liste d'initialisation pour l'implémentation de vos constructeurs.
- Ajouter le mot-clé *const* chaque fois que cela est pertinent.
- Appliquez un affichage « user friendly » (ergonomique et joli) pour le rendu final.
- Documenter votre code source.
- **Bien lire le barème de correction ci-dessous.**

## Questions

*Répondez aux questions au début du main.*

1. Pourquoi à t'on besoin de l'attribut type_ dans la classe Plat? Que ce serait-il passé s'il n'existait ?

2.Quelle est l'importance du static_cast dans la classe Client et ses classes dérivées?

3.(bonus)Pourquoi est-il intéressant de dériver ClientPrestige de ClientRégulier? (1pt)

## Correction

La correction du TP1 se fera sur 20 points.

Voici les détails de la correction :

- (3 points) Compilation du programme
- (3 points) Exécution du programme
- (4 points) Comportement exact des méthodes du programme
- (3 points) Utilisation adéquate de l'héritage
- (2 points) Documentation du code. Qualité du code
- (3 points) Utilisation adéquate de static_cast
- (2 points) Réponse aux questions.