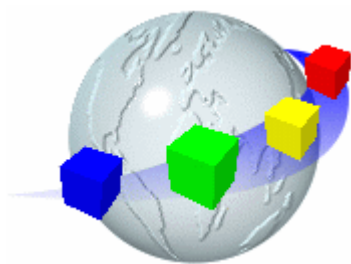


WebCab JGraph v2.1

WebCab
Components



Preface

JGraph is a suite of JavaBeans for the graphical display of discrete data sets. This documentation accompanies JGraph and offers advice and guidance on its use, functionality and customization.

Within this documentation we provide a detailed description of the component packages content, functionality and applicability. We also include an overview of the JavaBeans technology, its installation and how these JavaBeans can be easily customized with the use of any Java IDE. A Programmer's guide is provided which offers a road map for developers wishing to take advantage of every feature and functionality of this component. The fourth chapter contains UML diagrams of each of the four types of charts (i.e. chart, bean, pie and pictogram). In the next chapter screen shots and further explanation is provided which illustrate the end-user experience. Finally, we introduce WebCab Components, its philosophy and approach to serving the JavaTM development community with robust and powerful J2EE applications.

If you have any suggestions for additional functionality, which we could include into future development cycles or any other queries then please feel free to contact us at:

support@WebCabComponents.com

Good luck with your project and thank you for your interest in our components.

The WebCab Components Team

Contents

Preface	i
1 Introduction	1
1.1 Product Description	1
1.1.1 Overview	1
1.1.2 Details	1
1.2 Prerequisites and Compatibility	4
1.2.1 System Requirements	4
1.2.2 Compatibility	4
2 Graphical Display of Data	5
2.1 Introducing some concepts and terminology	5
2.1.1 Nominal, Ordinal and Interval data types	5
2.1.2 Central Tendency, Dispersion, Skew and Kurtosis	6
2.2 Types of Charts	6
2.2.1 Bar Charts	6
2.2.2 Frequency Polygons	7
2.2.3 Histograms	8
2.2.4 Line Graph	9
2.2.5 Ogives (Cumulative Distribution Curve)	9
2.2.6 Pie Charts	10
2.2.7 Stacked Bar Charts	11
2.2.8 Conclusions	12
3 JavaBeans™ Guide	14
3.1 Overview	14
3.2 Installing the JavaBeans	14
3.3 Using JavaBeans inside an IDE	15
3.3.1 Importing the <i>JAR</i> file into Sun's One IDE	15
3.3.2 Importing the <i>JAR</i> file into Borland's JBuilder IDE	16
3.3.3 Create a JavaBean charting client using JBuilder in 20 Steps	17
3.4 Developer Support	37
3.4.1 Documentation	37
3.4.2 Compilation and Custom Modification of Source Code	37

3.4.3	Online Support	37
4	Programmer's Guide	38
4.1	Introduction	38
4.2	How to import data	38
4.2.1	Loading data from a Database	38
4.3	Customizing JGraph	39
4.3.1	Customizing the Chart component	40
4.3.2	Customizing the Pie component	42
4.3.3	Customizing the Graph component	44
4.3.4	Customizing the Pictogram component	47
4.4	Developer's Support	48
4.4.1	Compilation and Custom Modification of Source Code	48
4.4.2	Online	48
5	JGraph UML Diagrams	49
5.1	Graph UML diagram	50
5.2	Chart UML Diagram	52
5.3	Pie UML Diagram	54
5.4	Pictogram UML Diagram	56
6	JGraph Screen Shots	57
6.1	The Chart Component	57
6.2	The Graph Component	64
6.3	The Pie Component	68
6.4	The Pictogram Component	70
7	Guide to WebCab Components	71
7.1	The Company	71
7.2	Presentation of Products	71
7.3	Supported Clients, IDEs, Containers and DBMSs	71
7.4	Transparent Functionality	72
7.5	Code Conventions	72
7.6	Company Culture and Activity	72
7.7	Product Life cycle	72
7.8	Support, Warranty and Upgrades	73

Chapter 1

Introduction

1.1 Product Description

1.1.1 Overview

Combine the power of the JavaBeansTM technology with JGraph's ability to visually represent data by adding powerful graphing and charting capabilities to your applications. Using JGraph inside an IDE or by direct implementation is easy and intuitive to use and gives you the possibility to represent data as Charts, Graphs and Pictograms. This component offers many features, from the simple possibility of choosing the colors and fonts, to powerful features like drag and zoom.

1.1.2 Details

The following features are supported:

- **The Graph Component:**
 - **Types of Graphs** - several interpolation methods can be used (e.g. cubic spline, linear interpolation).
 - **Multiple Series** - display an unlimited number of curves simultaneously.
 - **JDBC compliant** - connect via JDBC drivers to any JDBC compliant DBMS (e.g. Oracle, DB2).
 - **Zooming** - zooming is possible by specifying the zoom factor.
 - **Scaling** - the graph can be scaled in both the vertical and horizontal directions.
 - **Centering** - the visual display of the graph can be centered on a specified point.
 - **Saving as JPEG** - the chart can be exported as a JPEG with selectable image quality.
 - **Anti-aliasing** - anti-aliasing capabilities can be used when drawing the graphs. This makes all the curves and inclined lines look smoother by reducing the stair effect.

- **Dragging** - the position of each interpolation point can be modified by dragging the point with the mouse. The whole graph can be dragged which is useful when the graph exceeds the visible area.
 - **Tool-Tip** - the current mouse position can be shown as a tool-tip.
 - **Background Image** - the background can be changed. It can be either a solid color or an image (JPEG, GIF or PNG) URL link.
 - **Grid** - for easy reading a grid can be added. This is helpful if you want to observe how the function evolves on a specific region. It can also be used for “snap to grid” operations when you change the position of one interpolation point. The thickness and the color of the grid can be modified.
 - **Fonts and Colors** - all the colors inside the Graph component can be changed (i.e. graph, labels, axis, points, values and text). The user can select any one of the standard JavaTM fonts to display text.
- **The Chart Component:**
 - **Types of Charts** - Several types of charts are available (e.g. bar and pie).
 - **Multiple Series** - display multiple data series simultaneously.
 - **JDBC compliant** - connect via JDBC drivers to any JDBC compliant DBMS (e.g. Oracle, DB2).
 - **Anti-aliasing** - anti-aliasing capabilities can be used when drawing the charts. This makes all the curves and inclined lines look smoother by reducing the stair effect.
 - **Light Effects** - you may add light effects to 3D perspective charts.
 - **Transparency** - the chart can be varied between different levels of transparency, from being opaque to totally transparent (invisible).
 - **Shadows** - a shadow effect can be added to the charts.
 - **Background Image** - the background of the chart can be changed. It can be either a solid color or an image (JPEG, GIF or PNG) URL link.
 - **Fonts and Colors** - all the colors and fonts inside the Chart component can be changed.
 - **Background Grids** - Grids help you to estimate more easily coordinates on the screen.
 - **Saving as JPEG** - the chart can be exported as a JPEG with selectable image quality.
 - **Labels** - floating labels can be added to the chart.

- **The Pie Component:**

- **JDBC compliant** - connects via JDBC drivers to any JDBC compliant DBMS.
- **Anti-aliasing** - anti-aliasing capabilities can be used when drawing the charts. This makes all the curves and inclined lines look smoother by reducing the stair effect.
- **Light Effects** - you may add a light effect to 3D perspective charts.
- **Transparency** - the chart can be smoothly varied between different levels of transparency, from being opaque to totally transparent.
- **Background Image** - the background of the chart can be changed. It can be either a solid color or an image (JPEG, GIF or PNG) URL link.
- **Fonts and Colors** - colors and fonts inside the chart can be changed.
- **Background Grids** - Grids help you to estimate more easily coordinates on the screen.
- **Sorting** - the data can be sorted according to several criteria.
- **Saving as JPEG** - the chart can be exported as a JPEG with selectable image quality.
- **Labels** - floating labels can be added to the chart.
- **Rotation** - Pie charts can be rotated by dragging them while pressing the SHIFT button.

- **The Pictogram Component:**

- **JDBC compliant** - connect via JDBC drivers to any JDBC compliant DBMS.
- **Supported Images** - supports GIF, JPEG and PNG image formats which are used for drawing the chart.
- **Values** - the key (scale) of the image can be modified. This is the amount of data represented by one image.
- **Background, Fonts and colors** - the text supports standard Java fonts and colors (background's color can be changed).
- **Saving as JPEG** - the chart can be exported as a JPEG with selectable image quality.

1.2 Prerequisites and Compatibility

1.2.1 System Requirements

- An Operating System running Java™
- Pentium® III 733 MHz
- 256MB RAM

1.2.2 Compatibility

Operating System for Deployment

- Windows XP, 2000, NT, 9x
- Sun Solaris
- Linux
- IBM AIX
- HP-UX

Component Type

- JavaBeans™

Built Using

- Java™ 2 SDK Standard Edition 1.3.1

Compatible IDE's

- Borland JBuilder
- Eclipse
- Sun's ONE IDE (formerly Forte for Java)
- Oracle JDeveloper

Chapter 2

Graphical Display of Data

Graphs are pictorial representations of the relationship between two (or more) variables are an important integral part of descriptive statistics. Different types of graphs can be used for illustration purposes depending on the types of variable (nominal, ordinal or interval; we explain the meaning of these terms below) and the issues of interest.

Within this section we detail some of the types of graphs and related techniques which we have implemented within this component. Graphs in there various forms are used in order to visually summarize a proposed relationship between variables, particularly when the data sets being considered are large or unmanageable. Graphs can appeal to visual memory in ways that mere tables or frequency distributions cannot. However, if not used carefully, graphs can misrepresent relationships between variables or encourage inaccurate conclusions.

2.1 Introducing some concepts and terminology

Within this section we explain some notions and terminology which we use below in the classification of the differing approaches to the visual representing data. The reader may wish to jump to the following section or refer to the definitions contained within this section as and when needed.

2.1.1 Nominal, Ordinal and Interval data types

The terms nominal, ordinal and interval data types are widely used within data analysis and broadly means the following:

- **Nominal** - If binary data can be thought of as two-valued variables, then nominal data can be expressed as n-valued variables.
- **Ordinal** - Closely related to nominal data, ordinal data is multi-valued with an ordering relationship.

- **Interval** - a variable which groups values according to a finite number of collections (.e.g. a day can be divided in to 24 hour intervals)

2.1.2 Central Tendency, Dispersion, Skew and Kurtosis

Central tendency (CT) and Dispersion are inter-related since the CT is a means to evaluate where the ‘center’ of a data set is. Whereas the dispersion measure how closely the members of the data set are ‘bunched’ around this center. More formally:

- **Central Tendency** - Measures of central tendency are measures of the location of the middle or the center of a distribution. The arithmetic mean, median and mode are the three most commonly used measure of central tendency.
- **Dispersion** - Dispersion (or variability) measures indicate how spread out a set of data is. The variance (the arithmetic average of the sum of the squares of the distance from the mean) and its square root, the standard deviation of popular measures of dispersion.

The Skew and Kurtosis are used to further describe the “shape” of a data set. That is, two data sets which have the same mean and variance may have line graphs which have different qualitative properties. The Skew can be thought of as measuring the symmetry of the data set (or the corresponding line graph) and the Kurtosis can be thought of as measuring the degree to which values take values “far” from the mean (i.e. whether the distribution is flat or thin tailed).

The Skew is defined as the standardized third moment about the mean (with the variance being the second moment). For example, the Normal distribution has a Skew of zero; distributions where the left (resp right) tail is longer than the right (resp. left) tail has a negative (resp. positive) skew. The Kurtosis of the normal distribution is 3, such distributions are referred to as Mesokurtic distributions. When the Kurtosis is less than 3 the distribution is describe as Plattykurtic and when it is greater than 3, it is described as Leptokurtic.

2.2 Types of Charts

He we will briefly give an overview of the various types of charts implemented, there corresponding strengths and weaknesses and how they can be used to best represent and possibly clarify conjectures concerning the nature of a given data set.

2.2.1 Bar Charts

Description and use Bar charts are often used to show the number or proportion of nominal or ordinal data which possess a particular attribute. Bar graphs are generally used to to represent the number of observations in a given category. Bar charts can be used to show the proportions between different categories, as opposed to the more commonly used pie chart.

Advantages

- Show each nominal and ordinal category in a frequency distribution
- Display relative numbers or proportions of multiple categories
- Summarize a large data set in visual form
- Clarify trends better than tables or arrays
- Estimate key values at a glance
- Permit a visual check of the accuracy and reasonableness of calculations
- Easily understood due to widespread use in business and the media

Disadvantages

- Requires additional written or verbal explanation
- Easily manipulated to yield false impressions
- Inadequate to describe the attribute, behavior, or condition of interest
- Fails to reveal key assumptions, norms, causes, effects, or patterns

2.2.2 Frequency Polygons**Description and use**

Frequency polygons are the preferred way to graph the frequency distribution of “un-grouped” (raw) interval data. They represent the frequency of each class of data points as a line connecting the midpoints of the bars of a histogram. Frequency polygons can describe the behavior of the same interval variable under different circumstances.

Advantages

- Can display central tendency, dispersion, and clustering
- Estimate key values, particularly the mean, and show skew and kurtosis
- Summarize a large data set in visual form
- Clarifies trends more clearly than most other forms of graphs, tables or arrays
- Show the behavior and distribution of a variable
- Smoother characteristic as more data points are added
- Widely used in business and the media

Disadvantages

- Fails to delineate each interval in a frequency distribution
- Cannot observe details concerning relative values and proportions
- Require additional written or verbal communication
- Not sufficient to describe the attribute, behavior, or condition of interest
- Fails to reveal the key assumptions, norms, causes, effects or patterns
- Easily manipulated to get false impressions
- Unable to act as a visual check of the accuracy or reasonableness of calculations

2.2.3 Histograms

Description and use Histograms are the preferred method for displaying grouped interval data. They depict the number or proportion of data points falling into a given class. While bar charts and histograms have a similar appearance, histograms can display the continuous classes of data rather than discrete categories found in bar charts. Since histograms display continuous classes it is the convention to not have any space between the bars of a histogram.

Advantages

- Can show the central tendency and dispersion of a data set
- Shows each interval in the frequency distribution
- Summarize a large data set in a visual form
- Clarify trends more clearly than either tables or arrays
- Can be used to estimate the key values at a glance
- Allows a visual check of the accuracy and reasonableness of calculations
- Easily understood due to widespread use in business and the media
- The bar within the histogram reflect the proportion of data points in each class

Disadvantages

- Requires additional written or verbal explanation
- Easily manipulated to give false impressions
- Inadequate to describe the attribute, behavior, or condition of interest
- Fails to reveal key assumptions, norms, cause, effects or patterns

2.2.4 Line Graph

Description and use

Lines graphs use a single line to connect plotted points of an interval. Since such graphs are most commonly used to visually represent trends over time, they are sometimes referred to as time-series charts. We deal further will topics related direct to time series within the corresponding chapter.

Advantages

- Clarify patterns and trends over time better than most other graphs
- Be visually simpler than bar graphs or histograms
- Summarize a large data set in a visual form
- Smooths as data points and categories are added
- Easily understood due to widespread use in business and the media
- Require additional written or verbal explanation

Disadvantages

- May be inadequate to describe the attribute, behavior, or the condition of interest
- Fails to reveal key assumptions, norms, or causes in the data
- Can be easily manipulated the give false impressions
- Reveals little concerning key descriptive statistics such as skew and kurtosis
- Fail to provide a check of the accuracy or reasonableness of calculations

2.2.5 Ogives (Cumulative Distribution Curve)

Description and use

Ogives are a type of frequency polygon used to depict the cumulative number or cumulative proportion of grouped interval data. They represent the cumulative frequency of each class of data points as a line connecting the upper limit of each class. Ogives let us make such statements as “X people sampled earned below \$Y”.

