Travail à rendre avant le 3 janvier 2022

A envoyer par mail à papasamour.diop@ugb.edu.sn

Exercice 1 :

Vous devez développer pour une chaîne de télévision un logiciel permettant de gérer la programmation journalière d'émissions télévisuelles. Pour simplifier le problème, toutes les heures ou les durées sont représentées par des valeurs entières (13h, 15h, 22h…). Une émission peut être de trois types :
divertissement, fiction et reportage. Une émission de type divertissement dure obligatoirement 2 heures, possède un nom et est systématiquement présentée par un animateur. Une fiction se définit par le nom du film, l'année de sa réalisation, le nom du réalisateur et s'il s'agit ou non d'une rediffusion. Enfin un reportage est défini par un nom, un thème (fixé parmi les trois thèmes : information, animalier, culturel) et une durée.

1) Proposer une solution fondée sur les héritages entre classe pour représenter toutes les émissions possibles. Donner pour chaque classe, la liste de ses attributs et les paramètres de ses constructeurs.
Comment coderiez-vous le fait que le thème d'un reportage soit prédéfini ?

2) Implanter cette solution sous Eclipse en Java et tester tout d'abord les classes que vous avez imaginées en instantiant différents objets de votre choix (avec les constructeurs) pour chacune de celles-ci.

3) La programmation d'une émission dans la journée dépend du type d'émission mais se traduit par le fait de lui fixer une heure de début de diffusion et de calculer l'heure de fin.

a. Les divertissements durent systématiquement 2 heures, mais on ne peut les programmer qu'entre 18h et 23h.

b. Les fictions qui ne sont pas des rediffusion ne se programment qu'en début de soirée, c'est-à- dire qu'à 21h, alors qu'une rediffusion peut se programmer n'importe quand dans la journée.

c. Enfin, les reportages ne se programment qu'à des heures creuses (14h –18h et 0h-6h) et s'ils ont une durée inférieure, égale à 1heure.

Proposer une solution fondée sur la notion de classe abstraite et de polymorphisme permettant de décrire la programmation ou non n'importe quelle émission à une heure donnée. Comment initialiser efficacement ces heures de début et de fin de diffusion.

4) Définissez enfin un programme télé comme un ensemble fini (tableau) d'émissions hétérogènes que vous remplirez d'émission de votre choix, programmer à une heure de votre choix. Décrire puis implémenter les algorithmes vous permettant :

a. Afficher la liste des émissions programmées dans la journée

b. Tester s'il y a une superposition de programmation.

c. Afficher heure par heure les émissions programmées pour vérifier que tous les créneaux horaires ont bien été remplis.

Exercice 2

On veut écrire un programme gérant le coût des enseignants d'une université. Chaque enseignant a un nom, un prénom et un certains nombre d'heures de cours assurées dans l'année. Il existe plusieurs catégories d'enseignants.
Les enseignants-chercheurs sont payés avec un salaire fixe (2000 euros par mois pour simplifier) plus le revenu des heures complémentaires (40 euros de l'heure). Les heures complémentaires sont les heures effectuées au-delà des 192h. Ainsi, un enseignant-chercheur qui donne 200h de cours dans l'année recevra 2000*12+8*40=24320 euros sur l'année.

Les vacataires sont des enseignants venant d'un organisme extérieur à l'université. Cet organisme est décrit par une chaine de caractères. Les vacataires sont payés 40 euros de l'heure pour toutes leurs heures.
Les étudiants de 3e cycle (doctorants) peuvent assurer des cours et sont payés 30 euros de l'heure, mais ne peuvent dépasser 96h de cours. Un doctorant qui fait plus de 96h ne recevra que le salaire correspondant à 96h de cours. Finalement, sur tous les salaires versés, on applique des charges égales à un certain pourcentage du salaire, ce pourcentage est le même pour tous les enseignants (par exemple 100%, ce qui signifie que le montant des charges est égal au montant des salaires). On veut, pour chaque enseignant, pouvoir connaitre ce qu'il coûte à l'université (salaire + charges).
Proposez du code Java qui représente ces informations et soit le plus conforme aux principes de la programmation objet.

Exercice 3

Écrivez un programme orienté objets qui permet de gérer une pharmacie. La classe principale débute comme suit. Il s'agira de la compléter.

```
import java.util.Scanner;
class Pharmacie {
   private static Scanner scanner = new Scanner(System.in);
   public static void main (String args[]) {
      Client[]  clients = new Client[2];
      Medicament[]  medicaments = new Medicament[2];

      clients[0] = new Client("Malfichu",0.0);
      clients[1] = new Client("Palichon",0.0);

      medicaments[0] = new Medicament("Aspiron", 20.40, 5);
      medicaments[1] = new Medicament("Rhinoplexil",19.15, 5);
      int choix;

      do {
         choix = menu();

         switch (choix) {
         case 1 :
            achat(clients, medicaments);
            break;
         case 2 :
            approvisionnement(medicaments);
            break;
         case 3 :
            affichage(clients, medicaments);
            break;
         case 4:
            quitter();
         }
      }
      while (choix < 4);
   }
   // Les méthodes utilitaires
```

```
    static int menu() {
        int choix = 0;
       System.out.println();
       System.out.println();
       System.out.println("1 : Achat de médicament");
       System.out.println("2 : Approvisionnement en  médicaments");
       System.out.println("3 : Etats des stocks et des crédits");
       System.out.println("4 : Quitter");
        while ((choix != 1) && (choix != 2) && (choix != 3) && (choix != 4)) {
            choix = scanner.nextInt();
        }
        return choix;
    }
    //  Méthodes auxiliaires à compléter

}

// Autres classes à compléter
```

