TP 4 - Héritage, interface et exceptions

Machine à café [1]

On considère une *machine à café* (et autres boissons). La machine a pour but d'utiliser des *ingrédients : le café, l'eau, le lait et le chocolat.* Elle est constituée de *réservoirs* contenant chacun un ingrédient. La machine propose aussi des *recettes (expresso, café allongé, café au lait, chocolat chaud...)*. La machine est reliée à un *robinet* d'arrivée d'eau qui sera modélisé comme un réservoir de capacité infinie.

Exercice 1 – Eléments de base

Q 1.1 Situer tous les éléments en italique de la description dans une arborescence de classes en utilisant les liens d'héritage et de composition. Indiquer la/les classe(s) abstraite(s).
Penser à faire valider votre diagramme de classe auprès d'un chargé de TP.

Q 1.2 Donner le code de la classe abstraite `Ingredient` contenant un attribut `String nom`, un constructeur à un argument pour initialiser ce nom, un méthode `toString`. Dans la suite de l'exercice, le nom servira à décrire l'ingrédient (e.g. "café", "chocolat" ...)

Q 1.3 Ajouter une méthode standard `equals` testant l'égalité structurelle entre 2 ingrédients (c'est à dire entre leurs noms) en traitant le cas où nom n'est pas instancié (null).

Q 1.4 Donner les codes des classes `Cafe, Eau, Lait, Chocolat` héritant de la classe `Ingredient`. Le constructeur ne prend pas d'argument : le nom étant toujours celui de l'ingrédient.

Q 1.5 Donner le code du programme `TestMachineACafe` permettant de tester le début de ce code.

Exercice 2 – Réservoir

Q 2.1 Un réservoir a pour attribut un `Ingredient`, une capacite (un réel exprimé en litre), un niveau (également réel exprimé en litre). A la construction le réservoir est plein. Le réservoir dispose d'un accesseur sur l'ingrédient et d'une méthode remplir qui le remplit totalement.
Donner le code de la classe `Reservoir`

Q 2.2 Le réservoir a aussi une méthode `recuperer` qui prend en argument un réel (la quantité souhaitée par le client) : la méthode teste si le niveau est suffisant et, dans l'affirmative, décrémente le niveau ; sinon elle affiche un message explicite sur l'erreur.
On considère qu'une fois sur mille (aléatoirement), le réservoir connait une défaillance et n'est pas capable de délivrer l'ingrédient : Afin de simuler la machine à café, la méthode `recuperer` doit simuler ces défaillances. Vous ferez ici un simple affichage en cas d'erreur produite avec un message explicite. (Les erreurs de dépassement des quantités et de défaillance sont repris dans l'exercice Exception, vous pouvez ici les simuler directement). Donner le code de la méthode.

1. Enoncé originelle par Vincent Guigue

Q 2.3 Donner le code de la classe `Robinet`, qui est un réservoir de capacité infinie (`Double.POSITIVE_INFINITY`). On considère qu'un problème peut se produire une fois sur 500 : modéliser une méthode (aléatoirement) qui simule ce dysfonctionnement. Le message à donner à l'utilisateur est de vérifier que le robinet est bien ouvert.

Exercice 3 – Recettes

Une recette est composée d'un tableau d'ingrédients de taille fixe et d'un tableau de réels indiquant les quantités (toujours en litre) nécessaire à la conception d'une tasse de la boisson correspondante. Elle a aussi un prix (réel, en euros) et un nom. Le constructeur prend en arguments tous les éléments nécessaires à l'initialisation des attributs.

Q 3.1 La machine doit pouvoir accéder aux ingrédients d'une recette. Réfléchissez quel(s) accesseur(s) doivent être proposé(s).

Q 3.2 Donner le code de la classe `Recette`.

Exercice 4 – Machine

La machine est composée de deux tableaux comportant ses recettes et ses reservoirs. elle a un identifiant entier unique que vous initialiserez avec un compteur static. La machine reçoit un crédit de l'utilisateur (réel, en euros) : on va le supposer pour simplier égal au prix d'une boisson.

Q 4.1 Donner le code de base de la classe `Machine` avec un constructeur sans argument et deux méthodes `public void ajouterReservoir(Reservoir r)` et `public void ajouterRecette(Recette r)` pour configurer la machine. La classe possède aussi une méthode `public void ajouterCredit(double d)` pour mettre de l'argent (en euros) et une méthode `public void rendreLaMonnaie()` qui met le crédit à 0.

Q 4.2 Donner le code de la méthode public `void remplir()` qui correspond à l'action de l'agent d'entretien consistant à remplir tous les réservoirs au maximum.
La machine fonctionne de la manière suivante. D'abord, l'utilisateur ajoute de l'argent à la machine, puis sélectionne une recette, enfin la machine prépare la mixture en récupérant chaque ingrédient dans son réservoir. La machine sait rendre la monnaie (dans la pratique, on mettra simplement le crédit disponible à 0).

