FEKIH HASSEN Nadhem GL2/1

TP Java
hashCode(), equals() et les collections

equals():

Si on a besoin de savoir si deux objets sont réellement identiques (non leurs références) il faut utiliser la méthode equals().
La méthode equals() est :
Reflexive : x.equals(x) doit retourner true.
Symétrique : x.equals(y) retourne true (respectivement false) si et seulement si y.equals(x) retourne true (respectivement false).
Transitive : si x.equals(y) retourne true et y.equals(z) retourne true alors x.equals(z) doit retourner true.
Consistante : Si x et y restent fixes x.equals(y) retourne toujours le même resultat n'importe quel nombre de fois la méthode est invoquée .

hashCode():

le hashcode est utilisé pour créér un entier à partir d'un objet.
Si deux objets sont égaux selon la méthode equals() , ils doivent avoir le même hashcode.
De même deux objets différents ont des hashcodes distincts .
La méthode hashCode() appliquer plusieurs fois sur un même objet doit toujours retourner le même résultat.
Le hashCode() est utilisé pour augmenter la performance des larges collections de données. Les Collections comme HashSet et HashMap utilisent la valeur du hashCode d'un objet pour déterminer comment l'objet doit être trié dans la collection.

Les Collections :

Les Maps : HashMap, Hashtable, TreeMap, LinkedHashMap.
Les Sets : HashSet, LinkedHashSet, TreeSet.
Les Lists : ArrayList, Vector, LinkedList.
Les Queues : PriorityQueue.
Les Utilities : Collections, Arrays.

Lists : Liste des objets (classes qui implementent List).
Sets : Des objets uniques (classes qui implementent Set).
Maps : Les objets avec une unique ID (classes qui implementent Map).
Queues : Les objets sont disposés selon l'ordre dans lequel ils sont traités.

Méthodes importantes pour List(L), Set(S) et Map(M):
boolean add(element) : ajouter un élement (L,S).
boolean add(index, element) : ajouter un élement à un indice (L).
boolean contains(object) : rechercher un objet dans une collection (L,S,M).
boolean containsKey(object key) : rechercher une clé dans une collection (M).
boolean containsValue(object value) : rechercher une valeur dans une collection(M).
object get(index) : récupérer un objet selon l'index(L).
object get(key) : récupérer un objet selon la clé (M).
int indexOf(object) : récupérer la position d'un objet (L).
put(key, value) : ajouter une paire clé/valeur à une map(M).
remove(index) : supprimer un objet selon l'index (L).
remove(object) : supprimer un objet (L,S,M).
remove(key) : supprimer un objet selon la clé (M).
int size() : retourne la taille de la collection (L,S,M).
Object[] toArray() : retourne un tableau contenant les éléments de la collection(L,S).

Trier une liste :
static void sort(List)
static void sort(List, Comparator)

Recherche dans une liste triée :
static int binarySearch(List, key)
static int binarySearch(List, key, Comparator)

Pratique :

Utilisations de ArrayList .
Afficher et modifier un JTable à travers les boutons edit et apply.
Trier le JTable en selon la colonne sélectionnée , pour sélectionner une colonne on appuie sur son header .