La pharmacie gère des clients et des médicaments. Un client est caractérisé par un nom et un crédit. Le crédit représente la somme que ce client doit à la pharmacie. Le crédit peut être négatif si le client a versé plus d'argent que le montant. Un médicament est caractérisé par un nom (de type String), un prix (de type double) et un stock (de type int). Les méthodes à compléter auront les caractéristiques suivantes:

- affichage(..) permet d'afficher les clients et leurs crédits respectifs ainsi que les médicaments et leurs stocks respectifs.
- approvisionner(..) permet d'approvisionner le stock d'un médicament. Le nom du médicament à approvisionner ainsi que la quantité à ajouter au stock doivent être lus depuis le terminal. Lorsque le nom du médicament est introduit, il faut vérifier qu'il s'agit bien d'un nom connu dans la liste des médicaments de la pharmacie. Le programme doit boucler jusqu'à introduction d'un nom correct. Cette procédure de vérification sera prise en charge par la méthode lireMedicament(..) décrite plus bas.
- achat(..) permet de traiter un achat fait par un client. l'achat porte sur un médicament donné dans une quantité donnée. Pour cette transaction le client paie un certain prix. Une opération d'achat aura pour effet de déduire la quantité achetée du stock du médicaments correspondant et d'augmenter le crédit du client (d'un montant équivalent au montant de l'achat moins la somme payée).
  Les noms du client et du médicament doivent être lus depuis le terminal. Le programme doit boucler jusqu'à introduction de noms connus aussi bien pour les clients que les médicament. Ces procédures de vérification seront prises en charge par les méthodes lireClient et lireMedicament (voir plus bas). La quantité achetée et le montant payé sont aussi lus depuis le terminal. Ils seront supposés corrects.
- quitter(..) affiche le message ``programme terminé!''.
- Vous définirez une méthode auxiliaire lireClient(..) prenant comme paramètre un tableau de clients. Elle permettra de lire le nom d'un client depuis le terminal et de vérifier si ce client existe dans le tableau des clients. Dans ce cas le client sera retourné. Cette méthode doit boucler jusqu'à ce qu'un client soit trouvé. Elle sera utilisée par la méthode achat(..). Une méthode similaire, lireMedicament(..) sera fournie pour les médicaments. Elle sera utilisée par les méthodes achat(..) et approvisionnement(..).
- Vous êtes libre de définir, en plus de ces méthodes, toutes celles que vous jugerez nécessaires.

Lors de la programmation, il vous est demandé de respecter *scrupuleusement* les indications suivantes:

- Votre programme doit être bien modularisé à la fois sous forme de méthodes auxiliaires de la méthode main() et sous forme de classes pour les objets du programme. En particulier:
    - Chaque variable et méthode doit être déclarée dans la classe la plus adaptée.
    - Dans chaque classe liée à un objet, il faut utiliser le mot clé private *autant que possible*. En particulier, toutes les variables d'instances seront privées.
    - Votre programme doit montrer une utilisation judicieuse de tableaux aux endroits adaptés.
    - la comparaison de deux chaines s1 et s2 se fera au moyen de la méthode java equals() (s1.equals(s2) retourne true si s1 et s2 sont identiques).

- Vous pouvez utiliser les méthodes scanner.nextInt(), scanner.nextDouble() et scanner.nextLine().

Exercice 4

1. Écrire un programme dans lequel on demande à l'utilisateur de saisir un entier en gérant l'exception dans le cas où il ne saisit pas un entier correctement.
2. Écrire un programme dans lequel on demande à l'utilisateur de saisir un entier en gérant l'exception dans le cas où il ne saisit pas un entier correctement en lui demandant de refaire la saisie.

Exercice 5

Soit à développer une application pour la gestion d'un stock.

Un article est caractérisé par son numéro de référence, son nom, son prix de vente et une quantité en stock.

Le stock est représenté par une collection d'articles**.**

Travail à faire:

Créer la classe article contenant les éléments suivants :

- Les attributs/propriétés.
- Un constructeur d'initialisation.
- La méthode ToString().

Dans la classe Program créer :

Le stock sous forme d'une collection d'articles de votre choix.

Un menu présentant les fonctionnalités suivantes :

1. Rechercher un article par référence.
2. Ajouter un article au stock en vérifiant l'unicité de la référence.

3. Supprimer un article par référence.
4. Modifier un article par référence.
5. Rechercher un article par nom.
6. Rechercher un article par intervalle de prix de vente.
7. Afficher tous les articles.
8. Quitter