Advantages

- Visually approximate raw data
- Summarize a large data set in a visual form
- Delineate each interval in the frequency distribution
- Observe rates of change between classes better than other graphs
- Provide a visual check of the accuracy or reasonableness of calculations
- Smoothing of the graph as data points or classes are added
- Shows the number or proportion of the data points above or below a particular value
- Easily understood due to widespread use in business and the media

Disadvantages

- Difficult to explain the nature of the graph to others
- Fail to reflect the data points in a data set
- Reveal little about central tendency, dispersion, skew, or kurtosis
- Often require additional written or verbal explanation
- Inadequate to describe the attribute, behavior, or condition of interest
- Fail to reveal key assumptions, norms, causes or effects
- Fairly easily manipulated to produce false impressions

2.2.6 Pie Charts**Description of use**

Pie Charts are circles or cylindrical shapes which are subdivided into a number of “slices”. The area of each slice represents the relative proportion of data points which fall into a given category. Pie charts are the preferred method for graphing both nominal data and percentages.

Advantages

- Display relative proportions of multiple classes of data
- Show areas proportional to the number of data point in each category
- Summarize a large data set in visual form
- Visually easy to interpret
- Permit a visual check of the reasonableness or accuracy of the calculations
- Require minimal additional written or verbal explanation
- Easily understood due to widespread use in business and the media

Disadvantages

- Reveals little concerning central tendency, dispersion, skew, or kurtosis
- Fails to reveal key assumptions, norms, causes, effects, or patterns
- Fails to describe the attribute, behavior, or condition of interest
- Easily manipulated to give false impressions

2.2.7 Stacked Bar Charts**Description of use**

Stacked bar graphs (component bar graphs) add information to standard bar graphs by representing multiple categories of data in a single bar. Each bar is divided into sections whose heights indicate the proportions of observations falling into a given category. Since they deal with categories, these graphs are generally used for nominal data.

Stacked bar graphs are useful for comparing the contributions of different groups to the total value of a variable. They can also compare two identical nominal data sets under different conditions, especially at different times. Stacked bar graphs are often a good alternative to using multiple pie charts since they depict how much the size of a given category changes under varying circumstances. Such graphs also eliminate the need to construct a separate bar for each category, as would occur when using multiple bar graphs.

Advantages

- Contains richer information than bar graphs or multiple pie charts
- Shows how each category contributes to the overall value of a variable
- Show each nominal or ordinal category in frequency distribution

- Display relative proportions of multiple categories
- Summarize a large data set in visual form
- Clarify trends better than either tables or arrays
- Estimate key values at a glance
- Permit a visual check of the accuracy and reasonableness of calculations
- Easily understood due to widespread use in business and the media

Disadvantages

- Can be visually confusing
- Require additional verbal or written explanation
- Complicated to prepare
- Can be easily manipulated to give false impressions
- Inadequate to describe the attribute, behavior, or condition of interest
- Fails to reveal key assumptions, norms, causes, effects, or patterns
- Reveals little concerning the central tendency, dispersion, skew, or kurtosis

2.2.8 Conclusions

We summarize within the table below the key differences between the various types of charts which you will need to consider when selecting a chart for the display of a given data set. The table allows you to quickly select the table type which is most appropriate for your data set and the nature of the phenomenon you wish to emphasize.

The table consists of the number of columns in which various characteristics of the graphs are displayed. In the first column we display the type of the graph. In the second column we denote whether the graph is most suited to numbers (N), proportions (P) or both (N/P). In the third column we indicate whether the source data is better in raw (R) or grouped form (G). Please note, that for Pie Charts and Stacked Bars this characteristic does not apply. In the last three columns we indicate what type of data the graph is most suited to displaying. The primary data type is indicated by 'XX', and the secondary preference (if any) is denoted by 'X'.

Graph Type	N/P	R/G	Nominal	Ordinal	Interval
Bar Graph	N	R	XX	X	
Freq. Poly	N	R			XX
Histogram	N/P	G		X	XX
Line Graph	N/P	R		X	XX
Ogive	N/P	G			XX
Pie Chart	P	-	XX	X	
Stacked Bar	P	-	XX	X	

Chapter 3

JavaBeansTM Guide

3.1 Overview

The JavaBeans technology offers a new perspective on how to divide the labor between software developers and graphic designers. It provides mechanisms that both minimize the design and implementation effort for modest system upgrades while fully supporting the design, implementation, and assembly of enterprise wide software solutions.

The distinguishing feature of JavaBeans components as reusable components is that they operate across all platforms supported by Java. This means that any JavaBeans component will work inside any operating system running a Java Virtual Machine, or even inside a browser supporting Java applets.

3.2 Installing the JavaBeans

This JavaBeans suite contains the following files:

- Documentation
 - Product Details
 - * Product Description
 - * System Requirements
 - * License Agreement
 - JavaDocs Files
 - JavaBeans Guide
 - Guide to WebCab Components
- JavaBeans Java Archive
- Examples and Demos

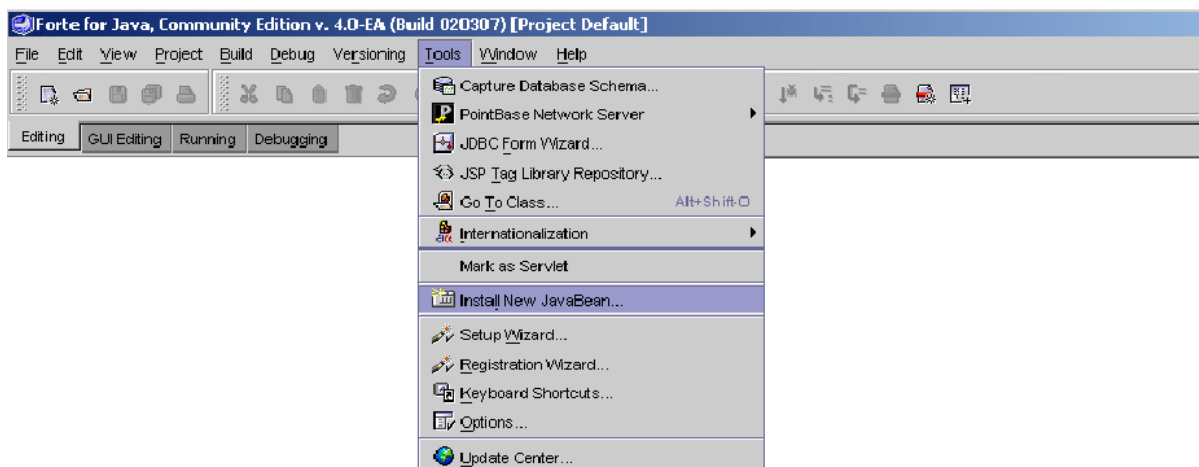
The documentation provided includes the description for the product, the *java generated documentation* with precise explanations for each *class*, *method* and *field*. The beans come wrapped into an archive with a *JAR* extension. This file contains the whole directory structure of the beans, additional classes and serialized components and a *manifest file*. The manifest file is required by IDEs to tell the Java beans from the other classes in the archive. It is located in *META-INF/MANIFEST.MF*. After identifying each of these components, you may take the *JAR* file and import it into any compatible IDE.

3.3 Using JavaBeans inside an IDE

By importing the *JAR* file into a JavaBeans integrated development environment (IDE), you can start adding the beans into your working area. As soon as you do this, you will be given the opportunity to edit their settings in a properties window. This is where attributes such as *font styles*, *colors*, *on/off switches*, *titles* can be fully customized, allowing the designer to control the design aspect of the bean without worrying about its implementation.

3.3.1 Importing the *JAR* file into Sun's One IDE

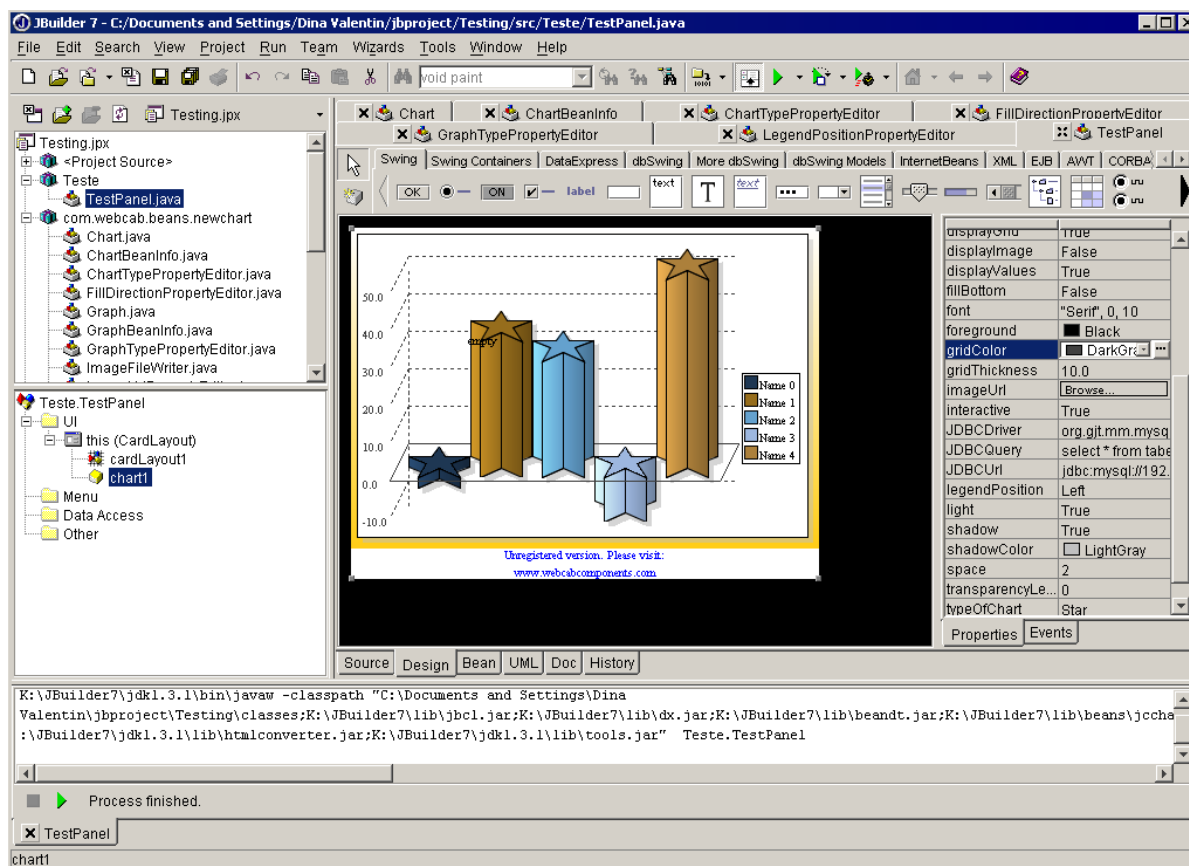
To make the *JAR* file available to the Sun's One IDE (formerly Forte For Java), first start up the IDE and select 'Tools > Install New JavaBean'. Now select the *JAR* file which contains the JavaBeans by browsing the local file directory within the pop-up window and click OK. Another pop-up window will display the names of the JavaBeans contained within the *JAR* file. Select the beans you wish to install and click OK. In the next window please select the 'Palette Category' you wish to add these JavaBeans to and click OK. This completes the installation of the JavaBean component to the Palette from which the beans can be used within future application development.



3.3.2 Importing the *JAR* file into Borland's JBuilder IDE

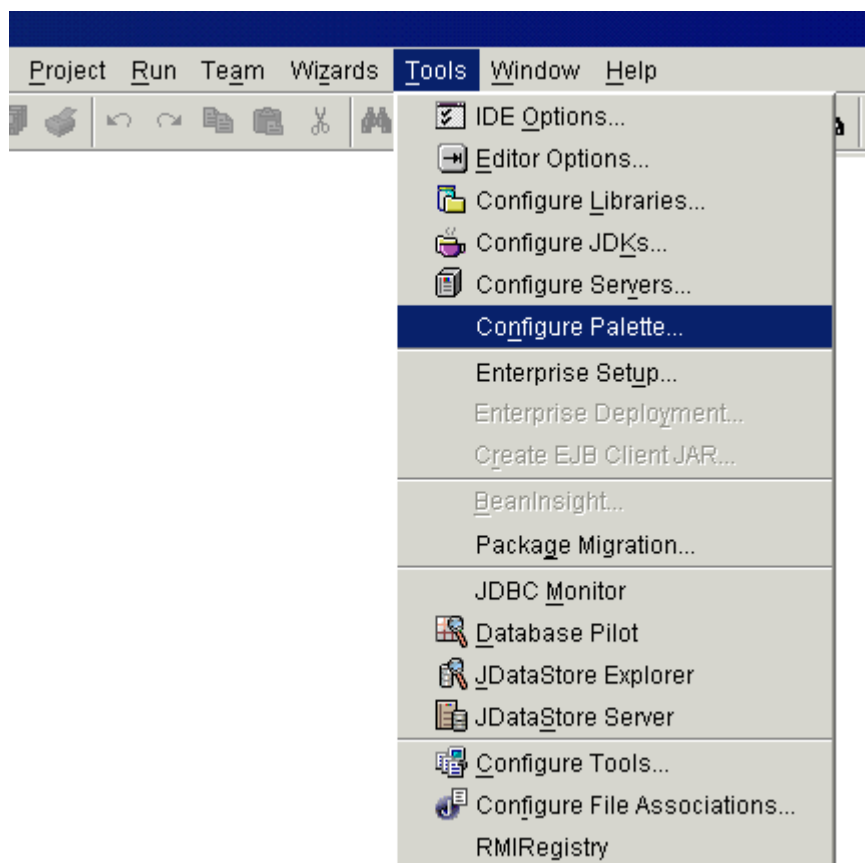
To import the *JAR* file containing the JavaBean components into a JBuilder project select from the menu 'Tools > Configure Palette'. Now a pop-up window will appear which has two tags, 'Pages' and 'Add Component'. You can either choose to create a page for the imported JavaBeans (i.e. *JAR*) or they may be added to an existing page. To create a new page click on 'Add' in 'Pages', choose a name and click OK. To add the *JAR* file first select the 'Add Components' tab and then click on 'Select Library'. Another window will pop-up entitled 'Select a different library', if the *JAR* file is not displayed in the 'user home' folder then click the 'New' button. Now within the pop-up entitled 'New Library Wizard' enter the name you wish to give the imported library of JavaBeans contained within the *JAR* and click 'Add'. Select the *JAR* file by browsing your local file directory in the pop-up window and then click OK. You should have now been returned to the 'New Library Wizard' with the *JAR* file location inserted within the 'Library Paths', if this is the case click OK, otherwise click 'Add' and repeat the previous step. Select the *JAR*'s library name within the 'Select a different library' window and click OK. To finish click the 'Add from Selected Library' button within the 'Palette Properties' window. The results of the import will be displayed within a 'Results' window which completes the imported onto the JBuilder Palette.

Below is a screen-shot of working with JGraph inside the JBuilder software.

