

Mise en route ... exercice

Considérons le code suivant :

```
public class IdentifyMyParts {  
    public static int x = 7;  
    public int y = 3;  
}
```

A. Combien de variable(s) de classe est(sont) déclarée(s) ? (et la(es) quelle(s))

B. Idem pour la(es) variable(s) d'instance

C. Quel est l'affichage pour le code suivant:

```
IdentifyMyParts a = new IdentifyMyParts();  
IdentifyMyParts b = new IdentifyMyParts();  
a.y = 5;  
b.y = 6;  
a.x = 1;  
b.x = 2;  
System.out.println("a.y = " + a.y);  
System.out.println("b.y = " + b.y);  
System.out.println("a.x = " + a.x);  
System.out.println("b.x = " + b.x);
```

L'exercice de la ménagère

Écrire une classe **article** disposant des attributs suivants:

- nbElem : entier (nombre d'articles différents)
- id : entier (identifiant unique)
- nom : string
- prix : float
- promotion : booléen

Remarques :

Les attributs doivent être protégés.

L'identifiant est calculé automatiquement

Question :

Ajouter 2 constructeurs. Un qui prend aucun paramètre et qui initialise le nom = « null », prix = 0 et promotion = false, et un autre prenant ces trois paramètres.

Ajouter les méthodes pour changer : le nom, le prix, la promotion. Ajouter les méthodes pour lire toutes les variables (identifiant, nom, prix et promotion)

Définir une classe **panier** qui contient 20 articles maximum (mesArticles).

Définir les méthodes :

- Prenant l'id, le prix, le nom et la promotion comme paramètre.
Vérifier que l'id utilisé est correct [0..19]. Retourne faux si id est faux.
- pour effacer les informations sur un article
- pour compter la somme des articles
- pour compter le nombre d'articles

Remarque :

Pour allouer un tableau d'objet il faut

1. Allouer le tableau (`a = new toto[dim]`)
2. Initialiser chaque objet (`for (int i=0; i<a.length; i++) a = new toto(...);`)

Soit l'interface et les deux classes suivantes :

```
public interface AppareilRéfrigérant
{
    public boolean EstActif();
    public float TempératureActuelle();
    public float TempératureIdéale();
}

class Frigo implements AppareilRéfrigérant
{
    private float températureActuelle;
    private final float températureIdéale = 5.0f;
    private boolean estActif = false;

    Frigo(float températureDépart)
    {
        températureActuelle = températureDépart;
        MiseAJour(0);
    }
    public void MiseAJour(int nombreMinutesEcoulees)
    {
        if (EstActif())
        {
            températureActuelle -= nombreMinutesEcoulees * 0.5f;
            if (températureActuelle < températureIdéale)
            {
                estActif = false;
                if (températureActuelle < 0.0f)
                    températureActuelle = 0.0f;
            }
        }
        else
        {
            températureActuelle += nombreMinutesEcoulees * 0.01f;
            if (températureActuelle > températureIdéale + 0.5f)
                estActif = true;
        }
    }

    // à compléter...
}

class Glacière implements AppareilRéfrigérant
{
    private float températureActuelle;
    private final float températureIdéale = 8.0f;

    Glacière(float températureDépart)
    {
        températureActuelle = températureDépart;
    }

    // à compléter...
}

// à compléter...
```

- a) Compléter les deux classes *Frigo* et *Glacière* en implémentant les méthodes manquantes.
- b) Ajouter une méthode *MiseAJour()* dans la classe *Glacière* reprenant le même paramètre *nombreMinutesEcoulees* que la méthode similaire de la classe *Frigo*. Cette fonction calculant simplement la dissipation de la chaleur à raison de *0.02* degrés par minute.
- c) Ecrire un programme *MarchandDeGlace* créant un frigo et une glacière avec comme paramètres initiaux respectifs 19. et 3 degrés. Ce programme
 1. Affichera l'état d'activation du Frigo juste après sa création.
 2. Opèrera une boucle de mise à jour de 60 minutes dans laquelle les deux méthodes MiseAJour() seront appelées.
 3. Affichera l'état d'activation du Frigo après la fin de la boucle.
 4. Affichera les températures actuelles finales du frigo et de la glacière.