Exercices sur les itérateurs

1 Feu Tricolore

Un feu tricolore se compose de trois couleurs (on peut utiliser la classe `Color` en Java, avec des valeurs définies sur le modèle de la constante `Color`.red). Itérer sur le feu tricolore consiste à parcourir (sans fin a priori) les trois couleurs. Ecrire une classe pour représenter les feux tricolores, un itérateur de feu tricolore et un programme montrant une utilisation. L'opération `remove` n'est pas supportée, ce que vous représenterez par le signalement de l'exception `UnsupportedOperationException`.

2 Itérer sur un objet complexe

Une feuille de salaire comprend les éléments suivants qui peuvent être représentés par des attributs dans une classe `FeuilleSalaire` :

- employeur
- salarié
- convention collective
- nombre d'heures payées
- prélèvements fiscaux : CRDS, CSG, cotisations salariales
- net à payer

Définir une classe `FeuilleSalaire` itérable. Définir un itérateur de feuille de salaire qui retourne successivement chacun des champs de la feuille de salaire. Cela peut servir par exemple pour écrire une méthode d'affichage ou pour récupérer toutes les informations qui sont des nombres (définir de telles opérations en utilisant l'itérateur). L'opération `remove` n'est pas supportée, ce que vous représenterez par le signalement de l'exception `UnsupportedOperationException`.

Puis écrivez un programme mettant en œuvre une feuille de salaire et un itérateur.

3 Le distributeur de bonbons Pez

On se propose de définir un distributeur de bonbons qui s'inspire du système des distributeurs de briques PEZ. Il s'agit d'un tube de 12 bonbons, que l'on récupère par le haut du tube grâce à un mécanisme qui fait remonter les bonbons au fur et à mesure.

Une classe minimale est donnée ci-dessous. Le programme `main` qu'elle contient montre le fonctionnement de l'itérateur que vous devrez écrire (essayez d'exécuter le main sans avoir défini l'itérateur, puis après l'avoir défini). L'itérateur est l'objet qui traverse le tube. Dans son implémentation, on choisit d'effacer la case contenant la brique récupérée. C'est donc un itérateur "destructif".

```
public class DistributeurBonbonPez{
       private Color couleurTube = Color.pink;
       private String formeBouchon = "Minnie";
       private BonbonPez[] tube = new BonbonPez[12];
      public void remplit()
       { for (int i = 0; i < 12; i++) tube[i] = new BonbonPez(); }
```

```
        public String toString()  {// à écrire ...}

        public static void main(String[] argv){
               DistributeurBonbonPez d= new DistributeurBonbonPez();
               d.remplit();
               System.out.println(d);

                for (BonbonPez b : d) // boucle for each{
                      System.out.println(b);
                      System.out.println("Apres retrait " + d);
}}}
```

4 Itérateur sur une suite réelle

Nous souhaitons représenter des suites réelles, et plus précisément nous nous limitons ici aux suites constantes à partir d'un certain rang. On rappelle qu'une *suite réelle* est une application u d'une partie de $\mathcal{N}$ dans $\mathcal{R}$. On considère ici les suites définies sur $\mathcal{N}$.

Une suite est *constante à partir d'un certain rang* s'il existe un entier naturel n et un réel a tels que pour tout entier naturel $n' \geq n$, $u(n') = a$.

Question 1 *Ecrivez une classe* `SuiteConstanteRang` *pour représenter les suites constantes à partir d'un certain rang. On pourra stocker les* $n-1$ *premiers éléments dans une* `ArrayList`*.*

Question 2 *Ecrivez une classe* `IterateurSuiteConstanteRang` *qui permette d'itérer sur une suite constante à partir d'un certain rang. Un élément a toujours un successeur. L'opération* `remove` *ne pourra enlever que des éléments entre les rangs* 1 *et* $n-1$*.*

Question 3 *Rendez la classe* `SuiteConstanteRang` *itérable.*

Question 4 *Que va écrire le programme suivant et comment l'interprétez-vous ? Est-ce que cela mérite une modification de ce qui a été précédemment écrit ?*

```
public static void main(String[] argv){
         ArrayList<Double> a = new ArrayList<Double>();
         a.add(3.0); a.add(4.0); a.add(5.0); a.add(6.0); a.add(7.0);
         SuiteConstanteRang suite = new SuiteConstanteRang(a,8);
         Iterator<Double> it = suite.iterator();
         for (int i=0; i<14; i++)  System.out.println(it.next());
         for (double e : suite) System.out.println(e);
}
```

5 Généralisations

(1) Tentez de généraliser l'itérateur sur une feuille de salaire à un itérateur sur un objet quelconque. Analysez les difficultés que vous rencontrez.
(2) Peut-on associer plusieurs types d'itérateurs à une classe d'objets itérables ? Par exemple, essayez d'ajouter à votre précédent programme un itérateur de suite constante qui ne retourne que les valeurs de rang pairs ($u(0)$, $u(2)$, $u(4)$, ...) et un itérateur qui ne retourne que les valeurs de rang impairs ($u(1)$, $u(3)$, $u(5)$, ...) . Comment les utiliser conjointement dans un programme ?