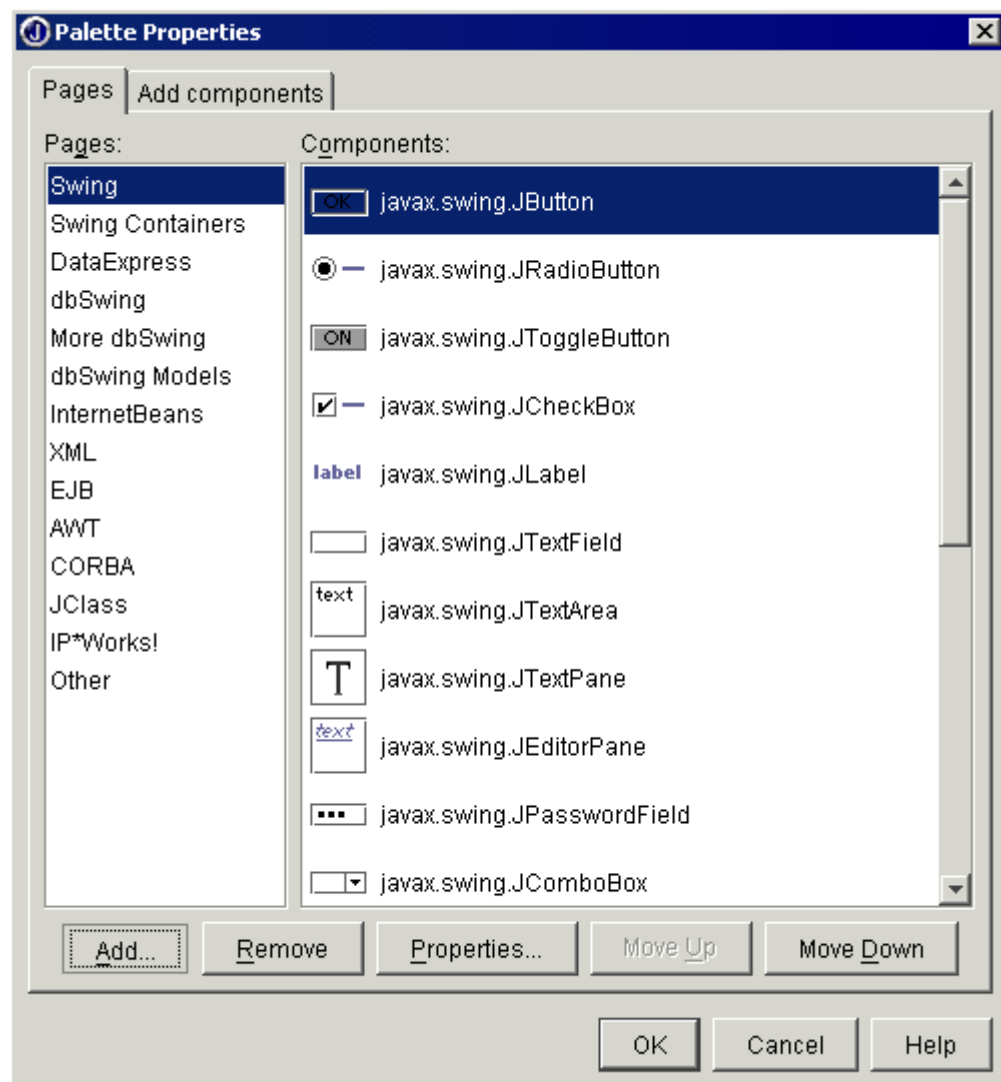


3.3.3 Create a JavaBean charting client using JBuilder in 20 Steps

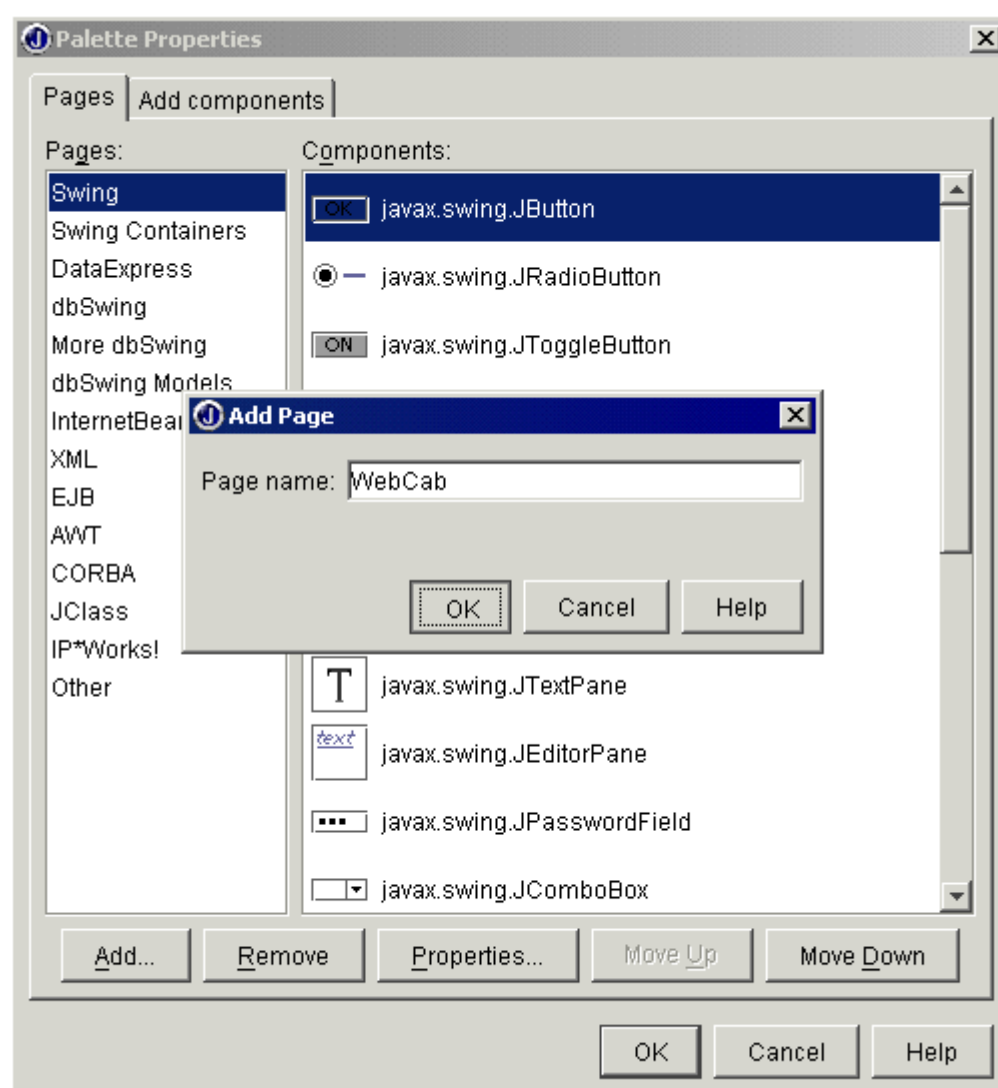
This section shows exactly how to get started with the JGraph JavaBean component using Borland's JBuilder IDE.



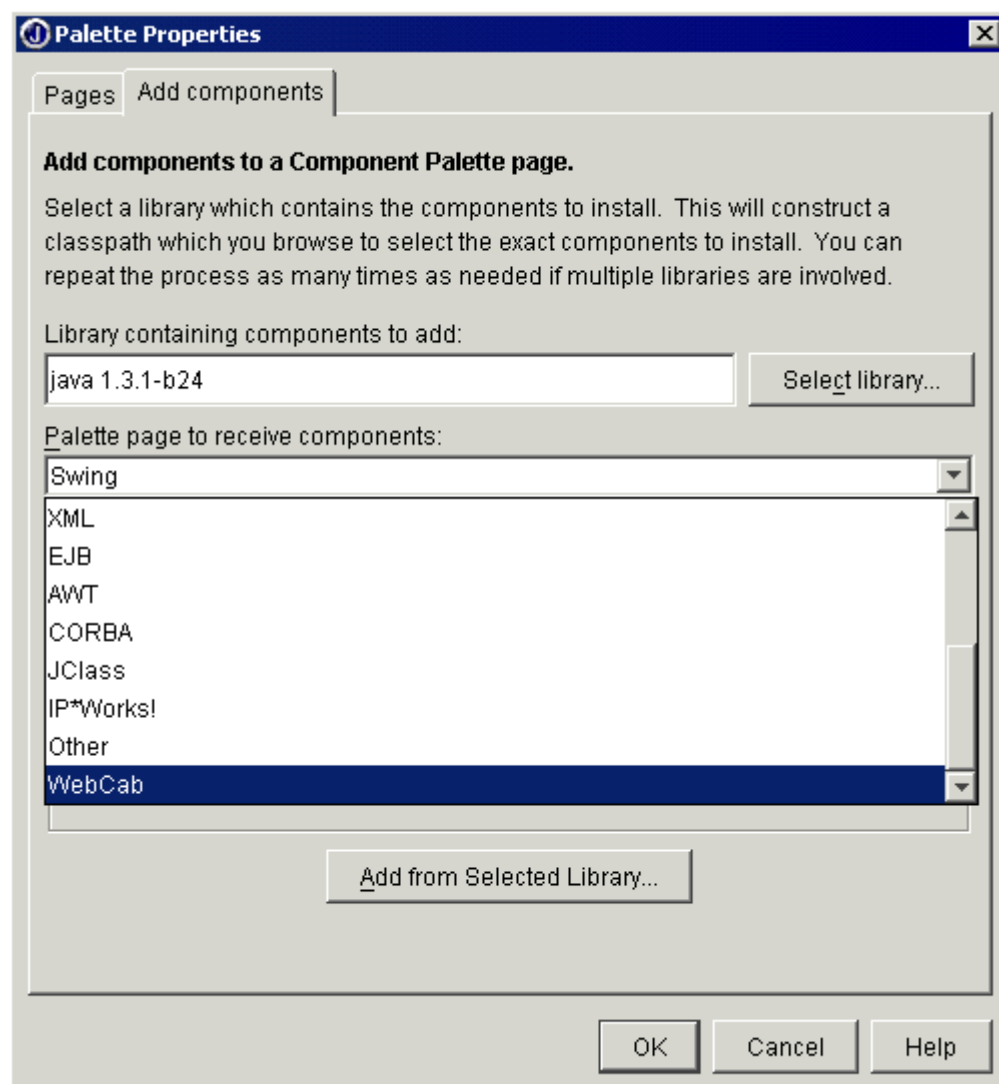
Step 1. Open the 'Tools' menu and select 'Configure Palette...'. You will get a dialog called 'Palette Properties'.



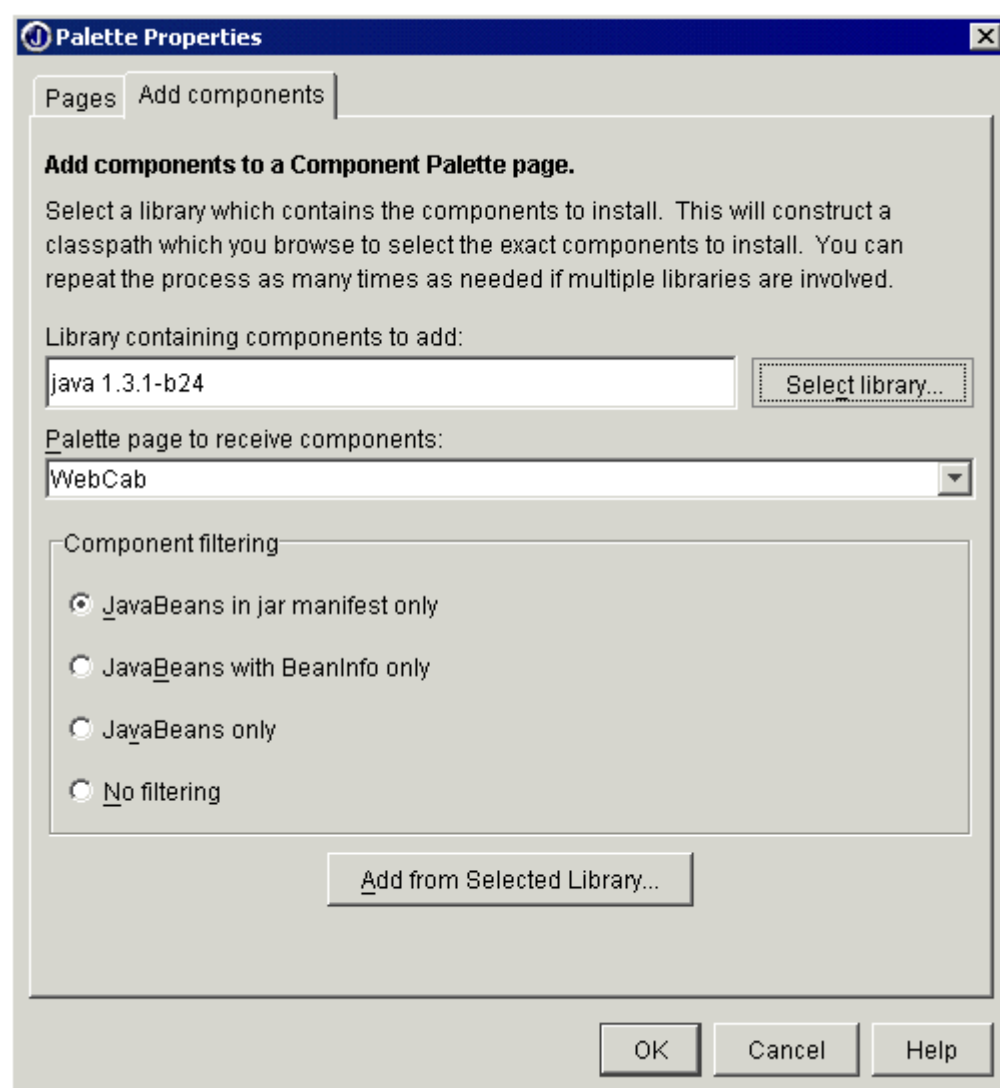
Step 2. In the 'Palette Properties' panel click on the 'Add' button. You will get a dialog called 'Add Page'.



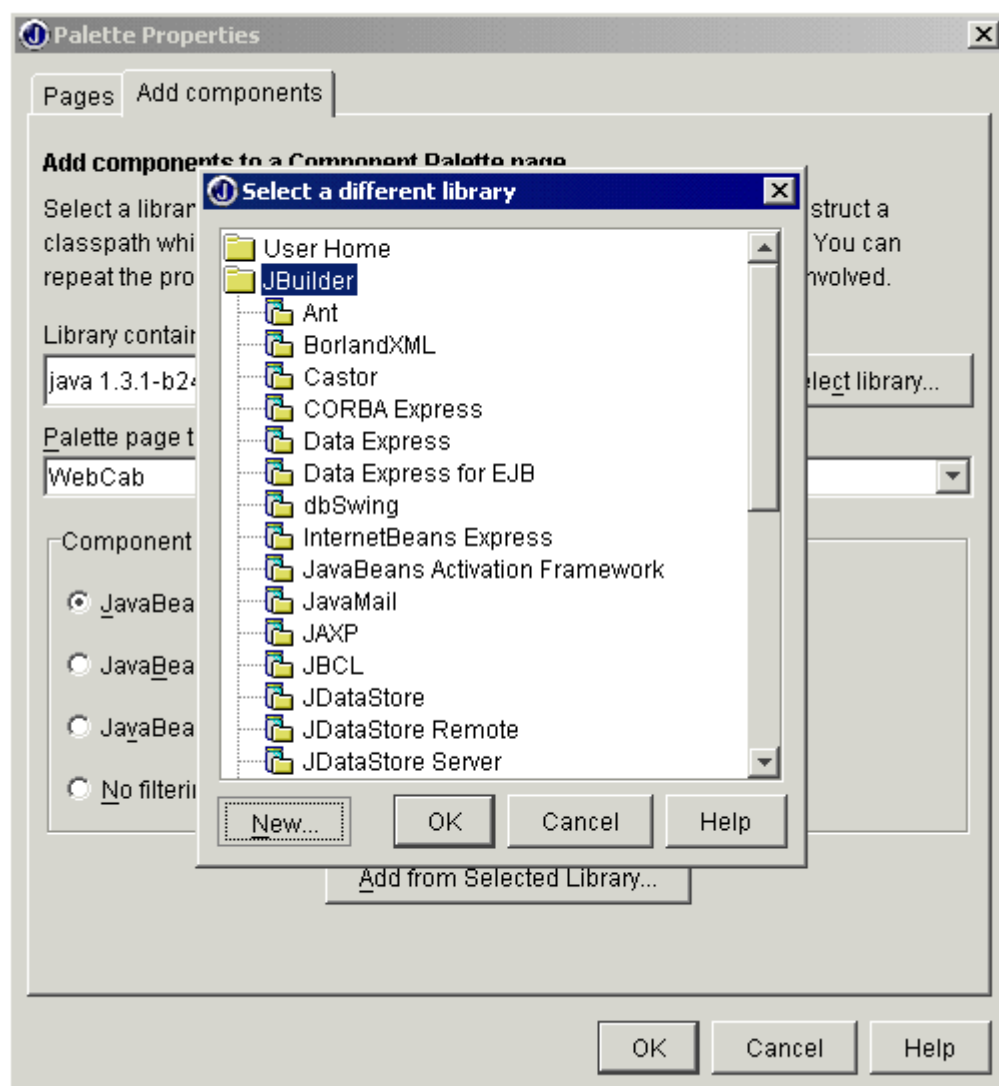
Step 3. Here write “WebCab” inside the text field, and press ‘OK’.



