TP 2 (partie 1) : création de classes

Le but de ce TP est de définir et de manipuler quelques classes élémentaires. Il se compose de deux exercices, le premier a pour but de définir une classe proposant des méthodes de classe pour les saisies. Le deuxième consiste à définir des classes pour une application.

Une classe est définie par le mot clé class et peut être structurée comme ceci :

```
class <nom de la classe> {

        <définitions de variables d'instance>        // utilisez le mot clé private pour
                                                     // forcer l'encapsulation

        <définitions de variables de classe>  // utilisez les mot clés static et
                                              //private

        <définitions des méthodes d'instance>        // vous pouvez utilisez public

        <définitions de méthodes de classe>          // utilisez le mot clé static

}
```

Une méthode de classe (static) est définie de la manière suivante :

```
static  public <type retourné> <nom méthode> ( <liste d'arguments>) {

                <déclarations et instructions>
  }
```

Le mot clé return délivre la valeur de la méthode. Si la méthode ne délivre rien (pas d'utilisation du mot clé return) le type retourné par la méthode est void (ce qui signifie vide). L'annexe donne un exemple de définition de classe.

Exercice 1 : Classe Voiture

Question 1/ Définir une classe Voiture avec les attributs suivants privés suivants :

- Un identifiant id (entier),
- Une marque (une chaîne de caractères),
- Sa vitesse (un nombre à virgule),
- [redacted] (un nombre à virgule).

kilomètres

On pose la contrainte suivante sur l'attribut « id » : celui-ci doit être incrémenté à chaque création d'un nouvel objet de type « Voiture ». Au bout du compte, chaque voiture possédera son propre identifiant. Par exemple, la voiture v1 l'identifiant 0, la voiture v2 l'identifiant 1, etc.

Question 2/ Définition des constructeurs

a- Définir un constructeur permettant d'initialiser les attributs (marque, vitesse[redacted]- kilomètres, [redacted] d'un objet de type « Voiture » par des valeurs passées en paramètre.

b- Définir un constructeur sans paramètre.

Question 3/ Définir les getters et les setters associés aux différents attributs de la classe.

public String toString()

Question 4/ Définir la méthode [redacted] permettant d'afficher les informations d'une voiture. Quel est l'intérêt de cette méthode ?

Question 5/ Écrire dans une autre classe contenant une méthode main (), un programme testant la classe Voiture.

Exercice 2 : Simulation de la gestion d'un stock

Dans cet exercice, on souhaite gérer un stock de produits frais.

Question 1 : la classe d'exécution

Créer une classe TestStock qui contient une méthode main (). Cette classe vous servira, au besoin, à chaque étape, à tester la validité de vos différentes classes.

Question 2 : création de la classe Produit

Un objet de type Produit est caractérisée par :

- sa référence (de la classe String).
- sa date d'entrée dans le stock (de type int).

Ces attributs seront considérés comme privés.

La date d'entrée correspond à la date du jour où l'entrée en stock du produit est effectuée. Il faudrait peut-être compléter par une indication, par exemple : *le 1er janvier est représenté par le 1, le 2 janvier par le 2... le 31 janvier par le 31 et le premier février par le 32 etc.*

De plus, elle comporte notamment deux méthodes :

- une méthode pour afficher un produit.
- un constructeur ayant en paramètre la date du jour et lisant au clavier la référence du produit à l'aide de la classe Scanner.

Question 3 : création de la classe Pile

À chaque entrée d'un produit dans le stock, ce produit est saisi au clavier et la date d'entrée en stock est la valeur de la date du jour (dateJour).

Le temps passant, la fraîcheur des produits en stock diminue. Dans cet exercice, nous partirons du principe que la gestion du stock s'effectue selon la doctrine : "*le produit le plus frais, donc le plus récemment entré en stock, est sorti le premier du stock*". Nous allons donc utiliser une structure de pile pour modéliser ce type de gestion de stock.

Proposez et testez une classe Pile ayant en variables d'instance privées un tableau de Produit et son indice, et comportant les méthodes suivantes :

- le constructeur Pile(int max) qui réservera la place pour *max* produits et initialisera l'indice du tableau ;
- boolean pileVide(), délivrant Vrai si la pile est vide, Faux sinon ;
- boolean pilePleine(), délivrant Vrai si la pile est pleine, Faux sinon ;

- void empiler(Produit p), permettant de stocker un nouveau produit p au sommet de la pile ;
- void depiler(), permettant de sortir le produit situé au sommet de la pile ;
- Produit sommet(), délivrant le produit au sommet de la pile.
- void afficherStock(), permettant d'afficher le stock complet de produits contenus dans le stock.

Question 3 : création de la classe Stock

La classe Stock a pour variables d'instance inaccessibles à l'utilisateur :

- une Pile.
- la date du Jour initialisée à 1. La date du jour sera donc la date du dernier produit rentré. Quand cela est nécessaire, cette date du jour sera augmentée, notamment lors de l'entrée d'un produit dans la pile représentant le stock.

Elle comporte les méthodes suivantes :

- le constructeur Stock(int taille) qui réserve un stock de taille produits ;
- void entrer(Produit p), qui permet d'entrer un produit p dans le stock ;
- void sortir(int dateJ), qui sort le produit le plus frais du stock. À la sortie d'un produit du stock, il ne peut être vendu si sa date d'entrée dans le stock est supérieure de 5 jours à celle de la date du jour dateJ. Dans un tel cas, on sortira successivement de la pile tous les produits ne pouvant être vendus.

Question 4 : ajout d'un menu

Dans la classe Stock, écrivez une méthode void afficheMenu() qui propose à l'utilisateur le menu :

- e : entrée d'un produit dans le stock ;
- s : sortie d'un produit du stock ;
- i : incrémenter la date du jour ;
- q : quitter.

- À chaque entrée en stock, votre programme devra afficher un message du style "*Produit xxx ajouté au stock*".
- De même, après chaque sortie vous devrez afficher soit "*Produit xxx sorti du stock*", soit "*Produit xxx périmé, stock intégralement supprimé*".

Annexe : rappel de la structure d'une classe

```
class Personne
{
        // variable d'instance donc privées
        private String nom;
        private int age;

        // variable de classe donc static
        static private int nb=0;

        // constructeur : de même nom que la classe
        // éventuellement plusieurs constructeurs
        public Personne(String lenom, int lage){
                nom=lenom;
                age=lage;
        nb++;
   }
        public Personne(){
                nom="Le Gac";
                age=20;
                nb++;
        }
        // Méthodes d'instance
        public int getAge(){
                return this.age;
        }
        public String getNom(){
                return this.nom;
        }
        public void setNom(String nouveauNom){
                this.nom=nouveauNom;
        }
        public void setAge(int nouvelAge){
                this.age = nouvelAge;
        }
        // Méthode de classe
        static public int obtenirNb(){
                return nb;
        }
        // éventuellement une méthode main pour tester la classe
        static public void main(String args[]){
                ...
        }
```

```
}
```