Université Sultan Moulay Slimane
FST – Béni Mellal
Département d'Informatique
2017/2018

Contrôle

(Module : Java)
(2heures)

Exercice 1 (3 points) :

Compléter le tableau suivant :

Vocabulaire	Signification
abstract class	.....
Final class	.....
Evénement	.....
ODBC	.....

Exercice 2 (17 points) :

Un pharmacien souhaite automatiser la gestion de son stock de médicaments en utilisant un ordinateur. Chaque médicament est identifié par les informations suivantes: Code, Nom, Prix unitaire, Type (Sirop, Comprimé, Injection, ...etc). Composants (Produits constituant le médicament), Patient (Personnes qui peuvent utiliser ce médicament : Enfant, Adolescent ou Adulte) et Quantité (Nombre de boites) dans le stock. La gestion de la pharmacie s'effectue à l'aide des classes suivantes :

1) Ecrire une classe **Medicament** contenant : (5pts)

- Le code du médicament (**code** : int).
- Le nom du médicament (**nom** : String).
- Le prix unitaire du médicament (**prix** : double).
- La type du médicament (**type** : String).
- Le nombre de produits constituant le médicament (**nb** : int).
- Les produits constituant le médicament (**comp[]** : String).
- Le patient qui va consommer le médicament (**pat**: String).
- La quantité du médicament dans le stock (**quan** : int).
- Un constructeur **Medicament()** qui prend en paramètre le code du médicament.
- Une méthode **clavier()** pour saisir et retourner une chaîne de caractères.
- Une méthode **lecture()** permettant la saisie du reste des informations (**nom, prix, type, nb, comp, pat et quan**).
- Une méthode **affichage()** pour afficher toutes les informations du médicament.

*Chaque donnée membre et chaque méthode devront être définies avec les modificateurs suivants: **private**, **public** ou **static**.*

Université Sultan Moulay Slimane
FST – Béni Mellal
Département d'Informatique
2017/2018

2) Tester la classe **Medicament** en écrivant une application JAVA (dans une autre classe) capable de créer un objet de type **Medicament**, *de saisir* et d'afficher les informations de l'objet crée. (1pt)

3) Ecrire une classe **Pharmacie** répondant au cahier des charges suivant : (4pts)

- Le nombre maximum des médicaments (**max**: int).
- Un tableau des médicaments (**table[]**: **Medicament**).
- Le nombre des médicaments enregistrés (**nbe**: int).
- Un constructeur qui prend en paramètre deux entiers pour initialiser le nombre maximum des médicaments que peut contenir la pharmacie et le nombre des enregistrements.
- Une méthode ***boolean ajouteMedicament(Medicament m)*** qui ajoute un médicament au tableau « **table** », la méthode renvoie ***true*** si l'ajout s'est correctement effectué, ***false*** sinon. Dans le cas où le médicament existe dans le tableau « **table** », il faut faire une mise à jour dans la quantité et le prix (en cas de changement du prix, il faut retenir le nouveau prix).
- Une méthode ***afficher(Medicament[] ph,int d)*** qui permettra d'afficher l'ensemble des médicaments enregistrés dans un tableau de type **Medicament**.

Chaque donnée membre et chaque méthode devront être définies avec les modificateurs suivants : ***private***, ***public*** ou ***static***.

4) Modifier l'application JAVA pour tester la classe **Pharmacie**. (1pt)

5) Enrichir la classe **Pharmacie** d'une méthode **float total_prix()** qui calcule et retourne le prix total des médicaments enregistrés dans le tableau « **table** ». (2pts)

	Enfant	Adolescent	Adulte
TVA	5%	7%	10%

→ patient

6) Ajouter à la classe **Pharmacie** une méthode **Medicament[] supprime_Medicaments(String cpt)** qui supprime les médicaments contenant le composant « **cpt** » dans le tableau « **table** ». Cette fonction retourne un tableau contenant les médicaments supprimés. (2pts)

7) Ajouter à la classe **Pharmacie** une méthode **int fichier_vehicule(String ch)** qui permet de générer un fichier texte contenant les informations d'un médicament identifié par le nom « **ch** ». (2pts)
Cette fonction retourne 1 dans le cas de réussite et 0 dans le cas inverse.

