Programmation Orientée Objet - TD

Exercice 1

Le concept général de télévision se rattache, comme toute notion, à des données (voire d'autres concepts) ainsi que des actions. Les données sont par exemple le poids, la longueur de la diagonale de l'écran, les chaînes mémorisées ainsi que l'état allumé/veille/éteint de la télévision. Les actions peuvent consister à changer de chaîne, à allumer, éteindre ou mettre en veille la télévision. D'autres actions peuvent consister en un réglage des chaînes.

- Proposer une représentation du concept de télévision.
- Proposer une représentation objet de télévision.

Exercice 2

Modéliser sous forme d'un enregistrement un nombre complexe, une date et une personne. On utilisera les notations suivantes : *TComplexe*, *TDate, TVehicule* et *TPersonne*. Une personne est décrite par son nom, son prénom, son jour, son mois et son année de naissance.

1. Quelles opérations peut-on associer aux types précédents ?
2. Ecrire les algorithmes de quelques opérations selon le modèle suivant :

```
{ description des données}
Opération ( paramètres )
{ description des résultats en fonction des données}
```

Exercice 3

Reprendre les types précédents et le transformer en type objet, en transformant les procédures et fonctions associées en méthodes et les variables en attributs.

1. Décrire la partie interface de l'unité **`date`** qui définit le type objet **`TDate`**. Une date est définie par un jour, un mois et une année.
2. Spécifier les méthodes qui permettront de générer une date au hasard puis de l'afficher.
3. Spécifier d'autres méthodes qui vous semblent utiles.
4. Ecrire un algorithme qui teste ces méthodes.

Exercice 4

Soit une application permettant de saisir deux personnes puis d'afficher le nom et le prénom de la plus jeune des deux. Décrire le programme principal de cette application et les nouvelles méthodes utiles à ce dernier.

Exercice 5

Considérons une population comprenant au maximum 20 personnes. Cette collection de personnes est implantée dans un tableau.

1. Décrire la population par un type objet, en y incluant une méthode `MoyenneAge` permettant de calculer la moyenne d'âge de la population, cette année.
2. Décrire un programme permettant de tester la méthode `MoyenneAge.`

Exercice 6

Pour enrichir le type `TPersonne` défini dans l'exercice n°2, on propose de créer un type `TAdulte` qui possède en plus de `TPersonne` les deux informations suivantes :

- une situation qui peut être célibataire, marié, divorcé ou veuf
- un conjoint de type `TAdulte`

Ajouter un nouveau type appelé `TEnfant` qui devra avoir :

- un père de type `TAdulte`
- une mère de type `TAdulte`.

1. Dessiner le graphe d'héritage entre les trois types d'objets.
2. Sur le graphe, placer les nouveaux champs et donner leur type.

Exercice 7

1. Spécifier, puis implanter une méthode `TAdulte.mariage` qui marie deux personnes adultes, sachant que deux personnes de même sexe ou déjà mariées ne peuvent contracter un mariage.
2. Que manque-t-il au type `TPersonne` pour que cette méthode fonctionne ?

Exercice 8

On a distingué les personnes adultes et les enfants et on a organisé les types `TPersonne`, `TAdulte` et `TEnfant` en hiérarchie. On veut à présent définir un type objet `TPopul` pour représenter une communauté de personnes.

Les éléments d'une population (objet de type `TPopul`) seront construits d'adultes et d'enfants et les méthodes qu'on leur appliquera s'adapteront dynamiquement à leur type réel.

- Compte tenu du cahier des charges, donner l'interface complète du type `TPopul.`

On suppose l'existence d'une méthode `TPersonne.afficher` qui permet d'afficher à l'écran le nom, le prénom et la date de naissance d'une personne.

- Spécifier puis implanter la méthode `TPopul.afficher`

Exercice 9

1. Spécifier une méthode qui recherche une personne dans une population.
2. Une personne est identifiable par son nom, son prénom et sa date de naissance.

Traduire cette idée en spécifiant une méthode du type `TPersonne`, qui sera utile à la programmation de la méthode de recherche dans une population.
3. Implanter ces deux méthodes.

Exercice 10
Dans le type `TPopul`, spécifier puis implanter une méthode d'ajout d'une personne :

- qui utilise la méthode de saisie de `TPersonne`
- qui n'ajoute que des personnes qui ne sont pas déjà dans la population.