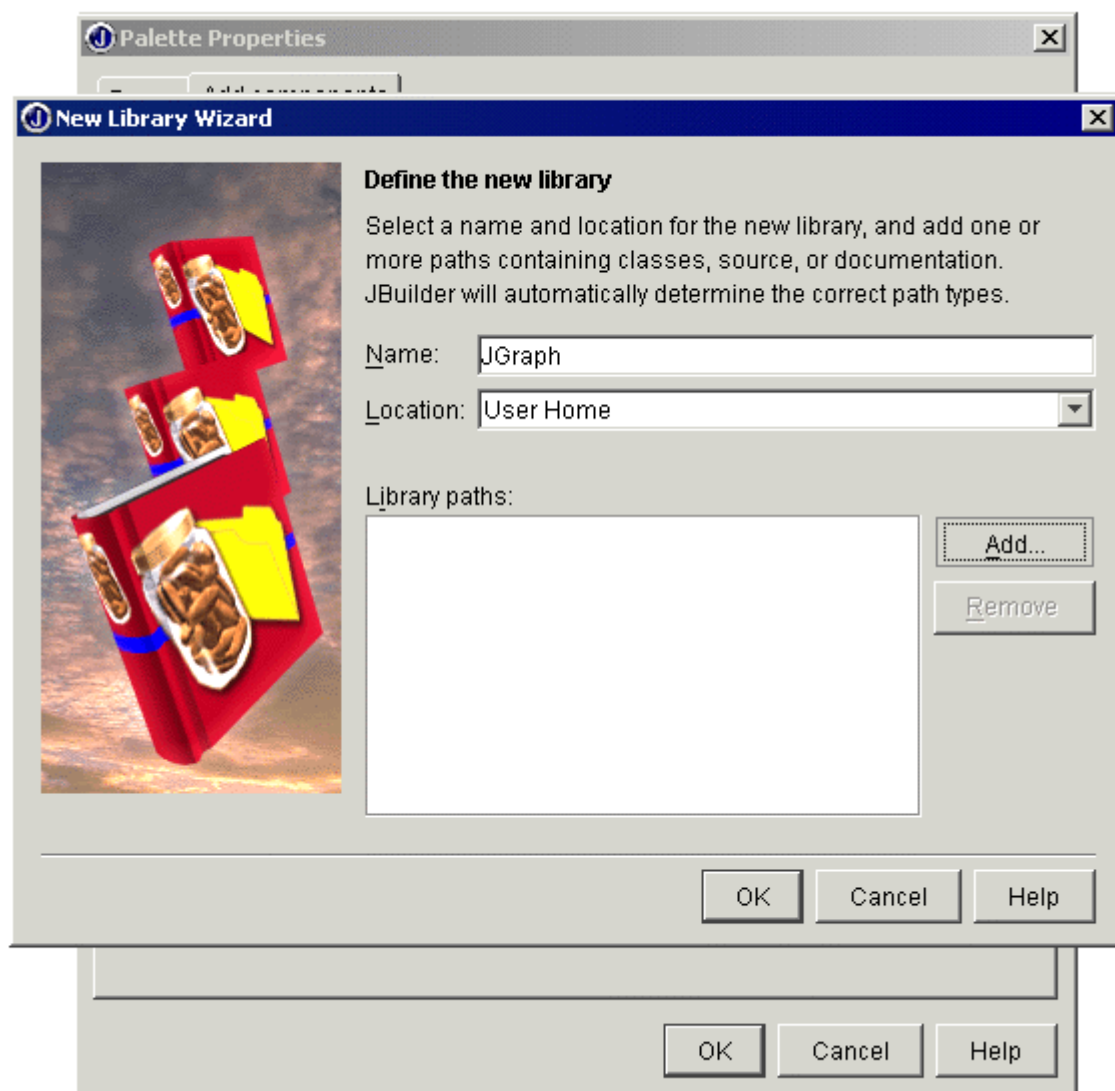
Step 4. Go to the tab 'Add components' and select 'WebCab' in the 'Palette page to receive components' drop down list.



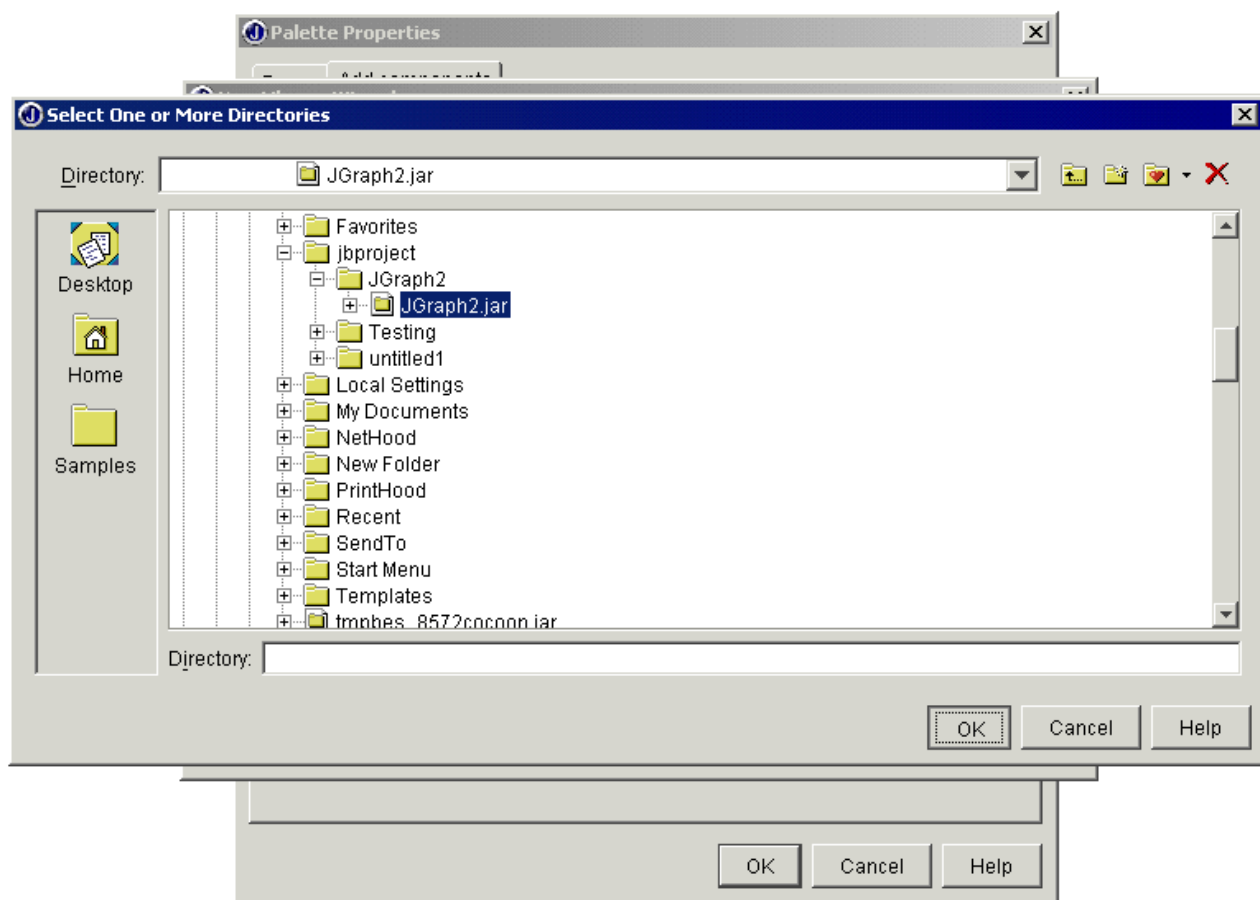
Step 5. Now click on the 'Select library' button. You will get a dialog called 'Select a different library'.



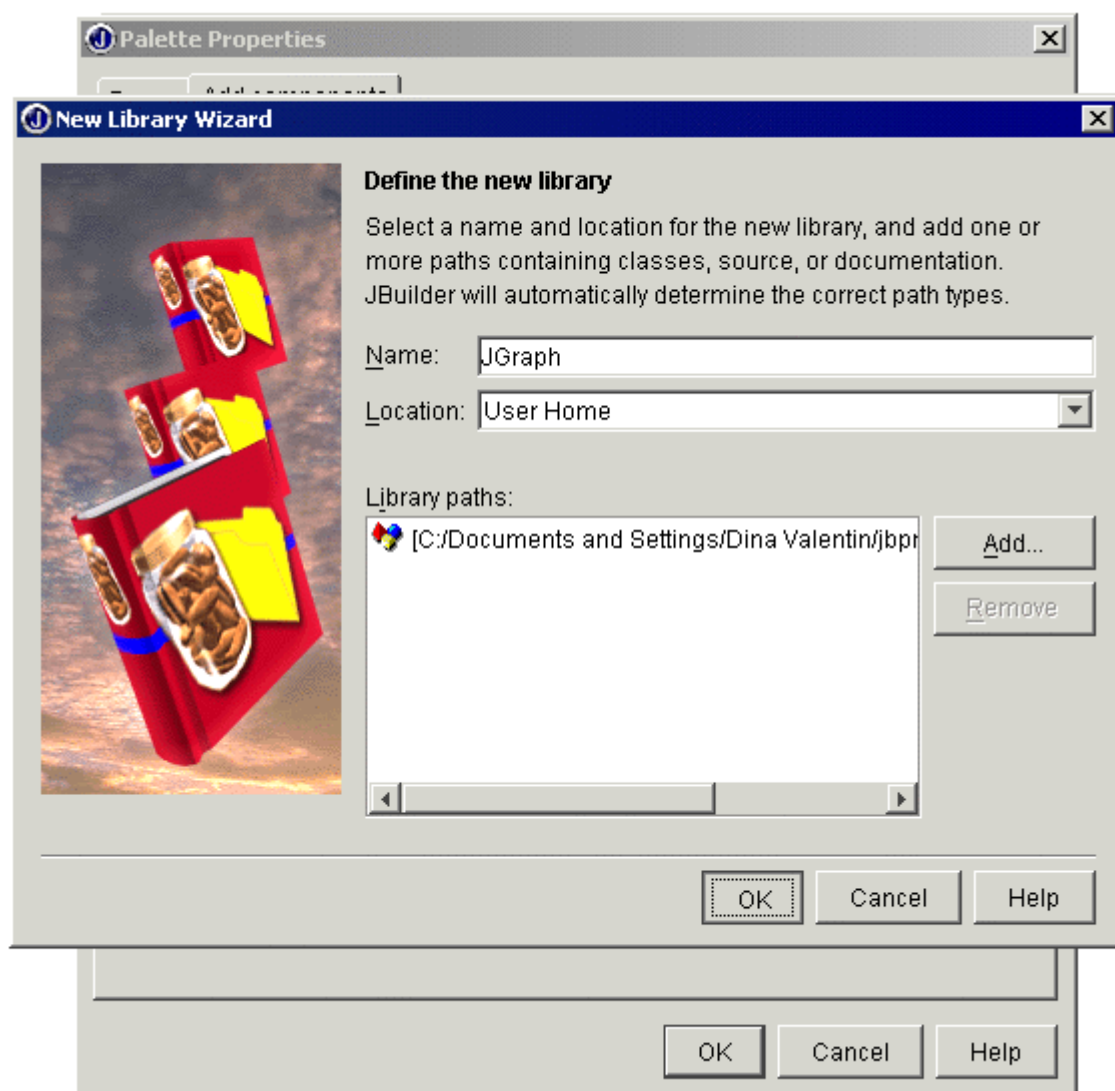
Step 6. Now click on the 'New' button. You will get a wizard called 'New Library Wizard'.



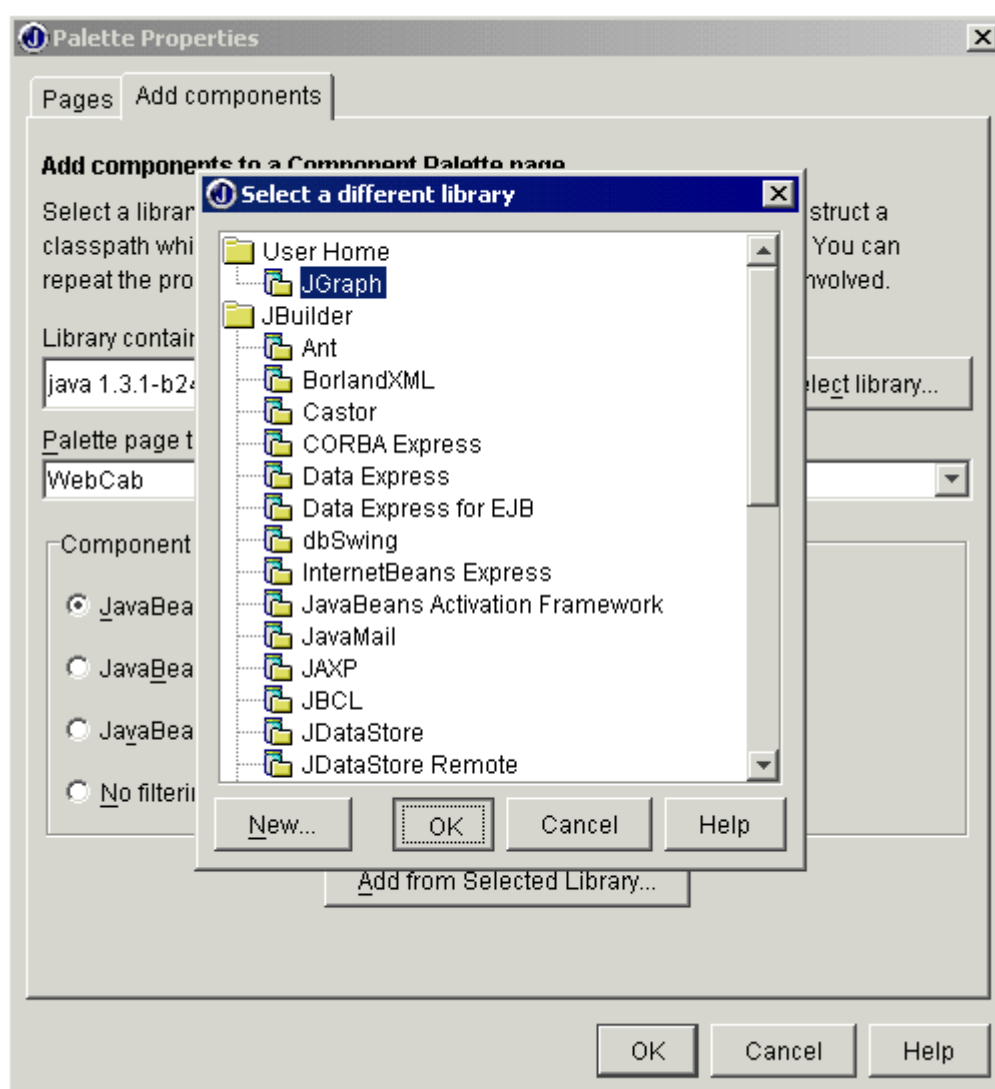
Step 7. On the 'Name' text field write "JGraph" and click on the 'Add' button. You will get a new dialog from which you can select the library.



