Côntrole de connaissances 1
POO M1 MIAGE

PUTREGAI Alexandru

2.´Ecrire la classe Main et réaliser quelques essais pour s'assurer :

A) Qu'il n'est pas possible de créer plusieurs catalogues:

```
Catalog catalog = new Catalog(); // Impossible
Catalog catalog = Catalog.getInstance(); // Pour accéder au Catalog on doit passer par Singleton
```

A cause du fait que Catalog est un Singleton, on ne peut l'instancier qu'une seule fois. Pour y arriver nous devons par une méthode getter qui retourne l'instance unique.

B) Qu'un produit est unique quand on ajoute deux fois le même article dans un catalogue :

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Product product = (Product) o;
    return Float.compare(product.prix, prix) == 0 && Objects.equals(nom, product.nom);
}

@Override
public int hashCode() {
    return Objects.hash(prix, nom);
}
```

Afin d'aboutir à un tel résultat nous devons procéder à créer 2 @override(qui est utilisé pour définir une méthode qui est héritée de la classe parente) et plus précisément les 2 méthodes **equals** et **hasCode**.

C) Que la création d'un client n'est possible qu'en lui assignant un profil valide

```
public Customer(String nom, String prenom, String adresse, String email, String telephone) {
    this.id = idCounter++;
    this.nom = nom;
    this.prenom = prenom;
    this.adresse = adresse;

    if(verifTelephone(telephone)==true){
        this.telephone = telephone;
    }else{
        System.out.println("Le numéro de téléhpone introduit n'est pas correct");
        System.exit( status: 1);
    }

    if(verifEmail(email) ==true){
        this.email = email;
    }else{
        System.out.println("L'email introduit n'est pas correct");
        System.exit( status: 1);
    }
}
```

Afin d'assurer les bons inputs au niveau du numéro de téléphone ainsi que de l'email, deux méthodes de vérification Email et Téléphone sont développées

```java
/*  Formule Regex trouvé sur:
    https://howtodoinjava.com/java/regex/java-regex-validate-email-address/
 */
private static boolean verifEmail(final String email){
    String regex = "^[A-Za-z0-9+_.-]+@(.+)$";
    return email.matches(regex);
}
/*
https://www.developpez.net/forums/d1400873/java/general-java/expression-regex-valider-numero-telephone-france/
*/
private static boolean verifTelephone(final String telephone){
 */
    String regex = "(0|\\\\+33|0033)[1-9][0-9]{8}";
    return telephone.matches(regex);
}
```

D) Qu'il n'est pas possible de modifier le prix d'un produit depuis un panier:

Ayant toutes les variables en private ainsi que sans aucune méthode setter, la sécurité du prix des produits est garantie.

```java
//LES VARIABLES
private final int id;
private final float prix;
private final String nom;
private static int prochainProduit = 0;

//CONSTRUCTEUR
public Product(float prix, String nom) {...}
```

E) Que depuis un panier on puisse savoir le prix de chaque produit qu'il contient:

La classe Basket (Panier) peut faire appel aux méthodes get de la classe Product (Produit) pour avoir l'information sur le produit en question

```java
//GETTERS
public int getId() {return id;}
public float getPrix() {return prix;}
public String getNom() {return nom;}
```

F)Que chaque produit contenu dans un panier est effectivement référencé dans le catalogue et que la quantité disponible dans le catalogue correspond au stock restant en comptant ce qu'il y a dans tous les paniers.

Afin d'avoir une bonne notion du stock de chaque produit mis en vente avec une prudence afin d'éviter une survente de la quantité mise en vente, il est donc nécessaire de vérifier si le stock est supérieur à la quantité demandé (si c'est le cas, de soustraire par la suite) et sinon afficher une erreur de rupture de stock.

```java
//LES METHODES
public void ajouterProduit(final int produitId, int quantite) {
    Catalog catalog = Catalog.getInstance();
    Product produitAjoute = catalog.getProductById(produitId);
    int quantiteProduite = catalog.quantiteProduit(produitAjoute);
    if (produitAjoute != null & quantiteProduite > quantite) {
        produits.put(produitAjoute, produits.getOrDefault(produitAjoute, defaultValue: 0) + quantite);
        catalog.setProductById(produitAjoute, quantite);
    } else {
        System.err.println("Produit est en rupture de stock");
    }
}
```

```java
public Product getProductById(final int id) {
    for(Product p : products.keySet()){
        if(p.getId() == id){
            return p;
        }
    }
    return null;
}
//LES SETTERS
public void setProductById(final Product produit, final int quantite){
    products.put(produit, quantiteProduit(produit)-quantite);
}
```

G)Qu'il est possible de modifier le prix d'un produit en catalogue seulement en le remplaçant par un nouveau produit. Qu'advient-il alors des paniers qui font référence à ces produits ?

Il n'est pas possible de modifier le prix d'un produit, d'un côté car toutes les variables de la classe Product sont en private mais également car la classe Product ne possède pas de Setters.
Ainsi pour modifier le prix, le seul moyen sera de passer par la méthode "put". Cependant en faisant cette opération, la référence de l'ancienne donnée est effacée et une nouvelle prend place.
La conséquence de cette opération c'est que le panier qui possède un produit qui a été modifié avec la méthode "put" ne pourra pas passer, due au problème de trouver la référence.

Voici donc un résultat final en faisant plusieurs simulations pour inclure plusieurs situations possibles d'arriver

```
Client{
 id=0,
 Nom='PUTREGAI',
 Alexandru='Alexandru',
 adresse='351 Avenue de la Libération',
 email='alexandru.putregai@gmail.com',
 téléphone='0667224080'}


Le contenu avant l'achat du client:
Catalog{produits={
Product{id=0, prix=2.0, nom='maïs'}=17,
Product{id=2, prix=1.0, nom='scooter'}=7,
Product{id=1, prix=1.8, nom='chaussure'}=63}}


Le contenu du panier du client:
Basket{id=0, produits={
Product{id=0, prix=2.0, nom='maïs'}=2,
Product{id=2, prix=1.0, nom='scooter'}=4,
Product{id=1, prix=1.8, nom='chaussure'}=1}}


Le contenu du catalog après l'achat:
Catalog{produits={
Product{id=0, prix=2.0, nom='maïs'}=15,
Product{id=2, prix=1.0, nom='scooter'}=3,
Product{id=1, prix=1.8, nom='chaussure'}=62}}


Vérification de l'annulation en cas de rupture de stock
Basket{id=1, produits={}}
Produit est en rupture de stock
```