TD n°2 – Types, Création de classes, Tableaux

Rappels :

- Vous devez organiser correctement le code réalisé en TD.
- Dans l'IDE Eclipse, vérifiez que vous travaillez toujours dans votre *workspace* correspondant au répertoire du cours M213 sur votre espace personnel.
- A chaque nouvel énoncé de TD, vous devez créer un nouveau projet (*Java Project*) sous Eclipse portant le numéro du TD concerné (ici TD2).
- Vous créez ensuite un package pour chaque exercice à l'intérieur du projet.

Types primitifs et String

Pour chaque ligne, indiquez si l'initialisation est autorisée. Si oui indiquez la valeur de la variable.

```
double a = 3 + 2;
int b = 3 + 2.5;
double d = (3 / 2);
double e = (3.0 / 2);
int f = "124";
String g = 5.78;
String h = ""+5.78;
String i = "Chat" + "Pitre";
char j = 'A';
boolean k = (a <= 5.78);
boolean l = k && !(d < e);
```

Triplet d'entiers

Définir une classe `TripletEntier` qui représente un triplet ordonné d'entiers (3 valeurs).

Ecrire la classe complète contenant les constructeurs adéquats (constructeur vide, par copie de 3 valeurs), les méthodes accesseurs des attributs, ainsi que les méthodes suivantes :

- `somme` : retourne la somme des attributs
- `moyenne` : retourne la moyenne des attributs
- `concatenation` : retourne la chaîne de caractères correspondant à la suite des attributs
- `ajoutElement` : ajoute l'élément donné en premier paramètre à la position spécifiée en deuxième paramètre
- `ajout1erElement` : ajoute l'élément donné en paramètre en première position
- `contient` : vérifie si l'élément passé en paramètre est contenu dans le triplet courant
- `equals` : vérifie si 2 triplets sont identiques (= ont le même état). Les positions des valeurs doivent être les mêmes dans les deux objets.
- `clone` : crée un nouvel objet `TripletEntier` ayant le même état que l'objet courant (utilise le constructeur par copie)
- `toString` : retourne la chaîne de caractères représentant les 3 entiers entre crochets, et séparés par des virgules

Attention à bien définir l'encapsulation vue en cours, les paramètres et les types résultats pour les méthodes.

Récupérer le fichier `TestTriplet` dans *SupportCours*, l'inclure dans le même package que la classe `TripletEntier`, et vérifier l'exécution de votre code à partir de cette classe de tests.

Tableaux à 1 dimension

Le but de cet exercice est de réaliser une classe `EnsembleEntierBorne` permettant de manipuler des ensembles d'entiers de taille bornée.

1) **Attributs :** Chaque instance de `EnsembleEntierBorne` est définie par 2 attributs :
   a. une constante entière `MAXIMUM` qui ne peut être initialisée qu'une seule fois, et
   b. un tableau de booléens où chaque case indique si oui ou non l'entier (correspondant à l'indice) est contenu dans l'ensemble.
2) **Constructeurs :** l'unique constructeur prend en paramètre un entier. L'ensemble ne pourra contenir que des entiers compris entre 0 (inclus) et cet entier (inclus). *Le tableau a donc une case de plus que cette valeur.*
3) **Méthodes :**
   a. `add(…)` Cette procédure prend un entier en paramètre et le rajoute à l'ensemble (si l'entier est déjà dans l'ensemble ou ne peut être contenu dans l'ensemble, la procédure est sans effet).
   b. `remove(…)` Comme pour la procédure `add`, mais cette fois-ci il s'agit d'enlever un élément.
   c. `doesContain(…)` Cette fonction prend un entier en paramètre et renvoie vrai si et seulement si l'entier est dans l'ensemble.
   d. `toString()` Représente l'ensemble sous forme `{2,4,6,7,99}` (on peut éventuellement laisser une virgule en trop dans une première version, sinon utiliser la fonction `substring` de la classe `String` afin de supprimer la dernière virgule inutile).
   e. `intersect(…)` Cette fonction prend un `EnsembleEntierBorne` en paramètre et renvoie l'ensemble correspondant à l'intersection entre l'ensemble courant et celui entré en paramètre.
4) **Main :** Dans une classe `TestEnsemble`, instancier des objets `EnsembleEntierBorne` et tester les méthodes développées.
5) En utilisant la description des ensembles d'entiers bornés de l'exercice précédent (constructeur + méthodes), ajouter dans une fonction associée à la classe `TestEnsemble` (qui ne dépend pas des instances de la classe) le code Java correspondant à l'algorithme suivant :

```
variables
        maxim : entier
        premiers : ensemble d'entiers entre 0 et maxim
        i, j : 0..maxim
début
        pour i de 2 à maxim faire
                inclure i dans premiers
        finpour
        pour i de 2 à maxim faire
                si i appartient à premiers alors
                        pour j  de 2i à  maxim par pas de i faire
                                enlever j de premiers
                        finpour
                finsi
        finpour
        afficher premiers
fin
```

Que fait cet algorithme ? Regarder ce qu'il se passe sur différents exemples.
Modifier le `main` pour appeler cette fonction et la tester sur différents exemples.

Jour de la semaine

Le but de cet exercice est d'écrire une classe `JourSemaine` permettant de gérer les jours de la semaine (sans se soucier du quantième, du mois, ou de l'année). Cette classe comprend :

1) **Constantes** : Créer une constante qui regroupe toutes les écritures en minuscules des jours de la semaine dans un tableau appelé *jours*, i.e. un tableau de chaînes de caractères.
2) **Attributs** : Les instances de `JourSemaine` ne seront définies que par un attribut : le numéro du jour de la semaine (un entier entre 0 et 6, il faudra toujours veiller à ce qu'il ne sorte pas de cet intervalle lors de l'instanciation). Le numéro correspondra bien évidemment à l'indice du jour ayant son écriture dans la constante *jours* : *par exemple : 0 – lundi, 1 – mardi, 2 – mercredi, …*
3) **Constructeurs** : Il y aura 3 constructeurs :
    a. un sans paramètre qui correspond à un lundi,
    b. un prenant un numéro de jour (entier) comme paramètre, et
    c. un prenant une chaîne de caractères en paramètre. Ce 3ème constructeur cherche si cette chaîne est dans le tableau *jours* : si c'est le cas (ex : « vendredi »), il est le numéro du jour indiqué ; sinon (ex : « van Dredi »), c'est par défaut un lundi.
4) **Méthodes** :
    a. `getNumero()` C'est un accesseur, il renvoie la valeur de l'attribut correspondant.
    b. `toString()` La méthode habituelle pour décrire une instance. Elle renvoie la chaîne de caractères correspondante contenue dans *jours*.
    c. `veille()` Cette fonction retourne une autre instance de `JourSemaine` qui correspond au jour précédant de la semaine (attention à la veille de lundi !)
    d. `lendemain()` Cette fonction retourne une autre instance de `JourSemaine` qui correspond au jour suivant (attention de même avec dimanche).