Le patron *Proxy*

Exercice 1 – *Codage d'un* proxy virtuel

On s'intéresse à l'implémentation d'un navigateur web dans lequel les images incorporées dans les pages seront prises en charge par la classe suivante :

```
public class WebImage implements IWebImage {
        // url de l'image :
        private final String _URL;

        // Donnees factices:
        private int _data;

        public WebImage(String URL) {
                _URL=URL;

                // Telechargement de l'image (simulation) :
                try {
                Thread.sleep(1000);
                } catch (Exception e) {
                        System.out.println("Download error !");
                }
                _data=12345;
        }

        public void afficher(int length, int height) {
                // Simulation d'affichage :
                System.out.println("J'affiche "+_URL+" en "+length+"x"+height);
        }
}
```

Bien sûr cette classe ne fait que simuler le téléchargement des images par un delai d'attente.

Question 1.1 : Écrivez un programme principal instanciant un tableau de 30 images dont les URLs respectives seront "http ://the.web.site/image-*i*.jpg" (où *i* est le numéro de l'image) et appelle la méthode *afficher* (disons en *320x200*) de 5 d'entre-elles.

Question 1.2 : Exécutez le programme.

On constate que le temps de téléchargement nécessaire pour afficher seulement une partie de la page est très important. On souhaite pouvoir afficher efficacement des pages de grandes tailles. Pour cela, on se propose de ne télécharger les images qu'en fonction des parties affichées de la page (à la demande). La classe *WebImage* étant un composant à part entière, on ne souhaite pas la modifier. Un *proxy virtuel* a pour but prendre la place d'un objet (on dit que c'est un *placeholder*[1]), pour en différer l'instanciation coûteuse par exemple comme c'est le cas ici.

Question 1.3 : Bien que l'on ne souhaite pas modifier la classe *WebImage*, quelle petite modification suggéreriez-vous pour pouvoir lui appliquer correctement le patron de conception *proxy* (corrigeant le laxisme de son auteur) ?

Question 1.4 : Dessinez le schéma UML de ce patron proxy.

1. Ce terme n'a pas de traduction exacte en français : on dit souvent emplacement réservé tandis que la signification est plutôt "qui garde la place de".

Question 1.5 : Écrivez une classe proxy pour la classe *WebImage* qui diffère le téléchargement des images sur le web.
Question 1.6 : Modifiez et ré-exécutez le programme de test.

Exercice 2 – Codage d'un proxy copy-on-write

Soit la classe suivante :

```
public class Matrix {
        private int _sizem, _sizen;
        private int[][] _mat;

        // Constructeur initialisant une matrice n*m avec la valeur val
        public Matrix(int m, int n, int val) {
                _sizem=m;
                _sizen=n;
                _mat=new int[m][n];
        }

        // Fonction d'acces :
        public void set(int m, int n, int val) { _mat[m][n]=val; }
        public int get(int m, int n) { return _mat[m][n]; }

        // Constructeur de recopie :
        public Matrix(Matrix m) {
                _sizem=m._sizem;
                _sizen=m._sizen;
                _mat=new int[_sizem][_sizen];
                for (int i=0; i<_sizem; ++i)
                for (int j=0; j<_sizen; ++j) _mat[i][j]=m._mat[i][j];
        }
}
```

Question 2.1 : Completez le code de la classe *Matrix*.
Question 2.2 : Écrivez une classe proxy pour la classe *Matrix* qui évite les copies inutiles en utilisant le patron de proxy Copy-On-Write.