TP1 - Création des interfaces de machines à café et distributeurs

0. Initialisation du projet et du dépôt GIT

- Créer un dépôt GIT sur la forge sans README ni autre fichiers
- Copiez la commande git remote add origin ...
- Ajoutez-moi en tant que developer
- · Créée un nouveau projet Java sur IntelliJ ou NetBeans
- Ouvrez un terminal dans le dossier (avec votre IDE)
- Tapez les commandes suivantes

```
git init
git add .
git commit -m "first commit"
git branch -M main
[ LA LIGNE git remote add origin... QUE VOUS AVEZ COPIÉ ]
git push -u origin main
git checkout -b TP1
```

Pensez à commit/push à la fin d'une partie du sujet ou de la séance.

1. Création d'un distributeur

Nous allons créer un distributeur (de gâteau, boisson, café...) qui sera implémenté par la classe Distributeur .

Un distributeur possède:

```
1. un nom name de type String
```

- 2. une position position de type Position
- 3. un stock stocks de type Map<Product, Integer> qui permet de connaître le stock de chaque produit, pour l'implémentation vous utiliserez une HashMap

Les 2 premiers attributs seront passés en paramètre au constructeur du distributeur, ainsi qu'une liste de produits products de type List<Product> qui permettra de remplir la variable stock. Vous mettrez un stock par défaut à 10 pour chaque produit.

Un distributeur doit permettre de :

- connaître la liste des produits disponibles via la fonction **getProducts** qui retourne l'ensemble des clés de l'attribut stock via la fonction keySet
- connaître le stock de chaque produit via la fonction getStock qui prend en paramètre un produit et retourne son stock
- effectuer un achat via la fonction order qui prend en paramètre un produit et
 - s'il est disponible, retire 1 à son stock puis retourne true

- s'il n'est pas disponible, retourne false
- recharger un produit via la méthode fill qui prend en paramètre un produit et une quantité

Les positions seront représentées par un Enum, vous pourrez ajouter tous les lieux de votre choix. Voici un exemple :

```
public enum Position {
    DEPT_INFO,
    DEPT_CHIMIE,
    DEPT_GMP
}
```

Les produits seront représentés par un autre Enum avec les valeurs suivantes (que vous pourrez enrichir) :

- KIT_KAT
- M_NMS
- JUS_D_ORANGE

2. Votre première fenêtre

Pour créer l'interface de la machine, vous allez allez créer une classe InterfaceDistributeur qui étend de JFrame .

Cette classe possède:

- un attribut de type Distributeur dont l'instance est passée en paramètre du constructeur
- un attribut de type JLabel qui permettra d'afficher la position de la machine
- un attribut de type JLabel qui permettra d'afficher la liste des produits disponibles
- un bouton de type JButton qui affichera "Commander"
- une liste de JLabel qui permettra d'afficher le stock de chaque produit avec un JLabel par produit

Enfin, le titre de la fenêtre sera le nom de son distributeur.

Le constructeur de cette classe doit :

- se charger du titre de la fenêtre
- appeler une méthode initComponents qui:
 - va initialiser les autres attributs de votre classe (JLabel, liste de JLabel...)
 - mette les bonnes valeurs dans ces attributs
 - remplir la liste de JLabel tel que chaque JLabel contienne le nom et la quantité d'un produit (par exemple KIT_KAT : 3)
 - ajouter tous les éléments à afficher dans des conteneurs (les stocks doivent être dans un conteneur différent du reste)
 - ajouter les conteneurs à la fenêtre
- gérer la fermeture de la fenêtre (clique sur la croix)
- afficher la fenêtre

Afin de remplir la liste de JLabel, vous devez itérer sur les produits de votre distributeur via votre getProducts qui retourne un ensemble de produits (Set<Product>). Afin de récupérer le nom de chaque produit vous pouvez utiliser la fonction name. Enfin, pour obtenir le stock de chaque produit vous devez utiliser la fonction getStock de votre distributeur.

3. Contenu du main

Le but du main est d'afficher l'interface de visualisation d'un distributeur.

- ajouter une classe Application où se trouvera la fonction main (public static void main(String[] arguments))
- dans cette fonction, instanciez un distributeur avec quelques articles
- instanciez également une fenêtre InterfaceDistributeur qui prendra en paramètre votre distributeur

4. Plusieurs types de machines

Nous allons maintenant faire évoluer notre modélisation de distributeur afin de permettre différents types de machines.

4.1 Généralisation du concept de "machine"

Renommez votre classe Distributeur en Machine (on parle de **refactoring**, utilisez votre IDE pour le faire), ensuite faites-en une classe abstraite.

4.2 Création du nouveau distributeur

- Créez une nouvelle classe Distributeur qui étend de Machine (pensez bien à appeler le constructeur du parent via super).
- Dans dans la fonction main , remplacez l'appel au constructeur de Machine par le constructeur de Distributeur .

Normalement votre code doit de nouveau compiler.

4.3 Ajout du type de machine sur les produits

- Créez un nouvel Enum TypeMachine qui contient les valeurs DISTRIBUTEUR et MACHINE_A_CAFE .
- Dans Product :
 - ajoutez un attribut caption de type String,
 - ajoutez un attribut type de type TypeMachine ,
 - o ajoutez un constructeur qui prend en paramètre le caption et le type,
 - modifiez vos valeurs d'enum de façon à passer les paramètres au constructeur (M_NMS)
 devient M_NMS("M&M's", TypeMachine.DISTRIBUTEUR)),
 - ajoutez des getters afin de rendre accessibles en lecture le type et le caption ,
- Dans toutes les utilisations de name sur des objets de type Product , utilisez le getter de caption à la place.
- Dans InterfaceMachine faite de même pour afficher le nom des produits.

Votre application doit de nouveau compiler et vous devriez voir les captions au lieu des noms des enums de produits.

4.4 Création d'une machine à café

Créez une nouvelle classe Machine ACafe qui étend également de Machine .

La machine à café possède une contrainte particulière, il s'agit du stock de gobelets qui peut limiter les commandes.

Pour modéliser cette contrainte :

- ajoutez un attribut stockGobelet de type Integer
- mettez la valeur 10 à cet attribut dans le constructeur
- surchargez la fonction getStock (avec un @0verride) de façon à retourner le minimum entre le nombre de gobelets et le stock d'un produit
- surchargez la fonction order de façon à empêcher de passer une commande s'il n'y a plus de gobelets
- ajoutez la méthode fillGobelets qui permet recharger les gobelets avec en paramètre le nombre de gobelets à rajouter au stock

Dans Product ajoutez des articles avec le type MACHINE_A_CAFE

4.5 Création de l'interface de machine à café

Pour tester votre machine à café, ajoutez une classe InterfaceMachineACafe qui étend de InterfaceMachine et prend en paramètre de constructeur une instance de MachineACafe.

Cette classe doit permettre d'afficher en plus des informations d'InterfaceMachine, le stock de gobelets et un message "Recharger les gobelets" doit apparaître si le stock de gobelets est à 0

Dans votre main instanciez une machine à café ainsi qu'une interface de machine à café.

Testez.