Strayer University

**Bar Chart Java**

Week 8

**Case Study**

for the

Course of

Java programming 2

02/27/17

By

Kennedy Kabaso.

Professor: Srinath Mirajkar

[**Teaching software componentization: A bar chart Java bean**](https://blackboard.strayer.edu/bbcswebdav/institution/CIS/407/1144/Week8/CIS407_week8_CaseStudy.pdf)

**Summarize the main points of the article.**

This article was about the current object-oriented paradigm, software construction increasingly involves creating and utilizing software components. The follow topics were explained such the pedagogical benefits of implementing and using java-based graphical software components, the description of the bar chart component and its use, the programming tasks for implementing the bar chart component and finally the programming tasks for utilizing the bar chart component.

Modern-day software applications can be characterized as assemblages of portable software components. This has led to a new approach to software development, often called Component Based Development (CBD). Learning how to use these components involves instantiation, placement in forms and containers, viewing and manipulating their properties, invoking behaviors via method calls, and responding to events generated by the components. These are the skill that most students get comprehensive training and practice in during their programming coursework.

Most college-level computer programming classes in information systems curricula go into a fair amount of detail in building applications that use software components.

In the Java world, graphical software components are implemented as JavaBeans (Liang 2009 pp1049-1064). JavaBeans are useful for rapid application development (RAD), because they provide developers with ready-made modular functional units that can typically be embedded in Java applications through an Integrated Development Environment’s (IDE) form-building design tools.

From a pedagogical perspective, learning how to develop and use JavaBeans (or bean-like components) offers students important software development skills in object-orientation, event-driven software architectures, encapsulation and data hiding, and object persistence via serialization. Students involved in graphical programming must learn about the XY-coordinate system, RGB color control, and the mathematical formulas that are required to translate numerical data into visualization results.

The bar chart JavaBean component discussed in this paper is a graphical component that can be embedded in Java applications, either through instantiation in the code, or by dragging onto a form in an IDE’s form builder. The BarChartPanel class is a subclass of JPanel, so this is a GUI component that can be placed into another Java Swing container, such as a frame, an applet, or another panel. The component requires two arrays for input data: an array of double (floating point) values and an array of strings (text). The strings form the labels of the bars, and the numeric values will be used to determine bar heights.

Bar charts as data visualizations tie in very well with database query skills. In particular, aggregate SQL queries involving sums or averages and including grouping clauses are excellent data sources for the BarChartPanel. In short, the BarChartPanel, once developed, can be useful for a wide variety of purposes. Once students build the BarChartPanel itself, they can then develop applications that make use of this component. This component has also been used for several years by CIS majors working on the capstone project in an advanced core CIS course, a project that involves designing and implementing a database, performing the interviewing tasks, use case modeling and user interface design for an application, and ultimately implementation and documentation of a finished software product.

Actually, in the implementation I use for my programming assignments, the BarChartPanel does not satisfy all the requirements for a component to be a true JavaBean. In particular, it does not actually have an associated Event class. Strictly speaking, the BarChartPanel does not generate an Event (i.e. an instance of a subclass of Java’s Event class). All it really gives is a number, which is the index value of the bar that was clicked or dragged. For a student assigned to create the BarChartPanel, the programming task requires three fairly advanced skills: (1) listener registration and notification, (2) geometric analysis and data mapping, and (3) bar-selection and listener notification based on mouse event processing.

Once students have developed their BarChartPanel classes, they can begin to use it in subsequent programming assignments. At this point, the BarChartPanel becomes a useful pedagogical tool for teaching about data visualization. In particular, programming projects can make use of JDBC for database connectivity and incorporate SQL aggregate queries to produce result sets that can be mapped into the BarChartPanel.

In addition to the standard database processing functions, such as scrolling forward and backward through the employee table, displaying data from these queries, identifying projects that an employee is working on, and supporting updates to the database, this application performs aggregate join queries in order to obtain statistics from the database. These aggregate queries return result sets of (a) employee counts per department (requiring a two-table joint combined with grouped aggregation), and (b) employee counts per project (requiring a three-table joint combined with grouped aggregation). The application then instantiates BarChartPanel objects to display the information, such as shown above. The application also listens for events generated by the BarChartPanel, so that when a user clicks on a bar representing a department or a project in the chart, the application retrieves the selected bar’s label and uses it in a query to obtain and display drill-down detail about the specific employees in that project or department.

This component has also been used for several years by CIS majors working on the capstone project in an advanced core CIS course, a project that involves designing and implementing a database, performing the interviewing tasks, use case modeling and user interface design for an application, and ultimately implementation and documentation of a finished software product.

Actually, in the implementation I use for my programming assignments, the BarChartPanel does not satisfy all the requirements for a component to be a true JavaBean. In particular, it does not actually have an associated Event class. Strictly speaking, the BarChartPanel does not generate an Event (i.e. an instance of a subclass of Java’s Event class). All it really gives is a number, which is the index value of the bar that was clicked or dragged. From a design perspective, assuming that the only important piece of information is the Event object around it.

For a student assigned to create the BarChartPanel, the programming task requires three fairly advanced skills: (1) listener registration and notification, (2) geometric analysis and data mapping, and (3) bar-selection and listener notification based on mouse event processing.

Once students have developed their BarChartPanel classes, they can begin to use it in subsequent programming assignments. At this point, the BarChartPanel becomes a useful pedagogical tool for teaching about data visualization. In particular, programming projects can make use of JDBC for database connectivity and incorporate SQL aggregate queries to produce result sets that can be mapped into the BarChartPanel.

The BarChartPanel component and its use in Java applications provide many pedagogical benefits for programming students. By building a graphical component students learn about elements of graphical programming, including geometric analysis and color control. By coding the component to generate events, students learn how to manage listener registration and notification, thereby building on their understanding of object-oriented principles such as inheritance and polymorphism. By using this component, students are able to gain an understanding of the mapping from aggregate SQL queries to data visualizations, as well as getting practical experience with drill-down capabilities that enable applications to go from summary (aggregate) information to more specific details at the click of a button

**Bibliography**

* Teaching Software Componentization: A Bar Chart Java Bean by Michel Mitri, CIS & MS Department, James Madison University Harrisonburg, VA 22801, USA

1. **Describe one (1) way in which one could apply Java graphics in education. Provide one (1) example of such use to support your response.**

Java graphics can be used for constructing Graphical User Interface (GUI) applications using two types of GUI elements such as Components, which are elementary GUI entities like Button, Label, and TextField, and the Containers like Frame and Panel which are used to hold components in a specific layout.

1. **Analyze at least three (3) commands that you would use in order to draw a bar chart. Provide a rationale for your response.**

If you want to build a bar chart in your Java application, you can create two axes, then instantiate the Bar Char class, then define the series of data, and assign the data to the chart. Then you can set the color, fill the shape and draw the shape like rectangle which will can be done as follows below using this command such as setColor(), fillRect() and drawRect() which will do as explained.

Example given: It will set the color of the rectangle to a black color and draw it with the width and the height which is specified

g.fillRect(valueP, valueQ, barWidth - 2, height);

g.setColor(Color.black);

g.drawRect(valueP, valueQ, barWidth - 2, height);

1. **Suggest one (1) way in which you would use to modify a bar chart in order to update it for new information. Justify your response.**

When you update the underlying data of a chart, the chart's data and appearance also changes. The extent of those changes depends on how you define the chart's data source. If you add more data to the chart, its height will increase because it is proportional to the increase in data.