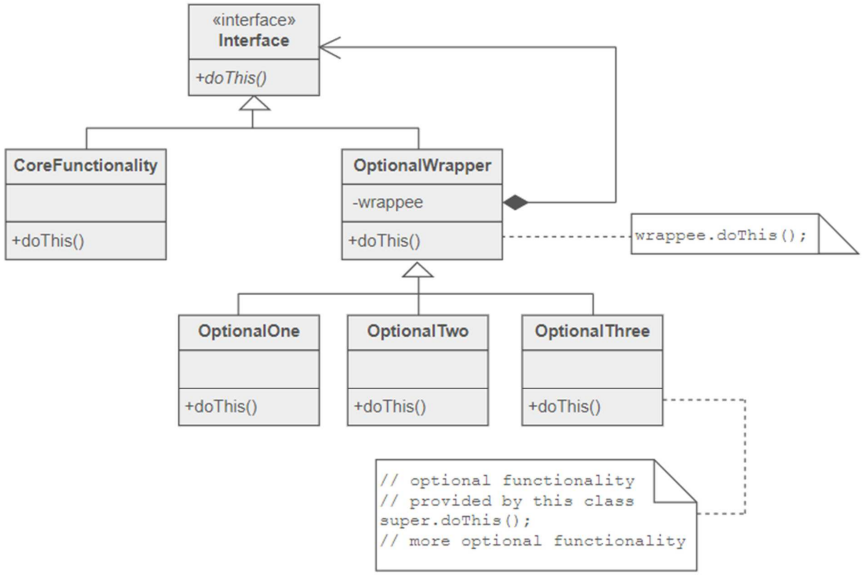
M3105 - Kata Kebab à notre sauce
Décorons les kébabs !

Il vous est conseillé de faire ce TD en pair-programming 🤣

Rappelons le contexte de notre client :
« Bonjour, je suis un vendeur de Kebab.
J'ai besoin de fabriquer des kebabs de toutes sortes.
Les ingrédients sont variés : Laitue, roquette, tomate, oignons, agneau, bœuf, cheddar, etc. »

Après cette première approche de compréhension des besoins du client (un vendeur de kébab), nous avons fait appel à un architecte pour vous aider à refactorer votre code et vous guider dans votre conception.
Ce dernier vous conseille d'implémenter ce projet en utilisant un **pattern Decorator**, c'est ce que vous allez donc faire maintenant 😉

1. Commencez par consulter un peu de documentation sur le pattern **Decorator**.
   Le diagramme de classes du pattern **Decorator** est le suivant :
   (extrait de sourcemaking.com : https://sourcemaking.com/design_patterns/decorator)



   Rendez-vous ensuite dans le dépôt https://github.com/iblasquez/enseignement-iut-m3105-conception-avancee où vous retrouverez les travaux de vos collègues, ainsi que d'autres liens dans references_patterns.md)

Pour faciliter l'implémentation du ***pattern Decorator****,*
nous nous limiterons pour ce TD,
uniquement aux régimes végétariens et pescetariens*,*
Ainsi un Kebab doit pouvoir seulement dire ***s'il est vegetarien ou s'il est pescetarien***
(nous ne traiterons pas pour commencer le fait de pouvoir doubler le fromage ou enlever les oignons : petits pas par petits pas…)

2. ***En mode déconnecté*** (sur *une feuille de papier* 😉), commencez par proposer une ébauche du diagramme de classes de votre application en mettant en œuvre le pattern **Decorator** dans votre contexte de kebab (limité pour le moment aux fonctionnalités `isVegetarien` et `isPescetarien`) .

   Il est bien sûr important de remarquer qu'un kébab n'est rien d'autre qu'une assiette avec des ingrédients dedans !!!

3. Sous votre IDE préféré, créez un nouveau projet **kebabdecorator** dans lequel vous allez implémenter le **kata Kebab** à l'aide du pattern décorateur en vous appuyant sur le diagramme de classes que vous venez de modéliser.

   Quelques remarques :
   - Le composant (*Interface*), l'élément « feuille» (*Core Functionality*) et les décorateurs (*OptionalWraper*) devront être des kébabs.
     Les *décorateurs*, étant en plus des *composites* de kebab, devront proposer un constructeur qui prendra au minimum un kébab en paramètre…
   - Pour traiter un régime, par exemple savoir si le kebab est végétarien ou non, il faudra :
     - que l'ingrédient renvoie `false` s'il ne respecte pas le régime végétarien
     - que l'ingrédient renvoie le `isVegetarien` du kebab interne s'il respecte le régime végétarien

   - D'après Wikipédia :
     ➔ Le **végétarisme** *est une pratique alimentaire qui exclut la consommation de chair animale.*
     ➔ *Le* **pescétarisme***, ou pesco-végétarisme, est un néologisme désignant le régime alimentaire d'une personne omnivore qui s'abstient de consommer de la chair animale à l'exception de celle issue des poissons, des crustacés et mollusques aquatiques*
     Remarque *:* **Pescetarien** = végétarien + poissons + **crevettes**

4. Vérifiez et validez le code que vous écrivez à l'aide de tests unitaires automatisés !

5. Utilisez votre IDE pour générer automatiquement un diagramme de classes à partir du code que vous venez d'écrire. Vérifiez que ce diagramme de classes est bien cohérent avec le diagramme de classes du pattern Decorator de la question 1.

6. Si le temps vous le permet, vous pouvez ajouter la fonctionnalité `doublerLeFromage` (toujours en utilisant le pattern décorateur). Vérifier et valider cette fonctionnalité à l'aide de tests !

Remarque : Nous ne traiterons pas dans le cadre de cet exercice enlever les oignons !!!
(sauf si vous êtes vraiment très rapide 😉)