PLANCHE D'EXERCICES N°3
- PROGRAMMATION ORIENTÉE OBJET EN JAVA -

Une proposition de correction des Tps est disponible sur github via ce lien :
https://github.com/m-assili/dev-isimg/tree/master/2021_2022/POO_Java

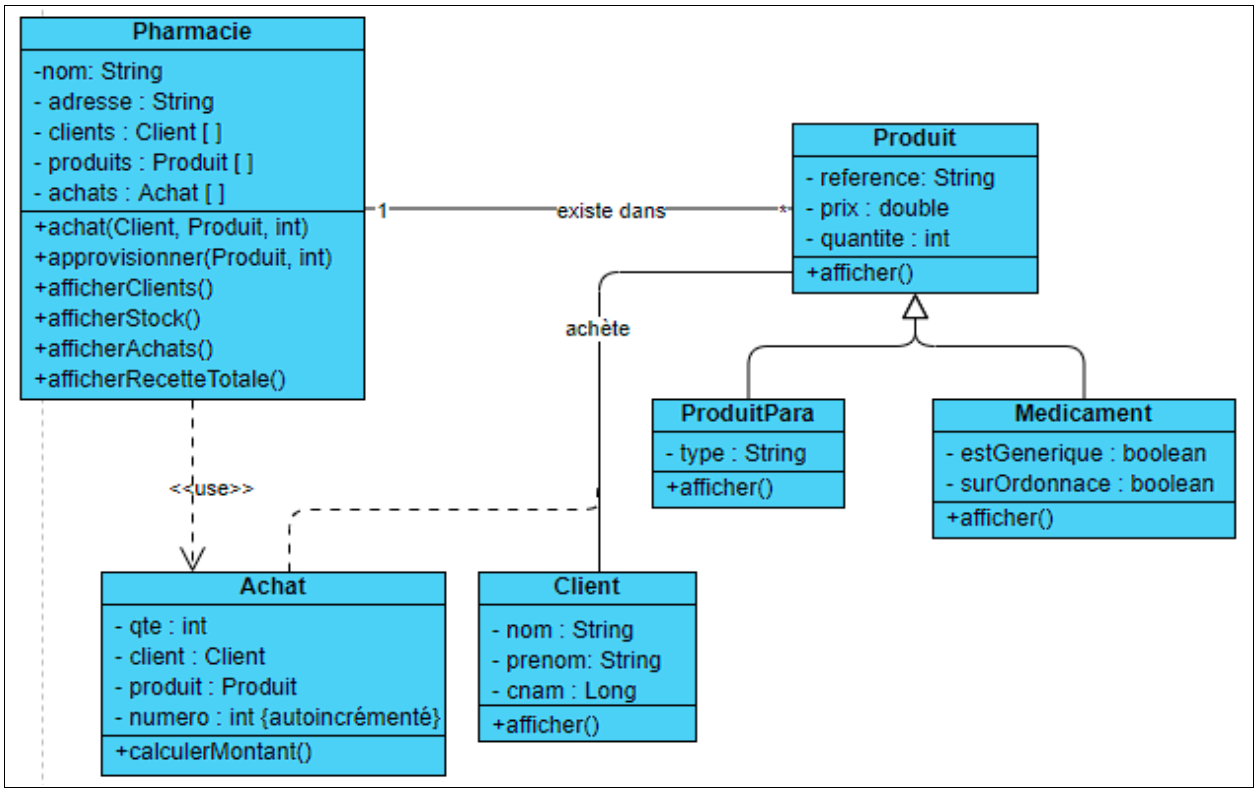
Exercice 1: Héritage

On désire automatiser la gestion d'une bibliothèque qui comprend des livres et des adhèrents.

