CLASE JTABLE

Vamos a explicar los JTable con 3 ejemplos (buscadlos):

1. EjemploJTable1.
2. EjemploJTable3
3. EjJTablePR5

1. Comentarios sobre EjemploJTable1: En este ejemplo utilizamos el constructor más sencillo:

   JTable(int numRows, int numColumns),

   Éste utiliza *DefaultTableModel* que es una implementación por defecto de un JTable. Nos ofrece un JTable vacío y editable. Observar, como os dejé comentado en el ejemplo, los distintos modos de selección y el ajuste del tamaño de las columnas.

2. Comentarios sobre EjemploJTable3: Creación de un JTable con datos de un array bidimensional, utiliza el constructor:

   JTable(Object[][] rowData, Object[] columnNames)

   Así, construimos un jTable cuando los datos nos vienen en un array bidimensional. Pero, esto no es suficiente pues los datos no siempre vienen en un array bidimensional o necesitamos que nuestra tabla admita más operaciones que las que vienen en *DefaultTableModel*.

   Observar en este ejemplo el método *getSource()*, nos devuelve el componente que ha originado el evento.

3. Comentarios ejJtablePR5: En este ejemplo se ve un JTable que muestra los datos de un arraylist, que es exactamente lo que tenéis que hacer vosotros en la PR2, pues tenéis que mostrar en un JTable la información de vuestros arrayList.

   En este ejemplo tenemos:

   -EventsPrueba: Representa a un evento con un tiempo y una prioridad

   -EventsTableModel: mi modelo de tabla para visualizar los datos desde un arrayList (después lo explicamos despacio)

   -EjJtablePR5: Es una ventana que contiene un botón y un JTable que al hacer clic en el botón, nos muestra la información de los eventos de un arrayList de eventos.

   Para crear un modelo de tabla hay que extender a *AbstractTableModel*, los siguientes métodos hay que sobreescribirlos obligatoriamente, si no, no compila:

   -public int getColumnCount();

   -public int getRowCount();

   -public object getValueAt(int rowIndex, int columnIndex),

Un Jtable es una representación visual de unos datos que están en una estructura de datos. El método *getValueAt(int rowIndex, int columnIndex)* indica cómo se va a cargar la tabla desde nuestra estructura de datos. Es decir, lo que vamos a ver en la posición (i,j) de nuestra tabla.

En este caso, la fila i-ésima muestra la información del evento i-ésimo de nuestro arrayList. La primera columna la utilizamos para enumerarlos.

El resto de métodos si no se sobreescriben utiliza una implementación por defecto, por ejemplo, en el método *getColumnName*, si no se sobreescribe , los nombres de las columnas serán A,B,C…

Como he dicho, un JTable es una representación visual de una estructura de datos, si la estructura de datos cambia hay que refrescar la tabla. En nuestro ejemplo refresca con el método *fireTableStructureChanged().* Ver en las transparencias más métodos para refrescar la tabla.