Exercice 11
Spécifier et implanter des méthodes calculant :

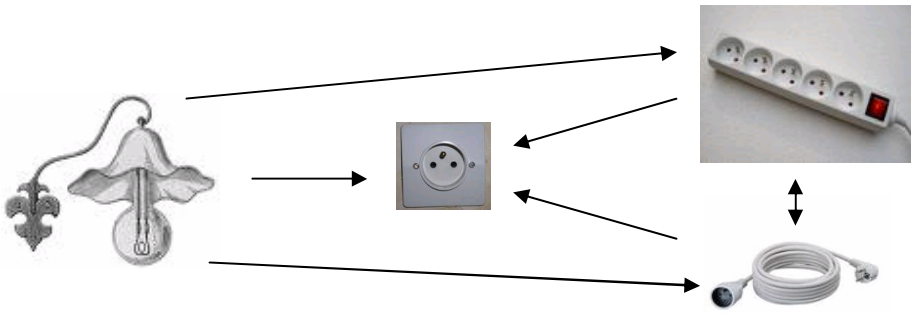
- le nombre de couples mariés dans une population donnée
- le nombre moyen d'enfants par foyer

Exercice 12
Spécifier et implanter une méthode permettant de créer une liste linéaire chainée d'entiers de la manière suivante :
- les entiers sont tirés au hasard entre 1 et 1000
- la suite se termine quand l'entier courant est un multiple de 10, ce dernier n'est pas inclus dans la suite

Exercice 13
Spécifier et implanter une méthode qui construit la sous-liste des entiers pairs de la liste courante.

Exercice 14
Spécifier et implanter les méthodes réalisant les opérations suivantes :
1. Calcul du nombre d'éléments d'une liste
2. Test de la présence d'un composant donné dans une liste
3. Test de la présence d'occurrences multiples dans la liste
4. Insertion d'un entier à un rang donné
5. Suppression de la première occurrence d'un élément donné

Exercice 15:
On veut modéliser un ensemble de composants électriques : des lampes, des rallonges de fil électrique et des prises multiples.
Les rallonges et les prises multiples sont considérées comme des sources électriques dans la mesure où on peut leur rajouter ou retirer en sortie (sur leur(s) prise(s) femelle(s)) un composant électrique (pour la rallonge) ou plusieurs (pour la prise multiple).



Chacun des trois types de composants électriques peut brancher son unique entrée (sa prise mâle) sur une source électrique ou sur une prise murale (femelle). Son entrée peut aussi se débrancher.
Chaque composant électrique est dans un des deux états suivants : soit sous tension, soit hors tension, suivant s'il est (directement ou indirectement) raccordé ou pas à une prise murale. Lorsqu'un composant électrique change d'état, il propage son état au(x) composant(s) électrique(s) qu'il a en sortie. Si c'est une lampe, elle ne propage rien et se contente d'afficher qu'elle s'allume ou qu'elle s'éteint. Un composant devient sous tension dès qu'il est branché à une prise murale ou à une source électrique sous tension ou parce que le composant auquel il est relié en entrée vient de passer sous tension. Inversement, un composant devient hors tension dès qu'il est débranché d'une prise murale ou d'une source électrique sous tension ou parce que le composant auquel il est relié en entrée vient de passer hors tension.
Pour simplifier, on considérera que les lampes n'ont pas d'interrupteur et qu'elles sont allumées si elles sont sous tension.
Le but final de l'exercice est de définir des classes qui permettent par exemple l'exécution de la suite d'instructions suivantes :

```
Lampe l1 = new Lampe("Lampe1");
Lampe l2 = new Lampe("Lampe2");
Rallonge r = new Rallonge();
PriseMultiple p = new PriseMultiple(2);
l1.brancheSur(p);
l2.brancheSur(p);
p.brancheSur(r);
r.brancheSurPriseMurale(); // affiche "Lampe1 allumée" puis "Lampe2 allumée"
l1.debranche(); // affiche "Lampe1 éteinte"
l1.brancheSur(p); // affiche "Lampe1 allumée"
r.debranche(); // affiche "Lampe1 éteinte" puis "Lampe2 éteinte"
```

Après certaines instructions, on a mis en commentaire l'affichage qui était déclenché indirectement par l'instruction.
- Décrire une hiérarchie de classes permettant de modéliser au mieux les composants électriques tels qu'ils ont été décrits et de façon à ce qu'on puisse exécuter la suite d'instructions donnée ci-dessus. Plus précisément, il vous est demandé de donner l'ensemble des classes à définir, liste des méthodes de chacune de ces classes.

Indices : les seules classes concrètes nécessaires sont Lampe, Rallonge et PriseMultiple. Ne pas déclarer de classe pour modéliser une prise murale (cf exemple). Une des possibilités de bonne modélisation de votre hiérarchie implique l'existence d'une classe abstraite et d'une interface.

- Donner les attributs des classes définies dans la question précédente et implémenter leurs méthodes.