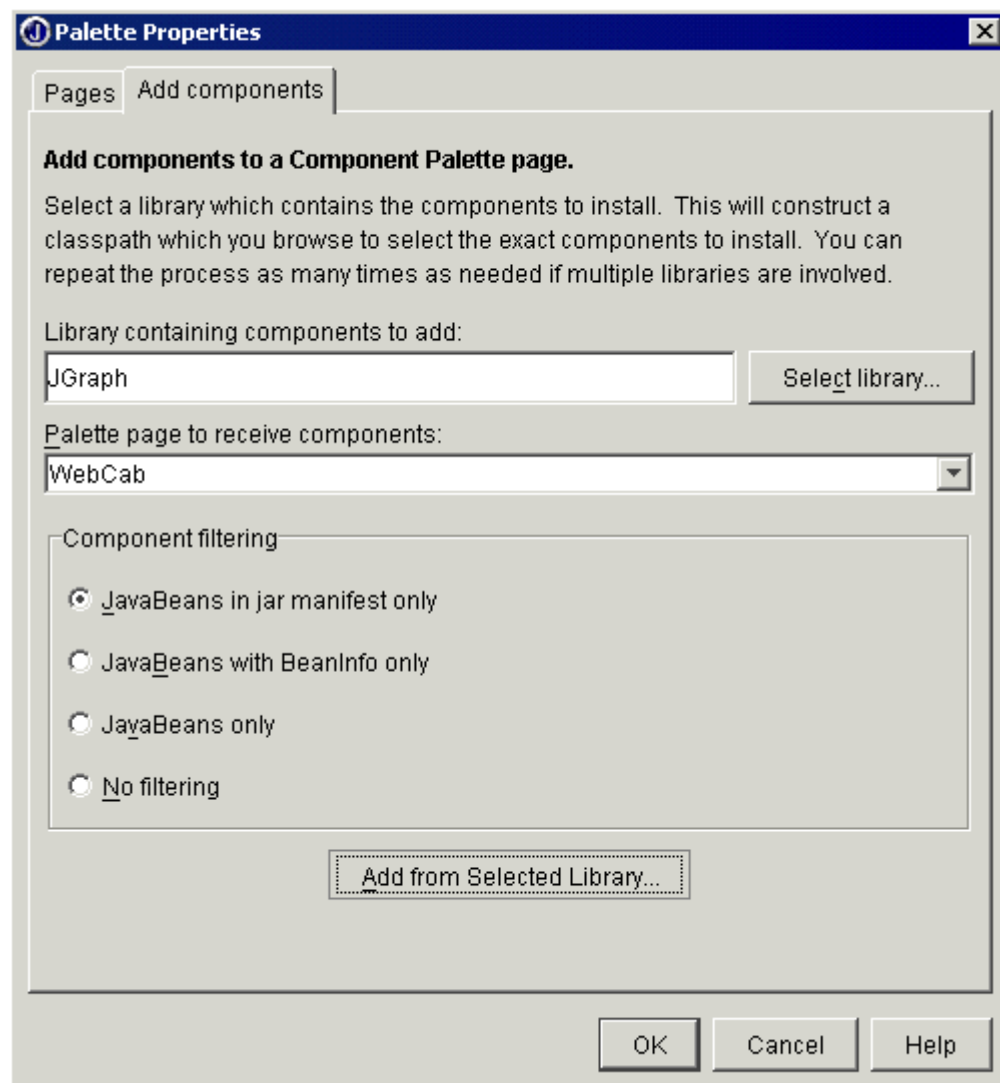
Step 8. Locate the 'JGraph2.jar' on your hard-drive, select it and press 'OK'.



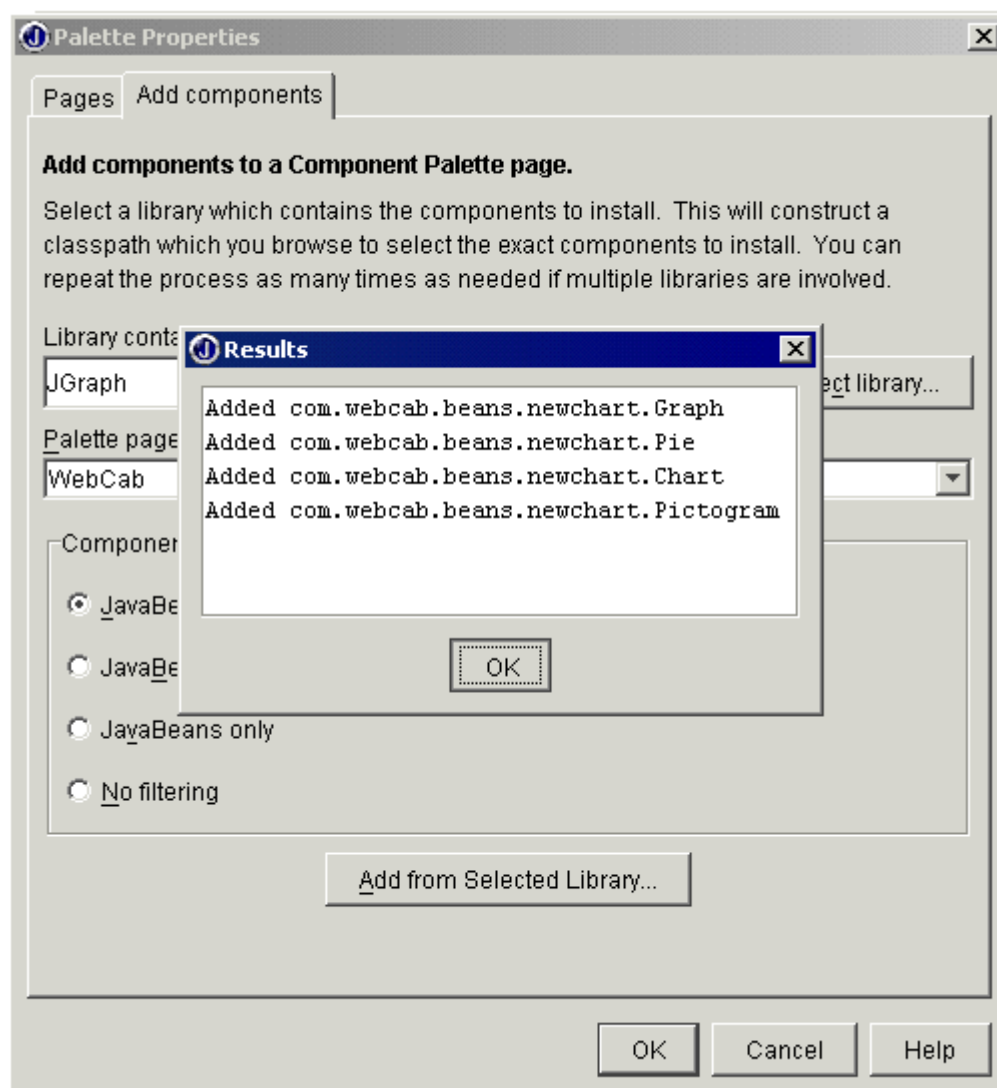
Step 9. In the 'Library paths:' a new entry is added. Click on 'OK'.



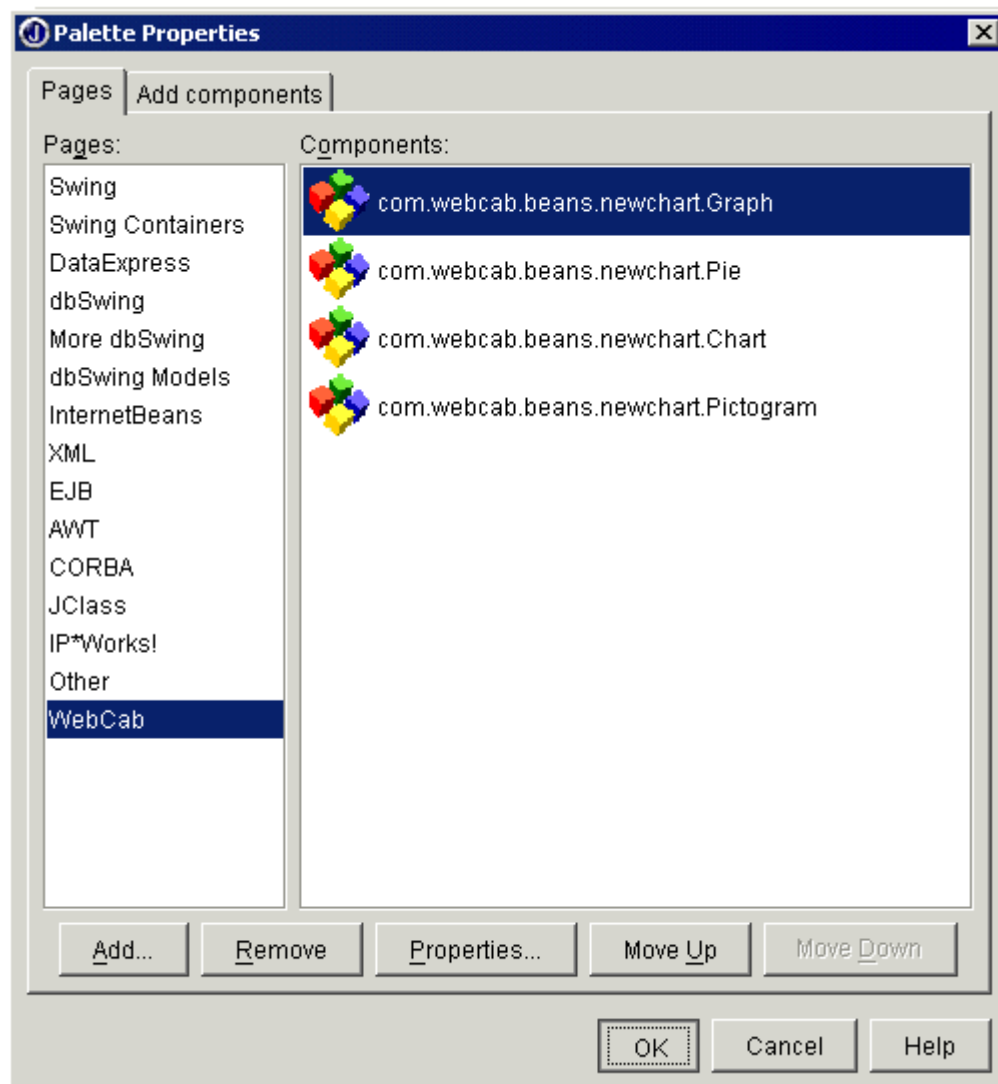
Step 10. Click 'OK' again to go back to the previous screen.



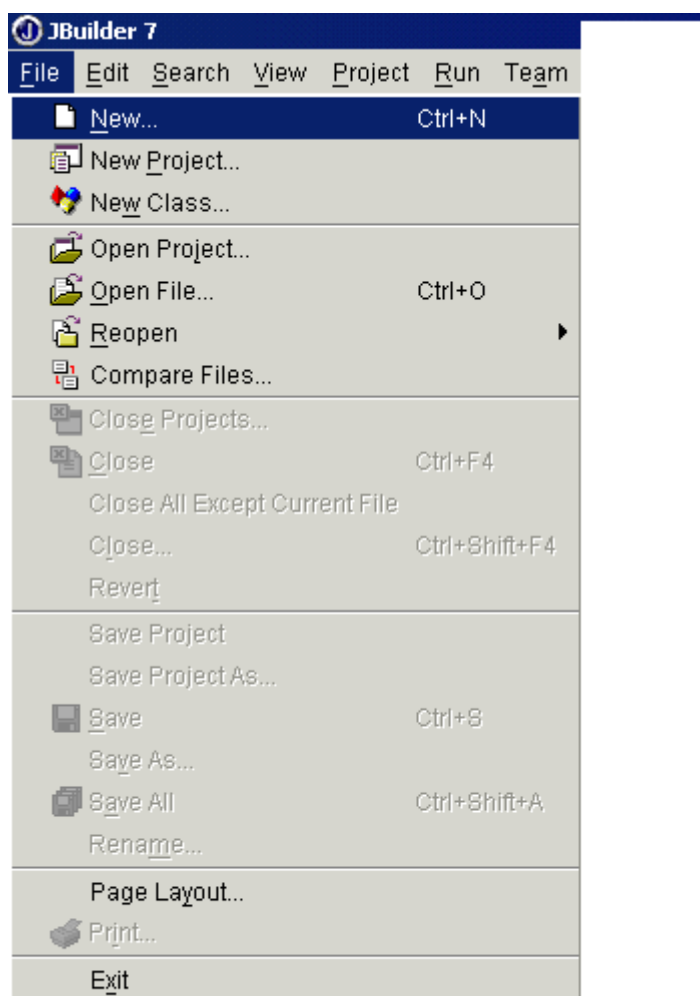
Step 11. Now click on the 'Add from selected library' button. A dialog will appear to inform you that the new components were added.



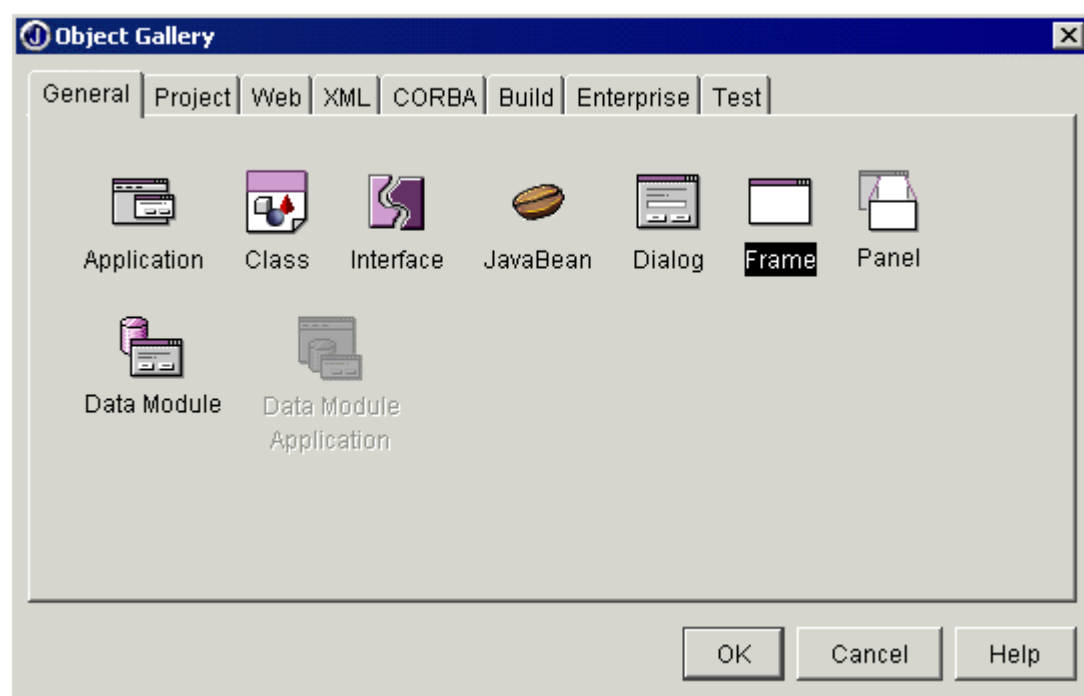
Step 12. Click 'OK' again to go back to the previous screen.