1. Créez une classe Personne qui contient :
   - les attributs privés : nom, prenom et age ;
   - un constructeur pour initialiser les différents attributs ;
   - une méthode afficher() qui affiche le nom, le prénom et l'age.
2. Créez une classe Adherent qui hérite de Personne et qui :
   - ajoute l'attribut numAdherent ;
   - redéfinit la mèthode afficher().
3. Créez une classe Auteur qui hérite de Personne et qui :
   - ajoute l'attribut numAuteur ;
   - redéfinit la mèthode afficher().
   - le programme génère une exception si l'utilisateur veut instancier un objet Auteur avec une valeur négative de numAuteur.
4. Un livre est définit par un numèro ISBN (entier), un titre et est écrit par au plus 4 auteurs.
   - créez la classe Livre ;
   - ajoutez une mèthode afficher() qui affiche le ISBN, le titre et les informations des auteurs.
5. Créez une classe Bibliothèque qui contient une méthode main(), dans laquelle :
   - déclarez (instanciez) un adhèrent ;
   - instanciez un livre qui est écrit par deux auteurs ;
   - affichez les informations de l'adhèrent et du livre.

Exercice 2: classe, héritage, agrégation, ...

Soit le diagramme de classe suivant:



1. A partir de ce diagramme de classe, implémentez les différentes classes. N'hésitez pas à ajouter des méthodes aux classes à chaque fois que c'est nécessaire.
2. Ecrire une classe TestPharmacie qui permet de :
    - créer une pharmacie
    - créer et ajouter des produits à cette pharmachie
    - créer un client
    - le client effectue des achats de quelques produits
    - afficher les achats effectués dans la pharmacie
    - afficher la recette totale de la pharmacie

Exercice 3: héritage, classe abstraite, ...

On se propose de programmer un jeu. De façon générale, on compte trois grandes familles d'éléments dans le jeu :

– Les personnages (héros, adversaires, monstres, ...)

– Les objets (équipement des personnages, trésors, ...)

– Les décors (essentiellement des murs)

Un niveau de jeu est décrit de la façon suivante :

– Une carte, qui comprend les éléments de décors, les objets et les personnages

– Un titre, qui décrit le niveau

– Un personnage spécial : celui du joueur.

Pour résumer :

un personnage est un élément

un objet est un élément

un décor est un élément

un niveau est composé d'une carte et d'un titre

Question 1 : Coder la classe abstraite **Element**.

Celle-ci doit contenir les méthodes publiques suivantes :

- getDescription( ) qui retourne une chaine de caractère décrivant l'élément.
- setDescription(String s) qui met à jour la description d'un objet .
- getType( ) qui retourne une chaine de caractère indiquant le type de l'élément :
  OBJET pour un objet, PERSONNAGE pour un personnage et DECOR pour un décor.

Question 2 : Coder la classe **Objet**

Cette classe dérive de la classe élément. Elle n'a aucun nouveau attributs ni méthodes, mais penser à implémenter la méthode getType().

Question 3 : Coder la classe Personnage

Cette classe hérite de la classe Element. Elle doit implémenter les méthodes publiques suivantes :

- getName( ) qui retourne une chaine de caractère (le nom du personnage);
- setName(String s) qui met à jour le nom du personnage ;
- getVitalite( ) qui retourne un entier correspondant aux points de vie actuel du personnage
- setVitalite(int i) qui met à jour la vitalité actuelle du personnage
- getVitaliteMax( ) qui retourne un entier correspondant aux points de vie total du personnage
- setVitaliteMax(int i) qui met à jour la vitalité totale du personnage

la description du personnage (retournée par la méthode getDescription()) doit comporter le nom du personnage, ainsi que des indications sur sa vitalité (totale et actuelle).
Penser à vérifier que la vie actuelle d'un personnage ne dépasse pas son maximum.
Un personnage peut porter jusqu'à trois objets.
Implémentez les méthodes suivantes de la classe personnage :

- afficheInventaire( ) retourne une chaine de caractère décrivant l'équipement
- ajouteObjet(Objet i) ajoute l'objet i à l'équipement si il y a encore de la place
- poseObjet(int numero) retire l'objet numero de l'équipement pour faire de la place et renvoie une
- référence vers cet objet

Pour la méthode ajouteObjet(Objet i), vous pouvez gérer le cas ou on tente de rajouter un équipement à un personnage qui est déjà entièrement chargé de deux façons : soit en changeant la valeur de retour de cette méthode (plus simple) soit en utilisant les exceptions (mieux noté). Dans tous les cas, utiliser un tableau d'objets semble une solution simple à mettre en oeuvre.

