Examen de fin de semestre

(Durée 1h30)

Exercice 1 :

Ecrire une classe contenant les méthodes suivantes :

a. *lireTabEnt* : initialise un tableau T de n entiers (n 10).

b. *divise :* affiche le résultat de la division de chaque élément T[i] du tableau par (T[i]-x), x étant une valeur entière donnée. En considérant le type d'Exception « division par zéro », si pour un élément du tableau T[i] – x = 0, la méthode affiche « Tentative de division par zéro » puis continue et affiche le reste des résultats de la division.
Exemple :

n=5 T :

2	4	6	9	15

Pour x=4 les résultats sont : *-1, Tentative de division par zéro, 3, 1, 1*

c. *main* : teste les méthodes définies précédemment.

Exercice 2 :

On désire gérer la programmation journalière d'émissions télévisuelles. Pour simplifier le problème, toutes les heures ou les durées des émissions sont représentées par des valeurs entières (13h, 15h, 22h,…). Une émission peut être de trois types : divertissement, fiction et reportage.
Un divertissement dure obligatoirement 2 heures, possède un nom et un animateur.
Une fiction est définie par le nom du film, la durée, 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, scientifique ou culturel) et une durée.

1. Proposer une solution fondée sur les héritages entre classes pour représenter toutes les émissions possibles. Ecrire le code de chaque classe avec la liste des attributs et les constructeurs avec paramètres. Définir aussi dans chaque classe une méthode *toString()* et une méthode *affiche()* qui affiche le type de l'émission ainsi que les attributs associés.

2. La programmation d'une émission dans la journée dépend de l'heure de début de diffusion et de l'heure de fin, pour cela il faudra définir une méthode qui retourne l'heure de fin.

3. Pour chaque type d'émission il est possible de modifier l'heure de début de diffusion et d'afficher les modifications précédées du type de l'émission.
Proposer une solution fondée sur la notion de classe abstraite et de polymorphisme permettant de décrire la programmation de n'importe quelle émission à une heure donnée.

4. Compléter la classe suivante :

```
public class Principale {
     public static void main(String[] args) {
           Emission[] programme=new Emission[15];
           programme[0]=new Fiction(6,3,"F1",true,1989,"RR");
           programme[1]=new Fiction(9,3,"F2",true,1990,"RR");
           programme[2]=new Fiction(12,3,"F3",true,1992,"RR");
           programme[3]=new Reportage(12,1,"Inde","culturel");
           programme[4]=new Reportage(16,1,"terre","information");
           programme[5]=new Divertissement(12,"D1","FF");
           programme[6]=new Fiction(21,3,"Ha",false,2010,"SSSS");
            …
// le nombre de programmes par jour = à n
a)// afficher l'heure de fin de chaque programme
           …
b)// modifier l'heure de début s'il y a une superposition de programmation
  …
  }// fin main
}// fin Principale
```

Exercice 3 :

Une liste chaînée peut être représentée dans un tableau de <valeur, indice>, la tête de liste étant l'indice du premier élément de la liste, chaque élément du tableau contenant une valeur (contenu) et un entier (suivant), cet entier indiquant l'indice dans le tableau du prochain élément de la liste. Le suivant du dernier élément de la liste est un indice spécial (-1 par exemple). Exemple :

tete | 2

La tête de la liste est à la position 2 's'
le suivant est à la position 0 'i'
le suivant est à la position 3 'm'
et ainsi de suite…

	contenu	suivant
0	'i'	3
1	'p'	5
2	's'	0
3	'm'	1
4	'e'	-1
5	'l'	4

a. Définir la classe *Element* avec les attributs sécurisés, *contenu* (char) et *suivant* (int). Dans cette classe définir un constructeur avec paramètres.

b. Définir la classe *TabListe* avec les attributs sécurisés, tableau *T* de type *Element*, sa taille *n* (n 100) et la position du premier élément *tête*. Définir un constructeur sans paramètres ainsi que la méthode affiche(). Définir aussi les méthodes :
   - *ajoutTete(char contenu)* : pour ajouter un élément en tête, il faudra le placer dans le tableau (en dernière position), le chaîner à l'ancienne tête et modifier la valeur de tête pour qu'elle indique ce nouvel élément,
   - *ajoutApres(char contenu)* : placer l'élément en dernière position du tableau et le chainer à l'ancien dernier élément. Pour cela il faudra définir une méthode *rechercheDernier()* qui retourne la position du dernier élément du tableau.
   - En utilisant *ajoutTete* et *ajoutApres* définir la méthode *creerFiFo()* pour la lecture des éléments du tableau.

c. Définir la classe *Test* (principale) et tester les méthodes définies précédemment.

Bon courage