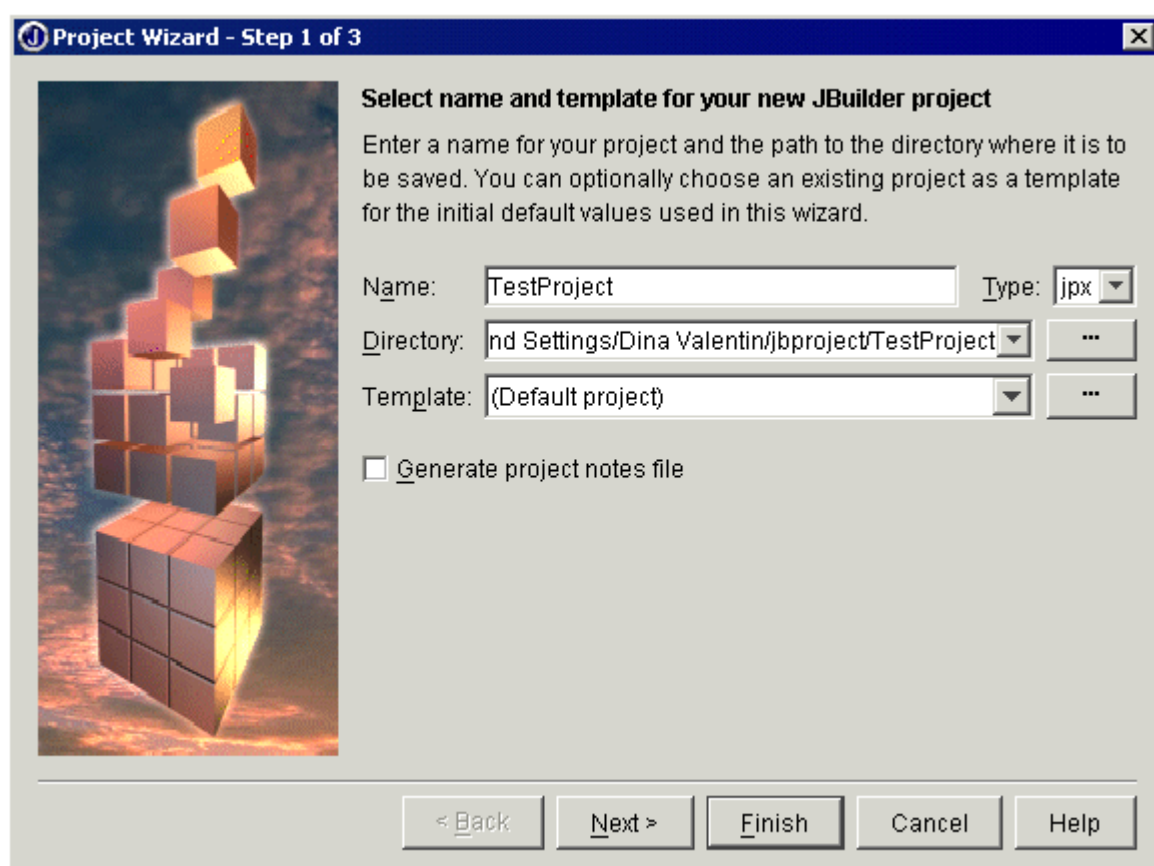
Step 13. On the 'Pages' tab you will see the new components. Click on 'OK' to close the window.



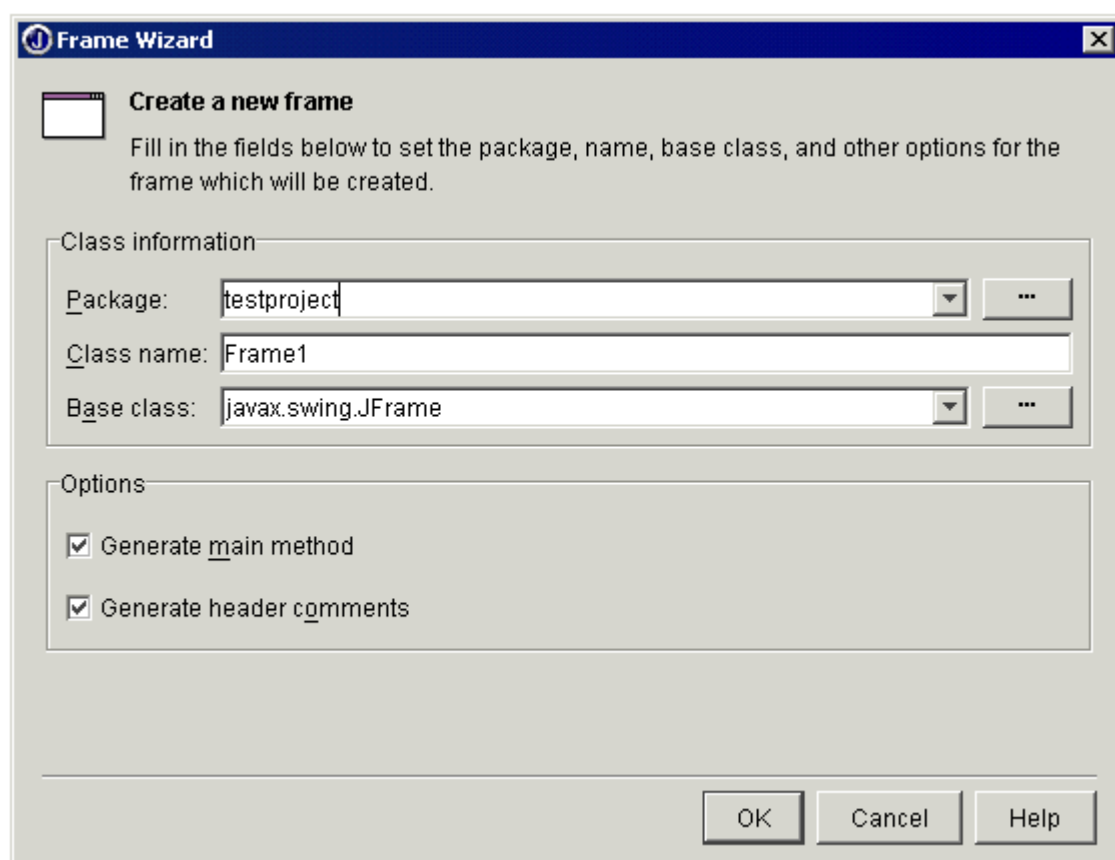
Step 14. Now let's build a container to test the beans. On the 'File' menu choose 'New'.



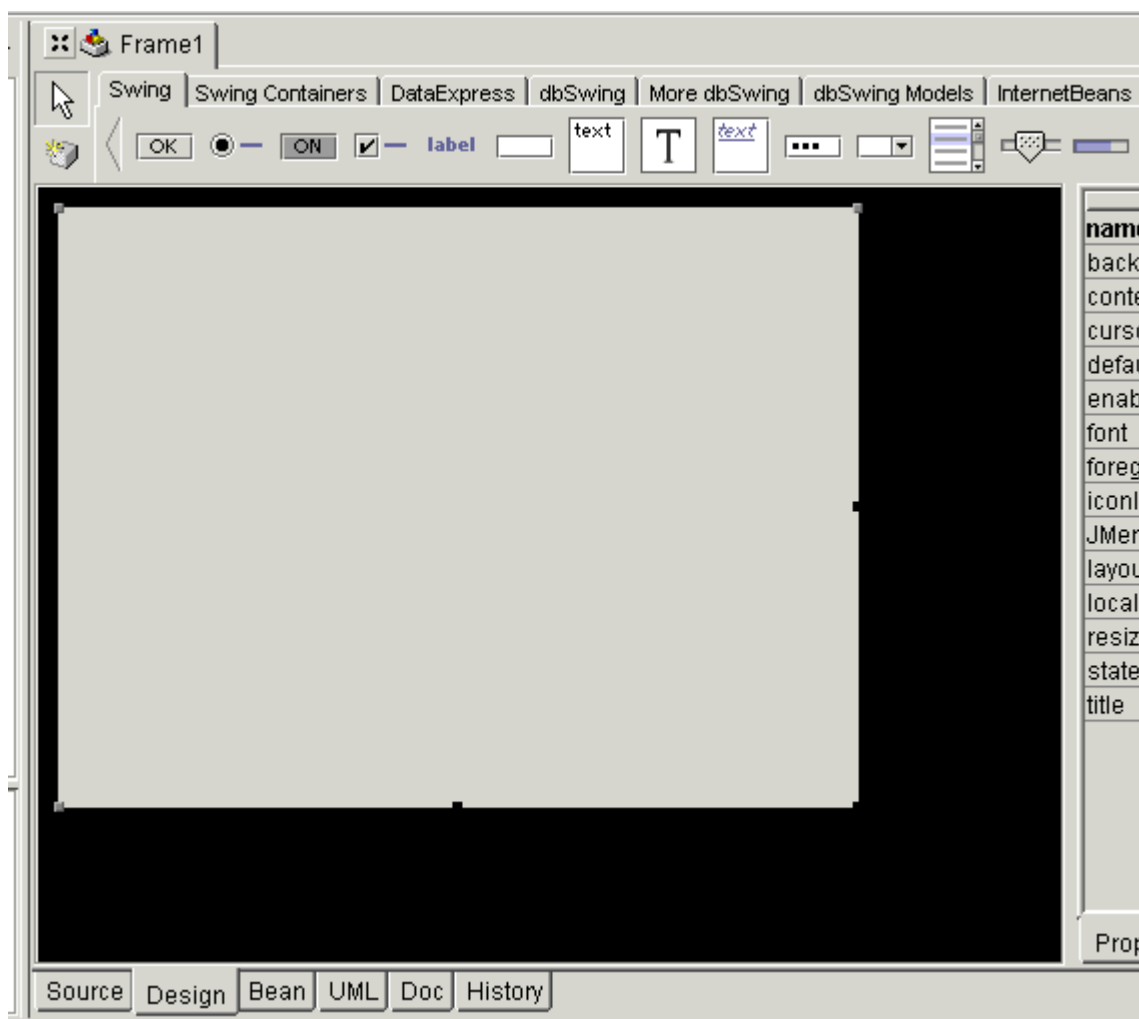
Step 15. On the 'Object Gallery' window choose 'Frame' and click 'OK'.



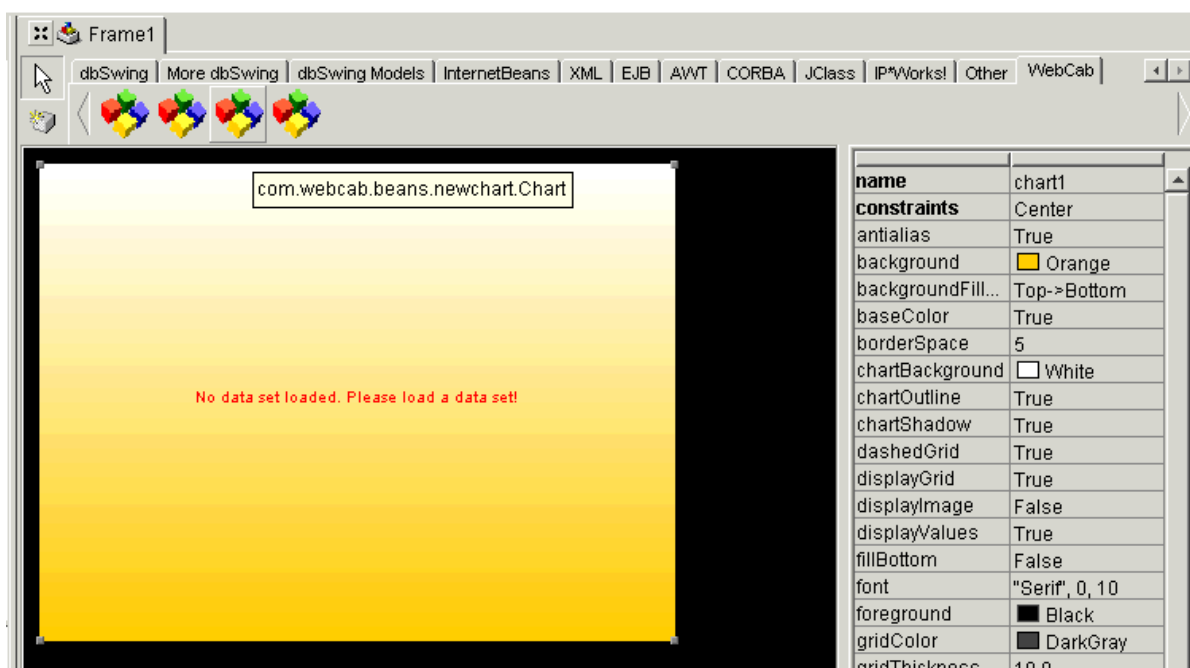
Step 16. Within the Project Wizard, type “TestProject” in the name field and press ‘Finish’.



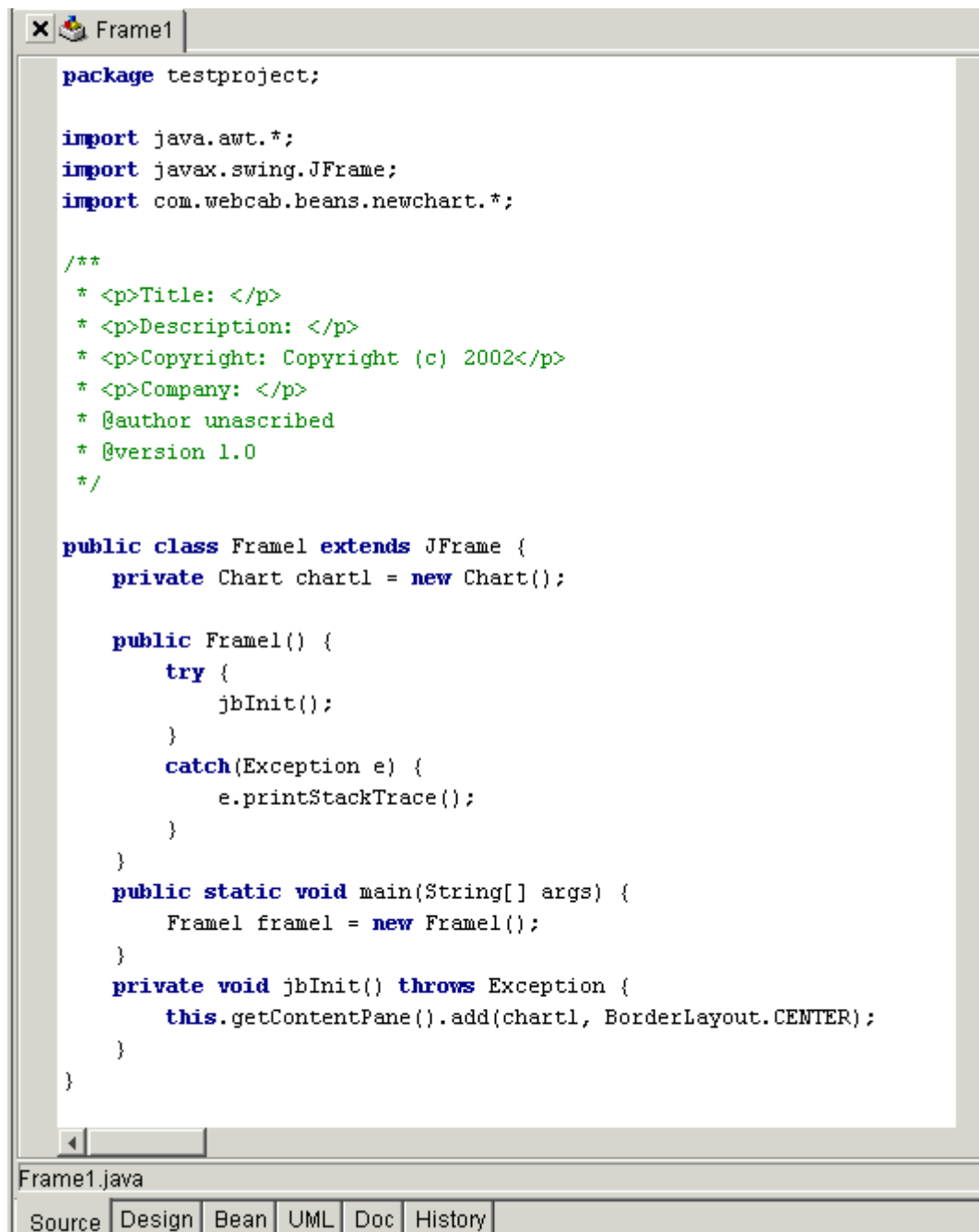
Step 17. Click 'OK' to close the dialog.