Question 4 :

Implémentez la classe Niveau qui doit comporter les attributs et les méthodes suivantes :
Attributs :

- carte : tableau d'Elements à deux dimensions (10 x 10 cases)
- PJ : référence sur un objet de type Personnage (le personnage qui sera controlé par le joueur)
- titre : le titre du niveau

Méthodes :

- void setElement(int i,int j, Element k) : Ajoute un élément k à l'emplacement [i,j] de la carte (une case ne peut contenir qu'un seul objet à la fois).
- Element getElement(i,j) : retourne une référence sur l'élément contenu à la case [i,j]
- String decritCarte(int i, int j) : retourne la description de l'élément contenu dans la case, ou le texte « case vide » si la case ne contient pas d'éléments.
- String decritCarte( ) : retourne la description, sous forme de chaine de caractères, de chaque case de la carte (avec toujours « case vide » pour décrire les cases qui ne contiennent pas d'éléments).
- Personnage getPersonnagePrincipal( ) : retourne une référence sur le personnage joueur.
- void setPersonnagePrincipal(Personnage p) : donne une nouvelle valeur à PJ.
- String getTitre( ) : retourne une référence sur le titre du niveau.
- void setTitre(String t) : change le titre du niveau.

Constructeurs :

Vous implémenterez un constructeur par défaut, qui crée un personnage principal

Vous implémenterez un constructeur ayant un personnage pour paramètre (personnage qui sera donc crée hors du constructeur)

Question 6 : tester vos implémentations en utilisant le programme principal mis à votre disposition (dans l'annexe), et en le modifiant si besoin.

Exercice 4: Interface

Une expression arithmétique peut être évaluer et décompiler (avoir une chaine contenant l'expression). Un Entier est une expression arithmétique ayant une valeur entière qui peut être évaluer (on récupère sa valeur) et decompiler (avoir une description de sa valeur). Une addition entière est aussi une expression arithmétique qui évalue et décompile deux entiers. Décompiler une addition arithmétique pour deux entiers 1 et 2, retourne une chaine de caractère sous cette forme (1+2).