Description du médicament

Code :.....
Nom :.....
Type :.....
Prix unitaire :.....
Quantité :.....
Composants :.....
Patient :.....

throws IOException

Exercice 1°/

abstract class : Aucun objet ne peut instancier cette class Seules les classes peuvent les déclarer des méthodes abstraite

final class : des classes finalise ne peuvent pas être héritable.

Evenement : est un message envoyé à l'application, il peuvent être d'origine Evenements diverses (action de l'utilisateur (clic sur bouton, deplacement souris..)) et événements systèmes (chargement d'un fichier ...)

ODBC : (Open DataBase Connectivity) : une interface développée par Microsoft pour uniformiser les accès aux bases de données.

Exercice 2°/

```
public class Medicament
{
    private int code;
    private String nom;
    private double prix;
    private String type;
    private int nb;
    private String[] comp;
```

```
private String pat ;
private int quan ;
    public Medicament ( int code)
    {
        this . code = code ;
         nb = 0 ;
         comp = new String [nb] ;
    }
    static String clavier ()
      {
        Scanner input = new Scanner(System.in);
         return input . nextLine ();
      }
// méthode lecture ()
// Application Methode — Principale .
// les getters
// les setters
```

```
public class Pharmacie
{
    int max ;
    int nbe ;
    Medicament [ ] table ;
    public Pharmacie ( int max , int nbe)
    {
        this . max = max ;
        this . nbe = nbe;
        table = new Medicament [max] ;
    }
/* Méthode rechercheMedicament. */
public int rechercheMedicament ( int code)
  {   int pos = -1;
    for (int i =0 ; i <nbe ; i++)
      {
        if( table[i] . getCode( ) == code)
          {
            pos = i ;
            break;
          }
      }
    return pos;
}
```

/* Methode ajoutMedicament() */

```
public boolean ajoutMedicament(Medicament m)
{   int indice = rechercheMedicament(m.getCode());
    boolean res = false;
    if(nbe < max)
    {
      if( indice == -1)
        {
          table[nbe] = m;
          nbe ++;
          res = true;
        }
      elseif( indice != -1)
        {
          table[indice].setQuan( m.getQuan());
          table[indice].setPrix(m.getPrix());
          res=true;
          }
      }
    return res;
}
```

```
/* Méthode afficher */
public void afficher(Medicament[] ph, int d)
{
    for(int i=0 ; i<d ; i++)
    {
        System.out.println(" de medicament "+(i+1)+ ":");
        System.out.println( ph[i].afficher() );
    }
}
/* Méthode totalP_prix */
public float totalP_prix()
{
    float PrixtotalP = 0;
    float prix = 0;      float prifTTC = 0;
    for(int i=0 ;  i<nbe  ; i++)
    {
        prix = (float) table[i].getQuan()*
tab[i].getPrix();
if( table[i].getPatient().equals("Enfant"))
    { prixTTC = prix + 0.05 ; }
```

```
elseif( table[i].getPatient().equals('AdoPesent'))
    { prixTTC = prix + 0.07 ; }
  elseif(table[i].getPatient().equals('Adulte'))
    { prixTTC = prix + 0.1 ; }
  PrixTotal += prixTTC ;
 }
 return PrixTotal ;
}
/* Methode supprime_Medicament */
public Medicament[] supprime_Medicament(String cpt)
 {
   Medicament[] Msupp = new Medicament[nbe];
   for(int i=0 ; i<nbe ; i++)
    {
    for(int j=0 ; j<nbe ; j++)
      {
       if(table[i].comp[i].equals(cpt))
        {Msupp[i] = table[i];
         table[i] = table[i+1];
          nbe -- ;
      } }
} return Msupp;
```

```
/* Methode fichier_Medicament */
public int fichier_vehicule ( String ch) throws IOException
{
    int res = 0;
    File myfile = new File ("Medicament.txt");
    FileWriter fw = new FileWriter ( myfile);
    PrintWriter pw = new PrintWriter (fw);

    for(int i=0 ; i < nbe ; i++)
    { if( table[i].getNom().equals(ch))
        {
          String chaine = table[i].toString();

          pw.print( chaine);
          pw.close();
          res = 1;
        break;
        }
    }
return res;
}
```