Master 1 TIIL Algorithmique et Programmation Objet - TP 2

Alain Plantec

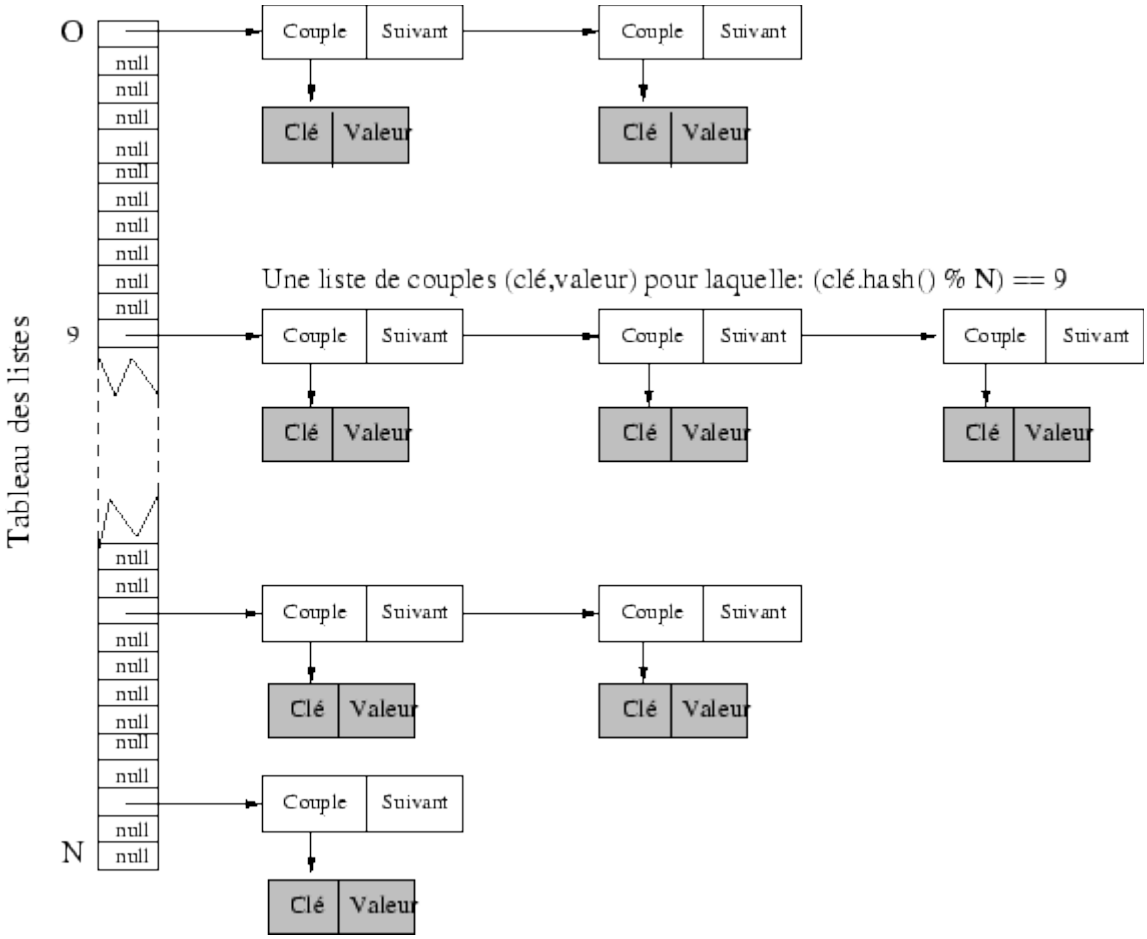
Préambule

La table de hash (rappels)

Une table de hash est une collection dans laquelle on stocke un ensemble de couples (clé,valeur). La clé et la valeur peuvent être de n'importe quelle classe. A une clé correspond une et une seule valeur. On ne peut pas trouver deux fois la même clé dans une même table de hash.

C'est une collection très intéressante pour stocker et retrouver très rapidement des données suivant des indexes autres que des indexes entiers.

Représentation schématique



On observe que le point d'entrée est un tableau de listes de taille *N*. Chaque élément de ce tableau est une liste. Les éléments stockés dans les listes sont des couples (clé,valeur). Pour une clé *k*, la position *p* dans le tableau de listes de taille *N* est calculée de la façon suivante :

$$p = k.hashCode()\%N$$

Deux clés différentes peuvent donc avoir la même valeur pour *p*.

Interface Map simplifiée

```
public interface Map {
    // creer un couple ('key','value') et le stocker dans la table
    Object put(Object key, Object value);
    // retrouver la valeur associee a la cle 'key'
    Object get(Object key);
    // supprimer le couple dont la cle est 'key'
    Object remove(Object key);
    // tester l'existance d'un couple dont la cle est 'key'
    boolean containsKey(Object key);
    // tester l'existance d'un couple dont la valeur est 'value'
    boolean containsValue(Object value);

    int size();
    boolean isEmpty();

    // la classe interne decrivant un couple
    public interface Entry {
        // retourner la cle
        Object getKey();
        // retourner la valeur
        Object getValue();
        // modifier la valeur
        Object setValue(Object value);
    }
}
```

Développement

Problème 1

Il est demandé de développer la classe *HashDict* qui met en oeuvre l'interface *Map*.

Problème 2

Il est demander de développer une visualisation pour votre classe HashDict à l'aide de Java 2D. Vous pouvez vous appuyer sur la représentation schématique données en préambule.

Vous trouverez une introduction à java 2D ici : Java : http://duj.developpez.com/tutoriels/java/dessin/intro/.

Voici un exemple de code pour dessiner un polygone dans une fenêtre.

```
import java.awt.Graphics;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.JComponent;
import javax.swing.JFrame;

class MyCanvas extends JComponent {
	private static final long serialVersionUID = 1L;
	public void paint(Graphics g) { // Fonction appelee pour dessiner
		int[] x = {10,120,120,10} ; // Tableau pour les x
		int[] y = {40,40,80,80} ; // Tableau pour les y
		int[] x2 = {30,150,150,30} ; // Tableau pour les x
		int[] y2 = {90,90,170,170} ; // Tableau pour les y
		Color previousColor = g.getColor();
		g.setColor(Color.BLACK);
		g.drawPolygon (x, y, x.length);     // Dessin du polygone
		g.fillPolygon (x2, y2, x.length);     // Dessin du polygone
		g.setColor(previousColor);
	}
}

class GestFenetre extends WindowAdapter {
	public GestFenetre(JFrame window) {
		window.addWindowListener(this);
	}
	public void windowClosing(WindowEvent e) {
		System.exit(0);
	}
}

public class DrawPolygon {
  public static void main(String[] args) { // Notre fonction main
```

```
    JFrame window = new JFrame(); // On cree une fenetre
    window.setBounds(30, 30, 300, 300); // On indique la taille de la fenetre
    window.getContentPane().add(new MyCanvas()); // On ajoute un composant pour dessiner
    window.setVisible(true); // Et hop, on demande à la fenetre de s'ouvrir
    new GestFenetre(window);
  }
}
```