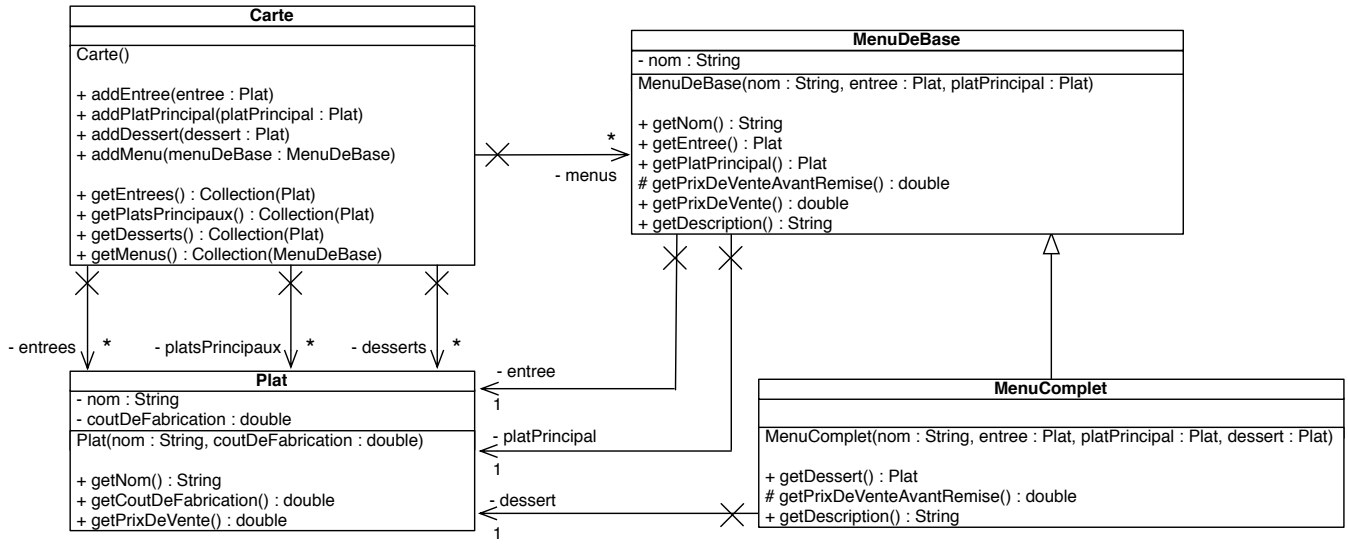
Examen R2.03

Rappel UML : – signifie visibilité privée, + signifie visibilité publique, # signifie visibilité protégée, *italique*$_i$ (un petit i a été ajouté pour lever toute ambiguïté) signifie abstrait ou interface (dans le cas d'une interface cela sera précisé clairement), ⟶▷ héritage, ⟶ association.

Exercice 1 : Héritage, Composition, Documentation, Test unitaire

Un restaurant souhaite dématérialiser sa carte (menus, entrées, plats principaux et desserts). Dans un premier temps, seulement pour la création et l'affichage de leur carte. Pour ce faire, le restaurant vous demande de créer les classes correspondantes au diagramme UML ci-dessous tout en respectant les informations et contraintes ci-après :



Aide UML : `Collection(Plat)` désigne une collection d'objets de type `Plat`. A vous de déterminer la classe Java la plus adaptée à votre besoin. De plus, **seules les visibilités public et protected sont montrées dans ce diagramme.** Vous pouvez ajouter des méthodes private si vous le souhaitez.

Les classes fournies :

- La classe `Plat`.
- La classe `PlatException` utilisée par la classe `Plat` pour lever des exceptions.
- La classe `TestCarte` **(NE PAS MODIFIER)**, elle contient un jeu de test à lancer. La trace attendue de ce test se trouve dans le fichier `TestCarte-trace.txt`. Pour le moment, le jeu d'instructions est commenté.
- La classe `CarteUtilitaire` contenant une méthode d'arrondi et une méthode `capitalize(…)` qui transforme la chaîne passée en paramètre en mettant la première lettre en majuscule et le reste en minuscule.

Informations complétant le diagramme UML :

Une carte est constituée d'entrées, de plats principaux, de desserts et également de menus qu'ils soient « de base » (seulement entrée et plat principal) ou « complet » (entrée, plat principal et dessert).

La méthode `getPrixDeVente()` de la classe `Plat` retourne le prix du plat basé sur le coût de fabrication : prix de vente = coût de fabrication × 1,5. Ce prix est arrondi en utilisant la méthode `arrondi(double valeur)` contenue dans la classe `CarteUtilitaire`.

La méthode `getPrixDeVenteAvantRemise()` de la classe `MenuDeBase` retourne le prix du menu basé sur la somme du prix de vente des plats le constituant.

La méthode `getPrixDeVente()` de la classe `MenuDeBase` retourne le prix du menu basé sur le prix de vente avant remise auquel on applique une remise de 20% : prix de vente = prix de vente avant remise × 0,8. Ce prix sera arrondi en utilisant la méthode `arrondi(double valeur)` contenue dans la classe `CarteUtilitaire`.

La méthode `getDescription()` de la classe `MenuDeBase` retourne une chaîne de caractères représentant le menu : son nom et les noms des plats le constituant (Vous pouvez utiliser les codes nouvelle ligne « `\n` » et tabulation « `\t` » dans une chaîne de caractères).

La méthode `printCarte()` de la classe `CarteUtilitaire` permet d'afficher la carte sur le terminal. Cette méthode est utilisée dans la classe `TestCarte`.

Quelques contraintes supplémentaires sont à prendre en compte pour cette spécification :

- Un nom de plat ou de menu commence par une majuscule et le reste est en minuscule.
- Un nom de plat ou de menu ne peut pas être nulle ou vide (levée d'exception).
- Le coût de fabrication d'un plat doit être compris entre 1 et 20 euros compris (levée d'exception sinon). La classe `Plat` possède deux constantes `COUT_DE_FABRICATION_MIN` et `COUT_DE_FABRICATION_MAX` correspondant à ces seuils.

Exercice 1.1 : réalisation et javadoc

En considérant les explications et le diagramme de classes précédents, **compléter** le code des classes `Carte`, `MenuDeBase`, `MenuComplet` et `CarteUtilitaire`.

Générer la documentation technique (javadoc) pour les méthodes suivantes :

- `addMenu(…)` pour la classe `Carte`.
- le constructeur `MenuDeBase(…)` pour la classe `MenuDeBase`.
- `getPrixDeVente()` pour la classe `MenuDeBase`.

Exercice 1.2 : tests unitaires

La classe `Plat` vous est fournie, mais vous devez réaliser les tests unitaires pour la vérifier. Suivant les contraintes données plus haut et les informations complétant le diagramme UML, **compléter** la classe `PlatTest` dans le dossier `test/exercice1` pour tester les méthodes `getNom()`, `getCoutDeFabrication()` et `getPrixDeVente()`.

Attention, une erreur s'est glissée dans une des classes fournies. En fonction de vos tests, merci de faire les modifications nécessaires pour corriger le problème. **Ajouter** un commentaire à l'endroit où vous modifierez le code.