Step 18. Click the 'Design' tab and scroll to the 'WebCab' palette tab at the top.



Step 19. Drag one of the components onto the panel.



Step 20. By clicking on the 'Source' tab you are able to edit the source code. Now you are able to test the beans within the SWING container generated by JBuilder. Now just press 'F9' to run this client application.

3.4 Developer Support

3.4.1 Documentation

In order to use the JavaBeans component in any application, you should first read the information provided in the *Product Description* and *System Requirements*. As soon as you are aware of any incompatibilities that might turn up, you may start browsing the *Java Generated Documentation* (JavaDocs) to understand the structure of each bean.

3.4.2 Compilation and Custom Modification of Source Code

All WebCab Components are compiled using Sun's JDK1.3, and should you prefer compilation of the source code to be done using another compiler then please contact us. Some functions (for example standard mathematical operations) are embedded within the source code, thus, if you license classes which you believe offer a similar level of functionality from a third party and wish to include these classes within our components then please contact us. Both of the above services are offered free of charge at WebCab's discretion.

3.4.3 Online Support

Should you have any questions or queries, please feel free to email us at:

support@WebCabComponents.com

For updates and new releases, visit:

www.WebCabComponents.com

Chapter 4

Programmer's Guide

4.1 Introduction

JGraph contains four components: Graph, Chart, Pie and Pictogram. Within this chapter we explain their functionality and features allowing you to fully take advantage of them.

JGraph can be used inside most Java IDEs such as Borland's JBuilder, Eclipse and Sun ONE (formerly Forte for Java). You may prefer to use direct implementation, in which case you should note that each component extends the Canvas class, so you can add them to any Container such as Frames or Panels.

4.2 How to import data

All the charts and graphs use the same mechanism for importing data and hence these remarks apply to all the chart types. To import data into one of the charts you have to use a database such as Oracle, DB2, SQL Server, MySQL, etc.

4.2.1 Loading data from a Database

To load data from a database you will need to have a JDBC driver installed. The most widely used databases today (i.e. Oracle, DB2) have freely distributed JDBC drivers. They either come bundled with the product or can be downloaded from the vendors' website. Please note that for some databases there are different JDBC drivers with widely varying performance for a given application. Also, note that there exists a generic ODBC-JDBC driver and therefore any database which is ODBC compatible for example MS Access can also be integrated with our components.

JDBC Drivers for Oracle, DB2, SQL Server and MySQL

Here we detail how to install JDBC drivers for Oracle9i.

Drivers for different types of databases can be obtained from the following address:

industry.java.sun.com/products/jdbc/drivers

The drivers are usually java archive files (.jar files). After you downloaded the driver be sure to make it available by inserting its path in the classpath.

For example, let's assume that you want to use an Oracle 8/9i database. You have downloaded a file named "oracle.jar". The path to the driver main class should be specified by the driver provider. This path will be used inside the setJDBCdriver property:

```
setJDBCdriver("oracle.jdbc.driver.OracleDriver");
```

Before using the JDBC feature inside JGraph please find the correct JDBC Driver for your database. The JGraph component pack doesn't include any JDBC Driver.

Access your database

Once the JDBC driver is installed, you can access your database by setting the URL and query. The URL must contain the type of the database, the IP address of the server where JGraph can find the database, the port and the service. The URL structure is driver specific, so it must be provided by the driver's vendor. For example if we want to access a Oracle 8/9i database called "MYDATABASE" that resides on a server having the 192.168.0.1 IP address, within the setJDBCurl property you will have to write the following:

```
jdbc:oracle:thin:@192.168.0.1:1521:MYSERVICE
```

After specifying the URL you must provide the user name and password.

After the connection to the database has been established we will write the query. For example, if inside the database we have a table that contains on the first column "double" type values and on the second column "string" values, it can be used as input for the Chart component. To do so, we have to use the following query inside the setJDBCProperty:

```
select * from MYTABLE
```

where MYTABLE is the name of the table to be used.

JDBCdriver	oracle.jdbc.driver.OracleDriver
JDBCPassword	MY_PASSWORD
JDBCQuery	select * from MY TABLE
JDBCurl	jdbc:oracle:thin:@192.168.0.1:1521:SERVICE
JDBCUser	MY_NAME

Figure 2. Illustrates how to set the JDBCQuery inside an IDE.

4.3 Customizing JGraph

Below we detail each of the JGraph's options.

4.3.1 Customizing the Chart component

- **Antialias** - This enables the anti-aliasing capabilities. Anti-alias means that all the inclined lines and curves will look smoother.
- **Background** - This is the background color of the chart. It is only visible on the border. See “chartBackground” for information on how to define a background for the space occupied by the chart.
- **BaseColor** - If the user chooses a chart with 3D perspective, he can choose to fill the base of the chart with the color specified by the “gridColor” property.
- **BorderSpace** - This value specifies the width of the border filled with color.
- **ChartBackground** - This is the background color for the space occupied by the chart. It is recommended to be left white, for easy reading.
- **ChartOutline** - When set to true, this option causes a bounding line to be shown for the surface occupied by the chart.
- **ChartShadow** - If this option is set to true, the chart will have shadow.
- **DashedGrid** - This option gives the user the possibility to choose between a solid or dashed line to be used for drawing the back grid.
- **DisplayGrid** - This option enables or disables the back grid. The axis will always be visible, regardless of the DisplayGrid property.
- **DisplayImage** - When this option is set to true the image specified by the “imageUrl” property is used as background instead of the current solid color.
- **DisplayValues** - This option gives the user the opportunity to choose if the values of the grid are shown or not.
- **FillBottom** - When you have a 3D perspective chart and you choose to fill the bottom, the base of the negative value bars will not be seen, unless you set this option to true.
- **FillDirection** - This specifies the gradient direction of the background color. If this is set to “Disabled” no gradient will be used. Otherwise, the color specified by the “background” property will be combined with white into a gradient color, the direction of the gradient being specified by this property. There are six directions:
 - from Top to Bottom
 - from Bottom to Top
 - from Left to Right
 - from Right to Left

- across the First Diagonal
 - across the Second Diagonal
- **Font** - This is the font used. The user has the possibility to choose between Java standard fonts.
- **Foreground** - This is the foreground color of the chart. The default value is black.
- **GridColor** - This option specifies the color used to draw the grid and, if the case, to fill the base of the 3D perspective charts.
- **GridThickness** - This specifies the space between the lines of the grid. The space is not specified in pixels, but in chart values.
- **ImageUrl** - This is the string representation of the image URL used as background.
- **Interactive** - When this value is set to true, the client user can interact with the chart (i.e. by moving it or changing the values). Otherwise all the mouse clicks will be ignored.
- **JDBCDriver** - This is the driver used to connect to a database. For more details go to the “How to import data” section.
- **JDBCQuery** - This is the query used to retrieve data from a database. For more details go to the “How to import data” section.
- **JDBCUser** - This is the user name used to connect to a database. For more details go to the “How to import data” section.
- **JDBCPassword** - This is the password used to connect to a database. For more details go to the “How to import data” section.
- **JDBCUrl** - This is the URL of the database. For more details go to the ”How to import data” section.
- **LegendPosition** - This option specifies the legend position. There are four positions available: Left, Bottom, Right and Top. The legend can also be disabled from here.
- **Light** - This enables the light effects. Light effects apply only to 3D perspective charts..
- **Shadow** - If this option is set to true, the bars of the chart will have shadows.
- **ShadowColor** - This is the color used for rendering the shadows. The default color is light gray.
- **Space** - This is the space in pixels between charts, when loading a chart with multiple charts.

- **TransparencyLevel** - This option specifies the transparency level of the chart. Valid values are from 0, opaque, to 100, fully transparent (invisible).
- **TypeOfChart** - This is the type of chart used to display the data. There are available 6 types of charts:
 - Bar Chart in 2D perspective
 - Bar Chart in 3D perspective
 - Cone Chart
 - Pyramid Chart
 - Cylinder Chart
 - Star Chart
- **TypeOfImage** - This is the type of image alignment, when using an image as background. The available options are:
 - Top Left
 - Top Center
 - Top Right
 - Middle Left
 - Middle Center
 - Middle Right
 - Bottom Left
 - Bottom Center
 - Bottom Right
 - Tiled
 - Stretched

4.3.2 Customizing the Pie component

- **Antialias** - This enables the anti-aliasing capabilities. Anti-alias means that all the inclined lines and curves will look smoother.
- **Background** - This is the background color of the pie chart. An image can also be used instead of a solid color.
- **Depth** - This is the depth of the 3D perspective pies.
- **DisplayGrid** - This option enables or disables the back grid.
- **DisplayImage** - When this option is set to true the image specified by the “imageUrl” property is used as background instead of the current solid color.

- **DisplayValues** - This option gives the user the opportunity to choose if the values of each pie are shown or not.
- **Font** - This is the font used. The user has the possibility to choose between Java standard fonts.
- **Foreground** - This is the foreground color of the pie. The default value is black.
- **GridColor** - This option specifies the color used to draw the grid.
- **GridThickness** - This specifies the space between the lines of the grid. This is the only bean from the ones that support grid where this value represents pixels. In the other two cases the values are zoom dependent.
- **ImageUrl** - This is the string representation of the URL of the image used as background.
- **Interactive** - When this value is set to true, the client user can interact with the pie. Otherwise all the mouse clicks will be ignored.
- **JDBCDriver** - This is the driver used to connect to a database. For more details go to the “How to import data” section.
- **JDBCQuery** - This is the query used to retrieve data from a database. For more details go to the “How to import data” section.
- **JDBCUser** - This is the user name used to connect to a database. For more details go to the “How to import data” section.
- **JDBCPassword** - This is the password used to connect to a database. For more details go to the “How to import data” section.
- **JDBCUrl** - This is the URL of the database. For more details go to the “How to import data” section.
- **Light** - This enables the light effects. Light effects apply only to 3D perspective pies.
- **NameOfPie** - A name can be associated with the pie chart, by giving this option a non null string value.
- **UseRandomColors** - If this option is set to true, the pie will use random generated colors. Otherwise, it will use colors generated by a function.
- **Space** - This is the space in pixels from the edge of the pie to the closest edge of the client area.
- **Transparency** - This option specifies the transparency level. Valid values are from 0, opaque, to 100, fully transparent (invisible).

- **Type** - This is the type of pie chart used to display the data. There are 3 types of pie charts available:
 - Pie Chart in 2D perspective
 - Pie Chart in 3D perspective
 - Donut Chart
- **TypeOfImage** - This is the type of image alignment, when using an image as background. The available options are:
 - Top Left
 - Top Center
 - Top Right
 - Middle Left
 - Middle Center
 - Middle Right
 - Bottom Left
 - Bottom Center
 - Bottom Right
 - Tiled
 - Stretched

4.3.3 Customizing the Graph component

- **Antialias** - This enables the anti-aliasing capabilities. Anti-alias means that all the inclined lines and curves will look smoother.
- **CenterX** - The graph's x axis will be centered on the coordinate specified here.
- **CenterY** - The graph's y axis will be centered on the coordinate specified here.
- **DisplayAxis** - Enables or disables the graph's axis.
- **DisplayGrid** - When this option is set to true, a grid is shown on the back of the graph. The grid helps to better estimate the position of the points. It can also be used for "snap to grid" operations.
- **DisplayImage** - When this option is set to true, the image specified by the "imageUrl" property is used as background instead of the current solid color.
- **DisplayPoints** - This specifies if the interpolation points are displayed or not. The interpolation points represent the values found in the database.

- **DragGraphEnabled** - If this option is enabled, the client can drag the whole graph to view a certain part of it. This is useful when you have a graph that doesn't fit into the viewing area.
- **DragPointsEnabled** - If this option is enabled, the client can drag the interpolation points, modifying the graph's curve.
- **DragTolerance** - When you want to drag a point, it is difficult to exactly grab the point with the mouse pointer. We introduced this option to make this thing easier. If you click near a point, and the distance between the mouse pointer is smaller than the value specified by this property, the program will consider that you clicked on the point. So the dragging becomes possible even if you didn't actually click on the point.
- **FirstDerivative** - The first and the last derivative are used by the cubic splines interpolation. These are the values of the first derivative for the first and last interpolation point.
- **GridColor** - This option specifies the color user to draw the grid.
- **GridX** - This specifies the space between the vertical lines of the grid. The space is not specified in pixels, but in graph values, zoom dependent.
- **GridY** - This specifies the space between the horizontal lines of the grid. The space is not specified in pixels, but in graph values, zoom dependent.
- **ImageUrl** - This is the string representation of the URL of the image used as background.
- **JDBCDriver** - This is the driver used to connect to a database. For more details go to the "How to import data" section.
- **JDBCQuery** - This is the query used to retrieve data from a database. For more details go to the "How to import data" section.
- **JDBCUrl** - This is the URL of the database. For more details go to the "How to import data" section.
- **JDBCUser** - This is the user name used to connect to a database. For more details go to the "How to import data" section.
- **JDBCPassword** - This is the password used to connect to a database. For more details go to the "How to import data" section.
- **LastDerivative** - The first and the last derivative are used by the cubic splines interpolation. These are the values of the last derivative for the first and last interpolation point.

- **LegendPosition** - This option specifies the legend position. There are four positions available: Left, Bottom, Right and Top. The legend can also be disabled from here.
- **LineWidth** - This is the width in pixels of the curves. Only strict positive values are allowed. Default value is 1.
- **PointsWidth** - This value is added to the “lineWidth” value to obtain the width of the bounding square of each interpolation point.
- **RoundPoints** - If this option is set to true the points will be represented by a disc. Otherwise, they will be represented by a square.
- **ScaleX** - This is the scale factor for the x axis. The functionality is similar to the zoom option, but it is designed for only one scale.
- **ScaleY** - This is the scale factor for the y axis. The functionality is similar to the zoom option, but it is designed for only one scale.
- **SnapToGrid** - If this is set to true, when the client will drag the interpolation points, they will be snapped to the existing grid.
- **TipBackground** - This is the color used for the background of the tool tip.
- **TipDelayMillis** - This is the amount of time in milliseconds before the tool tip will appear.
- **TipForeground** - This is the color used as foreground for the tool tip.
- **Type** - This is the type of interpolation used to generate the curve. There is a “Cubic Splines” interpolation method, three “Polinomial”, one “Rational”. In addition, direct lines can be used.
- **TypeOfImage** - This is the type of image alignment, when using an image as background. The available options are:
 - Top Left
 - Top Center
 - Top Right
 - Middle Left
 - Middle Center
 - Middle Right
 - Bottom Left
 - Bottom Center
 - Bottom Right
 - Tiled

– Stretched

- **UseToolTip** - When this is set to true, a tool tip will appear on the screen indicating the current mouse position relative to the graph. When you drag a point, the tool tip will also appear.
- **XaxColor** - The color of the x axis.
- **Xlabel** - The string label of the x axis.
- **XlabelColor** - This is the color of the label corresponding to the x axis.
- **XvaluesAngle** - The angle specified by this value is the angle between the base line of the label corresponding to the x axis and the y axis. Valid values are from -45 to 45, default value being 0.
- **YaxColor** - The color of the y axis.
- **Ylabel** - The string label of the y axis.
- **YlabelColor** - This is the color of the label corresponding to the x axis.
- **YvaluesAngle** - The angle specified by this value is the angle between the base line of the label corresponding to the y axis and the x axis. Valid values are from -45 to 45, default value being 0.
- **Zoom** - This is the zoom factor, in percentages. Using this option, correlated with the moving feature (it can also be used by specifying the “centerX” and “centerY” values), a certain part of the graph can be analyzed.