Q 4.3 Nous avons besoin d'une méthode `private Reservoir trouverReservoir(Ingredient i)` qui retourne le réservoir associé à l'ingrédient i s'il existe dans la machine et null sinon. On suppose qu'il n'y a qu'un réservoir par ingrédient pour éviter les complications.
Donner le code de cette méthode (en pensant fortement à la protection).

Q 4.4 Nous allons maintenant procéder au check-up de la machine pour vérifier que tout est OK. Donner le code de la méthode `public boolean checkup()` qui vérifie si toutes les recettes sont réalisables puis si tous les ingrédients de toutes les recettes sont bien disponibles dans la machine (à la première erreur, le test est interrompu et retourne false). Cette méthode affiche dans la console le nom des recettes avec la mention OK à coté si la recette est réalisable. Elle retourne true si toutes les recettes sont OK.

Q 4.5 Donner le code de la méthode `public boolean commander(int ri)` correspondant à l'appui sur le bouton ri de la machine. La machine indique la recette sélectionnée (si elle existe), vérifie que l'utilisateur a assez de crédit et prépare la recette en affichant des messages au fur et à mesure de la préparation. En cas de problème lors de la préparation, la machine affiche un message et retourne false. Le client n'est débité que si tout s'est bien passé (dans ce cas, on retourne true).
Note : on considère pour cette méthode ici que le check-up a été passé avec succès, tous les réservoirs associés à une recette sont toujours disponibles (ils peuvent simplement être vides ou en panne).

Exercice 5 – Testons

Q 5.1 Donner le code du programme `TestMachineACafe` permettant de valider le bon fonctionnement de la machine : création d'une recette simple avec quelques ingrédients (et ajout dans la machine), création des réservoirs associés (et ajouts dans la machine), check-up, test sur la monnaie disponible, test pour distribuer des boissons jusqu'à provoquer une erreur.

Exercice 6 – Réchauffement (Interface)

Q 6.1 Certains ingrédients et certaines recettes doivent passer par un système de réchauffement. Implémentez une interface `Chauffable` qui prévoit qu'un ingrédient ou une recette passe par ce système. Pour une recette ou un ingrédient chauffable, une méthode `testerSystemeRechauffement` qui teste si la recette et l'ingrédient pourront passer par le système de réchauffement. Pour simuler les défaillances de ce test, simuler une panne dans 1 cas sur 100.

Q 6.2 Modifier les recettes et les ingrédients que vous pensez être chauffables pour qu'ils intègrent cette propriété.

Q 6.3 On ajoute à la machine un tableau de toutes les recettes et ingrédients chauffables. Donner le code de la méthode `private testRechauffement()` qui teste si toutes les recettes et tous les ingrédients qui le nécessitent peuvent être réchauffés sans défaillance.

Q 6.4 Ajouter ce test de rechauffement diponible au checkup de la machine et vérifiez vos tests.

Exercice 7 – Exceptions

Q 7.1 Donner le code d'une classe `RecuperationIngredientException` qui étend les exceptions et qui a un constructeur avec un unique argument `String message`. Donner le code de la méthode en portant une attention particulière à la signature.

Q 7.2 On reprend ici les erreurs de la classe Réservoir : en cas de dépassement de capacité, la méthode `recuperer` lève une `RecuperationIngredientException` avec un message expliquant le problème (type d'ingrédient et quantité disponible). De même, une exception est levée dans le cas des défaillance.

Q 7.3 Les exceptions lancée par la méthode `recuperer` sont envoyés de la classe Réservoir. Déterminer l'emplacement le plus approprié pour la gestion des erreurs : ces erreurs seront des affichages le plus précis possible permettant de donner l'information pour la personne chargée du remplissage ou de

l'entretien de la machine.

Exercice 8 – Extensions

Q 8.1 Pour créer une machine connectée, on invente un nouveau type de `ReservoirConnecte` qui est construit avec 3 arguments : `Ingredient ingredient, double capacite, String adresse`. Il a une méthode `void mailTo(String message)` permettant d'envoyer un mail à la compagnie qui gère la machine (en utilisant l'adresse donnée lors de la construction). La machine envoie automatiquement un mail lorsqu'un réservoir passe à un niveau de remplissage inférieur à 10% (dans la pratique, on affiche simplement un message dans la console).
Donner le code de cette classe.

Q 8.2 L'architecture de la machine est-elle satisfaisante et évolutive ? Justifier cela en ajoutant une recette soupe à la tomate.