1. Identifier les objets à définir et implémentez les.
2. Ecrire une classe Main pour :
    - Créer deux entiers v1 et v2 vallant respectivement 42 et 17
    - Créer une addition entière pour v1 et v2
        - Afficher la décompilation de v1 et son évaluation
        - Afficher la décompilation de v2 et son évaluation
        - Afficher la décompilation de l'expression d'addition et son évaluation
    - Affecter v1 à v2
    - Créer dans v1 un nouveau entier vallant 13
        - Afficher la décompilation de v1 et son évaluation
        - Afficher la décompilation de v2 et son évaluation
        - Afficher la décompilation de l'expression d'addition et son évaluation
    - On affecte v1 à la partie gauche de l'addition
    - On affecte l'addition à la partie droite de l'addition (pour avoir l'addition de 3 entiers)
        - Afficher la décompilation de v1 et son évaluation
        - Afficher la décompilation de v2 et son évaluation
        - Afficher la décompilation de l'expression d'addition et son évaluation

```
public class main{
 public static void main(String S[])
 {
      Personnage PJ1 = new Personnage();
      PJ1.setName("Chevalier miteux...");
      PJ1.setDescription("personnage alcoolique mais chevaleresque...");
      Personnage gobelin1 = new Personnage();
      gobelin1.setName("Gobelin Teigneux");
      gobelin1.setDescription("Petit humanoide verdatre et primitif...");
      Objet gourde = new Objet();
      gourde.setDescription("Petite gourde en peau");
      Objet epee = new Objet();
      epee.setDescription("Epee Durandil");
      Objet casque = new Objet();
      casque.setDescription("Casque Lebohaum");
      System.out.println("----------------------------");
      System.out.println("Affichage Personnages : ");
      System.out.println(PJ1.getDescription());
      System.out.println(gobelin1.getDescription());
      System.out.println("----------------------------");
      System.out.println("Affichage Objets");
      System.out.println(gourde.getDescription());
      System.out.println(epee.getDescription());
      System.out.println(casque.getDescription());
      System.out.println("----------------------------");
      System.out.println("Un peu d'equipement ? ");
      /* Normalement, c'est vide */
      System.out.println(PJ1.afficheInventaire());
      /* Ajoutons quelques objets */
      PJ1.ajouteObjet(epee);
      PJ1.ajouteObjet(gourde);
      PJ1.ajouteObjet(casque);
      /* Normalement, c'est plein */
      System.out.println(PJ1.afficheInventaire());
      /* Un nouvel objet pour voir */
      Objet armure = new Objet();
      armure.setDescription("Armure rutilante");
      /* On essai de le rajouter, normalement, c'est impossible */
      PJ1.ajouteObjet(armure);
      System.out.println( PJ1.afficheInventaire() );
      /* On pose d'abord un objet, puis on essai à nouveau  */
      PJ1.poseObjet(0); /* dehors l'objet 0 */
      System.out.println( PJ1.afficheInventaire() );
      PJ1.ajouteObjet(armure);
      System.out.println( PJ1.afficheInventaire() );
      /* Testons la vitalité de notre personnage */
      System.out.println( "Point de vie actuels : "+PJ1.getVitalite() );
      System.out.println( "Point de vie maximal : "+PJ1.getVitaliteMax() );
      /* On change son nom ? */
      System.out.println(PJ1.getDescription() );
      PJ1.setName("Bob");
      System.out.println(PJ1.getDescription() );
      /*************************************************************
                  MISE EN PLACE DU NIVEAU...
       *************************************************************/
       /* on crée quelques types de décors ? */
      Decors mur1 = new Decors();
      Decors mur2 = new Decors();
      mur1.setDescription("C'est un mur en bois, pas très épais");
```

```
      mur2.setDescription("C'est un mur en pierre, ça à l'air solide");
      /* Construction d'un niveau avec le constructeur par défaut */
      Niveau monNiveau1 = newNiveau();
      /* Mur en bois */
      monNiveau1.setElement(1,1,mur1);
      monNiveau1.setElement(1,2,mur1);
      monNiveau1.setElement(1,3,mur1);
      /* mur en pierre */
      monNiveau1.setElement(5,5,mur2);
      monNiveau1.setElement(5,6,mur2);
      /* Certains objets trainent par terre... */
      monNiveau1.setElement(3,3,epee);
      monNiveau1.setElement(3,4,casque);
      /* Et en plus, y'a des monstres... */
      monNiveau1.setElement(7,7,gobelin1);
      monNiveau1.setElement(8,8,gobelin1);
      /*
        Question : Si le gobelin qui est dans la case 7,7 est modifié
        (perte de points de vie par exemple)
        Que ce passe-t-il pour le gobelin dans la case 8,8 ?
       */
      System.out.println(monNiveau1.decritCarte());
      /* Au fait, mon constructeur de niveau a-t-il bien crée un personnage par
      défaut ? */
      Personnage toto = monNiveau1.getPersonnagePrincipal();
      System.out.println(toto.getDescription());
      /* Changeons de personnage principal */
      monNiveau1.setPersonnagePrincipal(PJ1);
      /* Et verifions le deroulement... */
      toto = monNiveau1.getPersonnagePrincipal();
      System.out.println(toto.getDescription());
      /* On crée un nouveau niveau, avec le second constructeur */
      Niveau nouveauNiveau = new Niveau(PJ1);
      /* On teste ... */
      toto = monNiveau1.getPersonnagePrincipal();
      System.out.println(toto.getDescription());
      System.out.println(nouveauNiveau.decritCarte());
    }
}
```