Working with a JTable

There are four main steps involved in creating and populating a JTable. These steps are:

1. Place a JScrollPane on the UI (a JFrame), then place a JTable in the JScrollPane container. If you just put a JTable on the JFrame (and not in a JScrollPane), you will not be able to display the headings.

2. Using the Properties window for the JTable, select "new StringTableModel" from the Model property. Give this new model a name – my example is named "schedModel".

3. In the JFrame's windowOpened event, set the columns headings by using the setColumnHeaders() method. You pass this method a comma-delimited string.[1] The following shows an example (assumes model is named "schedModel"):

```
void JFrame1_windowOpened(java.awt.event.WindowEvent event)
{
    schedModel.setColumnHeaders("Month,Payment,Balance,Principal,Interest");
}
```

4. To fill the table, you need to create an array. The array can be any type but since the data placed in the JTable must be String type, it makes sense to use a String array. One strategy is to create a one-dimensional array with each element representing one row in the JTable. Within each element, a delimiter (such as a comma) would separate the column values. For example, consider the following array:

```
a[0] = "12,34,55"
a[1] = "75,86,10"
a[2] = "22,33,44"
a[3] = "50,60,70"
```

   When this array is placed into the JTable (using the setItems() method), the result will be four rows and three columns.

   If you are calculating numeric values that are going to be placed into the JTable, you need to convert them to Strings. The Format() method is ideal for this because it not only converts the numeric values to a String, but it also formats them according to the formatting rule(s) you specify.

---

[1] The commas delimit each heading. That is, if you want your headings to be "Column 1" and "Column 2", you would pass the event the string "Column 1,Column 2". If your heading includes a comma, e.g., "Last, First" and "Department", then you have to use an overloaded version of the setColumnHeaders() method. The alternative passes a second string that defines your delimiter. For example, you could do the following:

```
schedModel.setColumnHeaders("Last, First#Department", "#");
```

In this version, the "#" is the delimiter.

In the following code segment, the String array "outString" is created with one element for each month for the number of years specified by the variable "years". Within each element, five values are concatenated and delimited with the "#" character. Note that the default comma delimiter would not work in this example because the getCurrencyInstance() method places commas in large numbers. These commas would be treated as delimiters in a comma-delimited string. When the loop terminates, the array is passed as an argument to the StringTableModel's setItems() method which then fills the JTable.

```
String outString [] = new String[years*12];

for (int month = 1; month <= years * 12; month++)
{
   monthlyInterest = balance * rate/12;
   monthlyPrincipal = payment - monthlyInterest;
   balance = balance - monthlyPrincipal;

   String strPayment =
         NumberFormat.getCurrencyInstance().format(payment);
   String strInterest =
         NumberFormat.getCurrencyInstance().format(monthlyInterest);
   String strPrincipal =
         NumberFormat.getCurrencyInstance().format(monthlyPrincipal);
   String strBalance =
         NumberFormat.getCurrencyInstance().format(balance);

   outString[month-1] = Integer.toString(month) + "#" +
         strPayment + "#" + strBalance + "#" +
         strPrincipal + "#" + strInterest;
}
schedModel.setItems(outString,"#");
```

Useful JTable Methods

The following methods are generally helpful when dealing with tables. Refer to the API reference for more information.

```
setSelectionMode(int selectionMode)
sizeColumnsToFit(boolean lastColumnOnly)
getSelectedRow()
clearSelection()
```