épreuve d'informatique | **module Java** | **Sorbonne Université**
Durée 2 heures | **Licence 3** | **Année 2019–2020**

Examen - Programmation orientée objets - Java

Tout documents autorisés - 2 heures

Exercice 1 : Coupe du monde de Quidditch

C'est la coupe du monde de Quidditch et vous êtes chargé d'organiser et gérer les équipes et les rencontres. Il est à noter que les règles et la composition d'une équipe ont été très simplifiées pour les besoins du problème.

Q 1.1 Créez la classe `Joueur` sachant qu'un joueur a un prénom, un numéro (le premier joueur a le numéro 1, second 2, etc) et un attribut `boolean` pour dire si le Joueur a le souafle (ballon). `Ecrivez` un constructeur à un paramètre pour initialiser le prénom, le ballon est initialise à false. Ajoutez un accesseur et modificateur pour l'attribut `boolean`.

Q 1.2 Créez la classe `Poursuiveur` et `Batteur` qui hérite tous les attributs et méthodes de la classe `Joueur`. Les deux classes ont la méthode `monAction` qui vérifie si le Poursuiveur/Batteur a le ballon et affiche : *[prenom] Poursuiveur, je transmets le ballon* ou *[prenom] Batteur, je transmets le ballon.*

Q 1.3 Créez la classe `Equipe` qui a un tableau de `Joueur` de taille max 7. Pour contrôler le nombre de joueur dans l'équipe ajoutez un attribut qui est incrémenté quand un nouveau joueur est ajouté à l'équipe ;

Q 1.4 Ajoutez une méthode pour ajouter un nouveau joueur.

Q 1.5 Ecrire une méthode permettant de générer un nom de manière aléatoire.

Q 1.6 Nous voulons ajouter maintenant le fait que les équipes vont appartenir à différentes écoles. Chaque école possède des caractéristiques. Comment faire ?

Q 1.7 Ecrire les classes d'écoles et indiquer les modifications apportées aux autres classes.

Q 1.7 Créez la classe Match avec deux équipes. A chaque équipe ajoutez quatre Poursuiveur et trois Batteur, dont le prénom est généré de manière aléatoire. Donner le ballon au premier joueur de la première équipe. Pendant 50 itérations vous devez choisir au hasard un joueur d'une équipe (50% de chances pour chaque équipe) et passer le ballon pour simuler un vrai match.

Exercice 2 : Gestionnaire de Cloud Kebab (12 pts)

Un maitre kébabier audacieux décide de créer le concept du *Cloud Kebab*. Il s'agit d'un magasin qui vend des kebab de manière participative, sur le principe d'un kickstarter d'ingrédients (plateforme de financement collaborative).

Nous disposons d'une classe abstraite java `Ingredient` permettant de décrire des ingrédients. Chaque ingrédient possède deux propriétés : (1) son prix hors taxes et (2) son taux de graisse. Ce dernier dépend de la catégorie de l'ingrédient et prend la valeur de l'une des trois constantes déclarées dans l'interface `Ingredient` (constantes `graisseLegume`, `graisseSauce` ou `graisseViande`). Les deux propriétés d'un ingrédient peuvent être récupérées en appelant les deux méthodes `getPrix()` et `getPrix()`.

Q 2.1 Ecrire une classe abstraite `Ingredient` implémentant

1. Les champs décrits ci-dessus (attention aux types et modificateurs utilisés) ainsi qu'un compteur de référence globale (celui-ci indique le nombre d'ingrédients total instantiés)
2. Un constructeur `protected Produit(???)` permettant d'initialiser les champs et le compteur de référence

Q 2.2 Definir deux sous-classes `IngredientLegume` et `IngredientSauce` à la classe abstraite `Ingredient`.

On va s'intéresser à des clients qui proposent des commandes à des fournisseurs. Les fournisseurs choisissent les commandes qu'ils traitent selon leur propre stratégie de choix. Chaque commande a un identifiant unique. Une `Commande` contient un prix et un ensemble d'`Ingredient`.

Q 2.3 Définir une classe abstraite `Commande`, permettant d'ajouter des `Ingredient`. Attention a maintenir le prix global en fonction des ingrédients.

Les clients sont modélisés par des instances de la classe `Client`, qui contient

— Une liste de commandes en attente
— Une liste de commandes réalisées
— Une liste de fournisseur

Les fournisseurs sont des instances de la classe `Fournisseur`, qui contiennent

— Une liste de commandes traitées
— Un chiffre d'affaire
— Une stratégie de sélection des commandes
— Les fonctions `produitCommande(Commande commande)`, `augmenteChiffreAffaires(ajout : float)` et `choisitEtTraiteCommande(client : Client)`

Q 2.4 Ecrire les classes `Client` et `Fournisseur`. (Lire la question suivante pour les fonctions

Q 2.5 La méthode `choisitEtTraiteCommande` sélectionne l'une des commandes encore en attente du client passé en paramètre.

Q 2.6 Pour effectuer son choix, le fournisseur s'appuie sur la stratégie de choix fournie à sa création. Une stratégie de choix est définie par l'interface `Strategie` qui choisit une `Commande` dans une liste. Si aucune commande n'a pu être choisie, le fournisseur entre en maintenance (méthode maintenance). Lorsqu'une commande est choisie, elle est produite et le chiffre d'affaires est augmenté du prix de la commande.

La beauté d'un cloud kebab est que les participants (`Client` comme `Fournisseur` peuvent participer à la création d'un kebab répondant à une commande en ne réglant qu'une partie de la commande (seulement certains ingrédients).

Q 2.7 Décrire et implémenter les modifications que vous devez apporter à votre code pour permettre de gérer le cloud kebab.