JTablex.java:

```
import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.Dimension;
import java.util.*;
class JTableButtonRenderer implements TableCellRenderer {
        public Component getTableCellRendererComponent(JTable table, Object value,
boolean isSelected, boolean hasFocus, int row, int column) {
                JButton button = (JButton)value;
```

```
                return button;
        }
}

public class JTablex extends JFrame {
        private JTable table;
        JScrollPane scrollpane;
        JPanel pSouth;
        DefaultTableModel dataModel;
        private ArrayList<ArrayList<Object>>  valeurs = new
ArrayList<ArrayList<Object>>() ;
        JTextArea ta = new JTextArea("");
        JTextArea ta1 = new JTextArea("");
        JTextArea ta2 = new JTextArea("");
        JTextArea ta3 = new JTextArea("");
        ArrayList<Object> list ;
        ArrayList<Object> list1 ;
        ArrayList<Object> list2;
        ArrayList<Object> list3 ;
        ArrayList<Object> list4 ;
        ArrayList<Object> colonnesNom;
        JFrame frame = new JFrame();
        public JTablex() {
                ArrayList<Object> list = new ArrayList<Object>();
                ArrayList<Object> list1 = new ArrayList<Object>();
                ArrayList<Object> list2 = new ArrayList<Object>();
                ArrayList<Object> list3 = new ArrayList<Object>();
                ArrayList<Object> list4 = new ArrayList<Object>();
                JButton edit = new JButton("Edit");
                list.add(new Integer(15));
                list.add("Ahmed");
                list.add("Salah");
                list.add("27-04-1995");
                list.add(edit);
                valeurs.add(list);
                list1.add(new Integer(20));
                list1.add("Salah");
                list1.add("Lakhdher");
                list1.add("15-16-2005");
                list1.add(edit);
                valeurs.add(list1);
                list2.add(new Integer(25));
                list2.add("Ali");
                list2.add("Khdhiri");
                list2.add("12-02-2000");
                list2.add(edit);
```

```
                valeurs.add(list2);
                list3.add(new Integer(30));
                list3.add("Mohamed");
                list3.add("Neji");
                list3.add("13-07-1999");
                list3.add(edit);
                valeurs.add(list3);
                list4.add(new Integer(25));
                list4.add("Imen");
                list4.add("Souissi");
                list4.add("29-05-2003");
                list4.add(edit);
                valeurs.add(list4);

                final ArrayList<Object> colonnesNom = new ArrayList<Object>();
                colonnesNom.add("Cins"); colonnesNom.add("Prénom");
colonnesNom.add("Nom"); colonnesNom.add("DN");
                colonnesNom.add("Edit_Button");
                dataModel = new DefaultTableModel() {
                        public int getColumnCount() {return colonnesNom.size();}
                        public int getRowCount() {return valeurs.size();}
                        public Object getValueAt(int l, int c)
                        {return valeurs.get(l).get(c);}
                        public void setValueAt(Object val, int l, int c)
                        {valeurs.get(l).set(c, val);}
                        public String getColumnName(int col)
                        {return (String)colonnesNom.get(col);}
                        public Class getColumnClass(int cl)
                        {return getValueAt(0, cl).getClass();}
                };// fin de définition de AbstractTableModel
                table = new JTable(dataModel);
                table.setEnabled(true);
                scrollpane = new JScrollPane(table);
                TableCellRenderer buttonRenderer = new JTableButtonRenderer();
                table.getColumn("Edit_Button").setCellRenderer(buttonRenderer);
                scrollpane.setPreferredSize(new Dimension(500, 300));
                this.getContentPane().add(scrollpane);
                JSplitPane sp1 = new JSplitPane();
                pSouth = new JPanel();
                pSouth.setLayout(new GridLayout(2, 1));
                JPanel panel = new JPanel();
                JPanel panel1 = new JPanel();
                JButton apply = new JButton("Apply");
                JButton cancel = new JButton("Cancel");
                panel.setLayout(new GridLayout(4, 4, 10, 20));
                TableRowSorter<TableModel> sorter = new
```

```
TableRowSorter<TableModel>(table.getModel());
                table.setRowSorter(sorter);
                panel.add(new JLabel("Cin"));
                ta = new JTextArea("");
                ta1 = new JTextArea("");
                ta2 = new JTextArea("");
                ta3 = new JTextArea("");
                panel.add(ta);
                panel.add(new JLabel("Nom"));
                panel.add(ta1);
                panel.add(new JLabel("Prénom"));
                panel.add(ta2);
                panel.add(new JLabel("DN"));
                panel.add(ta3);
                panel1.add(apply); panel1.add(cancel);
                pSouth.add(panel);
                pSouth.add(panel1);
                table.addMouseListener(new MouseAdapter() {
                        public void mouseClicked(MouseEvent e) {
                            int x = table.getSelectedRow();
                            if (table.getSelectedColumn() == 4) {
                            ta.setText((dataModel.getValueAt(x, 0)).toString());
                            ta1.setText((dataModel.getValueAt(x, 1)).toString());
                            ta2.setText((dataModel.getValueAt(x, 2)).toString());
                            ta3.setText((dataModel.getValueAt(x, 3)).toString());
                                }
                        }
                });
                apply.addActionListener(x2);
                cancel.addActionListener(x1);
                sp1.setTopComponent(scrollpane); sp1.setBottomComponent(pSouth);
                sp1.setDividerSize(7);
                sp1.setOrientation(JSplitPane.VERTICAL_SPLIT);
                this.getContentPane().add(sp1, BorderLayout.CENTER);
                this.setBounds(10, 10, 500, 400);
                this.pack();
                this.setVisible(true);
        }
        private ActionListener x2 = new ActionListener() {
                public void actionPerformed (ActionEvent e) {
                        int i = Integer.parseInt(ta.getText());
                        int x = table.getSelectedRow();
                        dataModel.setValueAt(i, x, 0);
                        dataModel.setValueAt(ta1.getText(), x, 1);
                        dataModel.setValueAt(ta2.getText(), x, 2);
                        dataModel.setValueAt(ta3.getText(), x, 3);
```

```
                        dataModel.fireTableDataChanged();
                        ta1.setText("" + '\u0000');
                        ta2.setText("" + '\u0000');
                        ta3.setText("" + '\u0000');
                        ta.setText("" + '\u0000');
                }
        };
        private ActionListener x1 = new ActionListener() {
                public void actionPerformed (ActionEvent e) {
                        ta1.setText("" + '\u0000');
                        ta2.setText("" + '\u0000');
                        ta3.setText("" + '\u0000');
                        ta.setText("" + '\u0000');
                }
        };

        public static void main(String[] args) {
                new JTablex();
        };
}
```