4.3.4 Customizing the Pictogram component

- **Background** - This is the background color of the pictogram.
- **Font** - This is the font used. The user has the possibility to choose between Java standard fonts.
- **ImageUrl** - This is the string representation of the image URL that will be used to represent the data within the pictogram.
- **JDBCDriver** - This is the driver used to connect to a database. For more details go to the “How to import data” section.
- **JDBCQuery** - This is the query used to retrieve data from a database. For more details go to the “How to import data” section.
- **JDBCUser** - This is the user name used to connect to a database. For more details go to the “How to import data” section.

- **JDBCPassword** - This is the password used to connect to a database. For more details go to the “How to import data” section.
- **JDBCUrl** - This is the URL of the database. For more details go to the “How to import data” section.
- **TextColor** - This is the color used to render all the texts.
- **ValueForKey** - This is the value represented by each image. For example, if you want to represent 20, and the “valueForKey” property is 10, you will get two images.

4.4 Developer's Support

4.4.1 Compilation and Custom Modification of Source Code

This component and related Java™ source files have been compiled using Sun's JDK1.3.1, should you prefer compilation of the source code using another compiler then please contact us. Some functions are embedded within the source code, if you license classes which you believe offer a similar level of functionality from a third party and wish to include these classes within our components then please contact us. Both of the above services are offered free of charge at WebCab's discretion.

4.4.2 Online

Should you have any questions or queries, please feel free to email us at:

support@WebCabComponents.com

For updates and new releases, visit:

<http://www.WebCabComponents.com>

Chapter 5

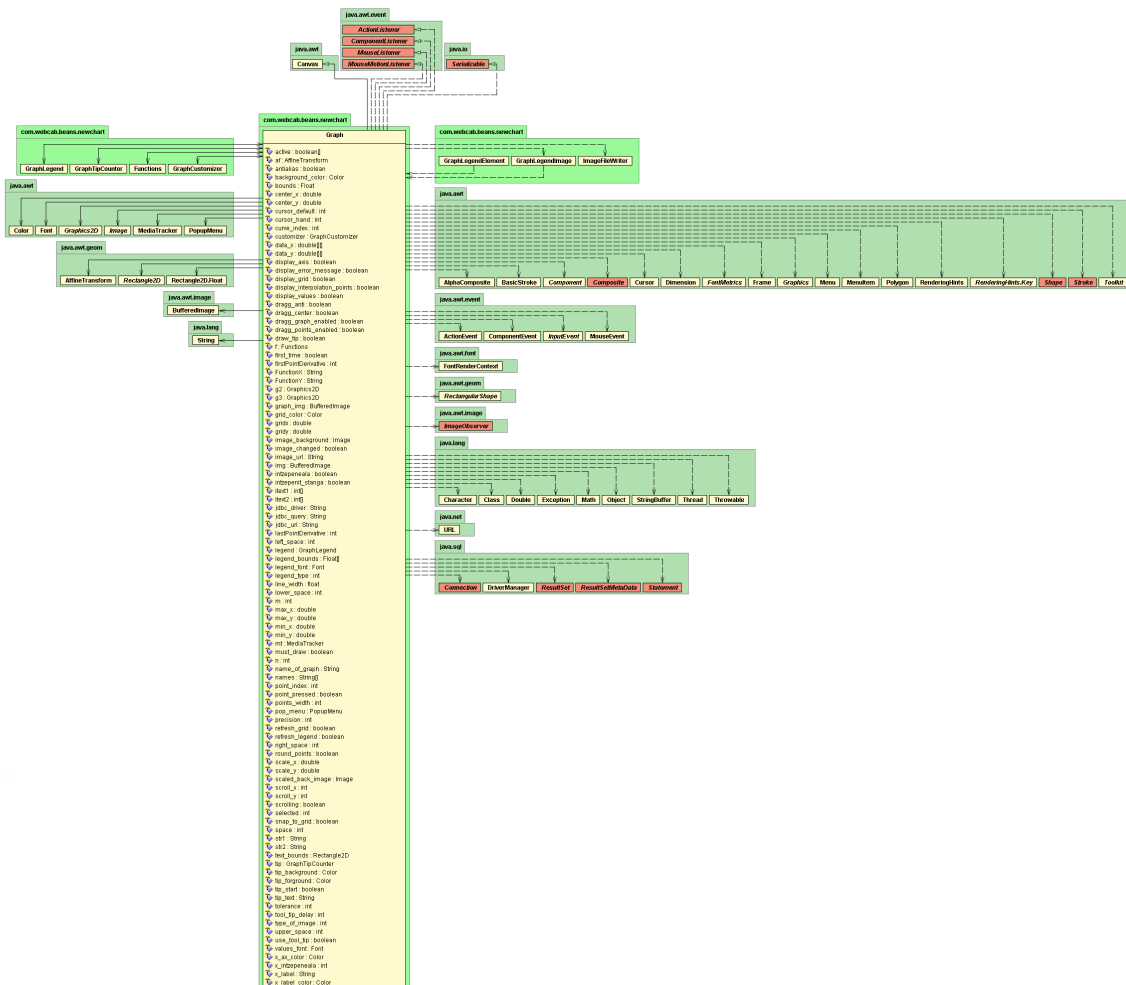
JGraph UML Diagrams

Within this chapter we provide UML diagrams for the interfaces of the JGraph component. The diagrams use standard UML notation detailing the inheritance, dependency, association, interface, methods, fields and properties of this component and associated packages. We present the UML diagrams with a large amount of white space so that the developer has space in which to add additional remarks.

UML Diagram Key

- **Extended Classes (interfaces)** - A solid line with a large triangle that points from the subclass (resp subinterface) to the superclass (resp inherited interface)
- **Classes** - Displayed in a rectangular box with a default yellow background with the name at the top and field, methods, and properties listed below it
- **Abstract Class and Methods** - Displayed in italic font
- **Implementing Classes** - A dashed line with a large triangle which points from the implementing class to the inherited interface
- **Interfaces** - A rectangle with orange background and the interface name in italic font
- **Implemented Interfaces** - A dashed line with a large triangle which points from the implementing class to the implemented interface
- **Dependencies/Reverse Dependencies** - A dashed line with arrowhead
- **Associations/Reverse Associations** - A solid line with an arrowhead
- **Packages** - A green rectangle with a tab and the package name in the tab or below it
- **Methods** - Listed below members and fields, including the return type
- **Members/fields** - Listed below the class name, including the return type
- **Properties** - Properties are displayed separately in the bottom of the class diagram
- **Static** - Static members, fields, variables, and methods are underlined in the diagram

Zoom to at least 360% in order to be able to read the names of the fields, methods, component names and packages.

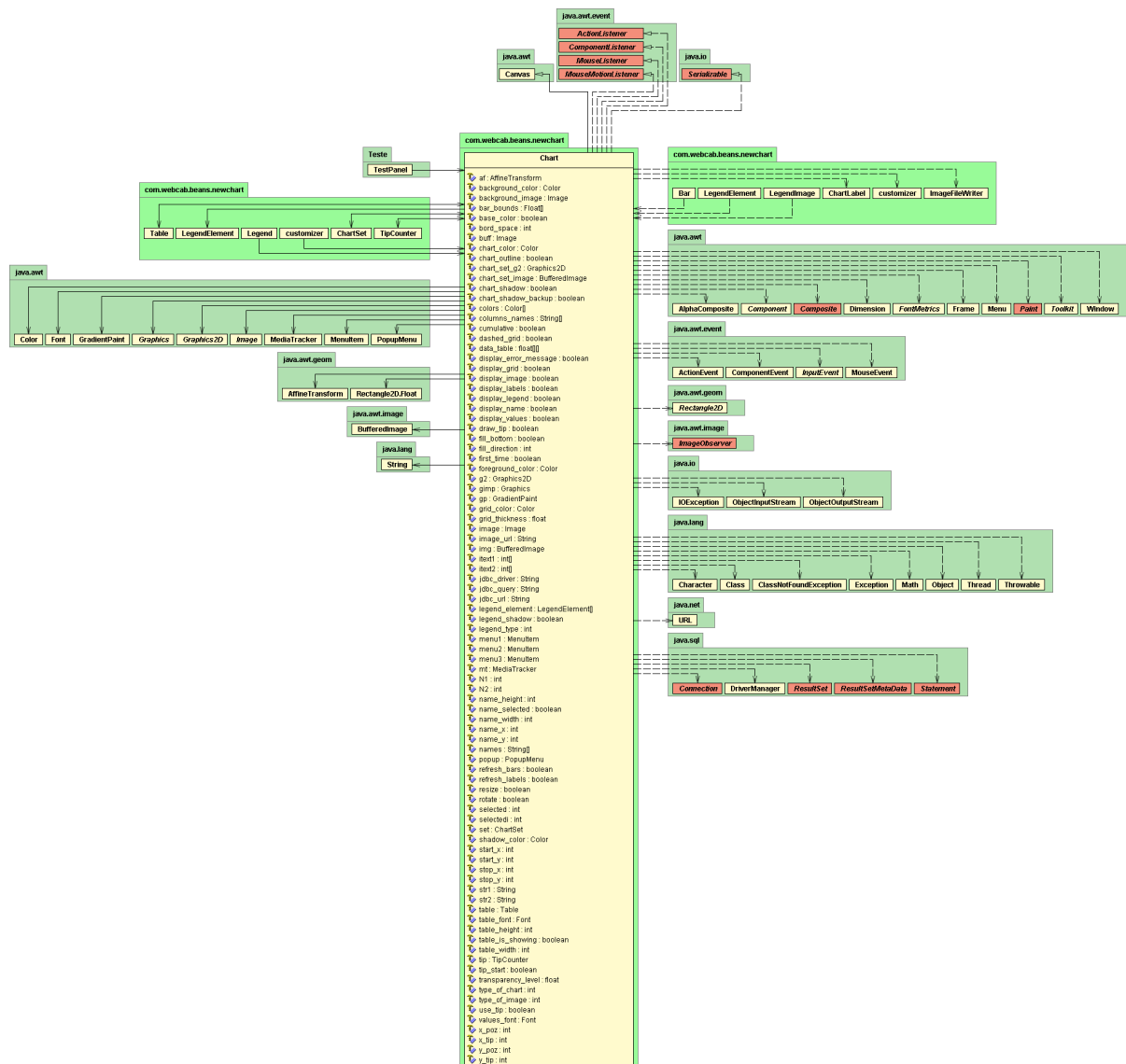


50

- x_pos : int - x_tip : int - x_values_degrees : int - y_ax_color : Color - y_label : String - y_label_color : Color - y_pos : int - y_tip : int - y_values_degrees : int
- actionPerformed() : void - componentHidden() : void - componentMoved() : void - componentResized() : void - componentShown() : void - getAntialias() : boolean - getCenterX() : double - getCenterY() : double - getDisplayAxis() : boolean - getDisplayGrid() : boolean - getDisplayImage() : boolean - getDisplayPoints() : boolean - getDraggGraphEnabled() : boolean - getDraggPointsEnabled() : boolean - getDraggTolerance() : int - getFirstDerivative() : int - getGridColor() : Color - getGridY() : double - getImageUri() : String - getJDBCDriver() : String - getJDBCQuery() : String - getJDBCUri() : String - getLastDerivative() : int - getLegendPosition() : int - getLineWidth() : float - getMinimumSize() : Dimension - getName() : String - getPointsWidth() : int - getRoundPoints() : boolean - getScaleX() : double - getScaleY() : double - getSnapToGrid() : boolean - getTipBackground() : Color - getTipDelayMillis() : int - getTipForeground() : Color - getTypeOfImage() : int - getUseToolTip() : boolean - getVaxColor() : Color - getVlabel() : String - getVlabelColor() : Color - getVvaluesAngle() : int - getVaxColor() : Color - getVlabel() : String - getVlabelColor() : Color - getVvaluesAngle() : int - Graph() : void - mouseClicked() : void - mouseDragged() : void - mouseEntered() : void - mouseExited() : void - mouseMoved() : void - mousePressed() : void - mouseReleased() : void - paint() : void - setCenterX() : void - setCenterY() : void - setDisplayAxis() : void - setDisplayGrid() : void - setDisplayImage() : void - setDisplayPoints() : void - setDraggGraphEnabled() : void - setDraggPointsEnabled() : void - setDraggTolerance() : void - setFirstDerivative() : void - setGridColor() : void - setGridX() : void - setGridY() : void - setImageUri() : void - setJDBCDriver() : void - setJDBCQuery() : void - setJDBCUri() : void - setLastDerivative() : void - setLegendPosition() : void - setLineWidth() : void - setName() : void - setPointsWidth() : void - setRoundPoints() : void - setScaleX() : void - setScaleY() : void - setSnapToGrid() : void - setTipBackground() : void - setTipDelayMillis() : void - setTipForeground() : void - setTypeOfImage() : void - setUseToolTip() : void - setVaxColor() : void - setVlabel() : void - setVlabelColor() : void - setVvaluesAngle() : void - setVaxColor() : void - setVlabel() : void - setVlabelColor() : void - setVvaluesAngle() : void - update() : void - calculate() : void - drawAxis() : void - drawAxisLabels() : void - drawBack() : void - drawGraph() : void - drawGridLines() : void - drawGridValues() : void - drawLegend() : void - drawName() : void - drawPoints() : void - render() : void - directx() : double - directy() : double - inversedx() : double - inversedy() : double
- colors : Color[] - type : int - zoom : double

5.2 Chart UML Diagram

Zoom to at least 360% in order to be able to read the names of the fields, methods, component names and packages.



Please see the following page for the remainder of this diagram.

✚ actionPerformed() : void
✚ Chat() : void
✚ componentHidden() : void
✚ componentMoved() : void
✚ componentResized() : void
✚ componentShown() : void
✚ getAlias() : boolean
✚ getBackground() : Color
✚ getBackgroundDirection() : int
✚ getBaseColor() : boolean
✚ getBorderSpace() : int
✚ getCharBackground() : Color
✚ getCharOutline() : boolean
✚ getCharShadow() : boolean
✚ getCharShown() : boolean
✚ getDisplayOrder() : boolean
✚ getDisplayImage() : boolean
✚ getDisplayValue() : boolean
✚ getFillColor() : boolean
✚ getFont() : Font
✚ getForeground() : Color
✚ getFillColor() : Color
✚ getFontThickness() : float
✚ getImageURL() : String
✚ getInteractive() : boolean
✚ getJDBCDriver() : String
✚ getJDBCQuery() : String
✚ getJDBCURL() : String
✚ getLegendPosition() : int
✚ getLight() : boolean
✚ getMinimumSize() : Dimension
✚ getShadow() : boolean
✚ getShadowColor() : Color
✚ getTransparencyLevel() : int
✚ getTypesOfChat() : int
✚ getTypesOfImage() : int
✚ mouseClicked() : void
✚ mouseDragged() : void
✚ mouseEntered() : void
✚ mouseExited() : void
✚ mouseMoved() : void
✚ mousePressed() : void
✚ mouseReleased() : void
✚ paint() : void
✚ setBackground() : void
✚ setBackgroundDirection() : void
✚ setBaseColor() : void
✚ setBorderSpace() : void
✚ setCharBackground() : void
✚ setCharOutline() : void
✚ setCharShadow() : void
✚ setCharShown() : void
✚ setDisplayOrder() : void
✚ setDisplayImage() : void
✚ setDisplayValue() : void
✚ setFillColor() : void
✚ setFont() : void
✚ setForeground() : void
✚ setFillColor() : void
✚ setFontThickness() : void
✚ setImageURL() : void
✚ setJDBCDriver() : void
✚ setJDBCQuery() : void
✚ setJDBCURL() : void
✚ setLegendPosition() : void
✚ setShadowColor() : void
✚ setTransparencyLevel() : void
✚ setTypesOfChat() : void
✚ setTypesOfImage() : void
✚ update() : void
✚ getFunctionColor() : Color
✚ updateLegend() : void
✚ updateTable() : void
✚ readObject() : void
✚ writeObject() : void

✚ alias : boolean
✚ interactive : boolean
✚ light : boolean
✚ name : String
✚ shadow : boolean
✚ space : int

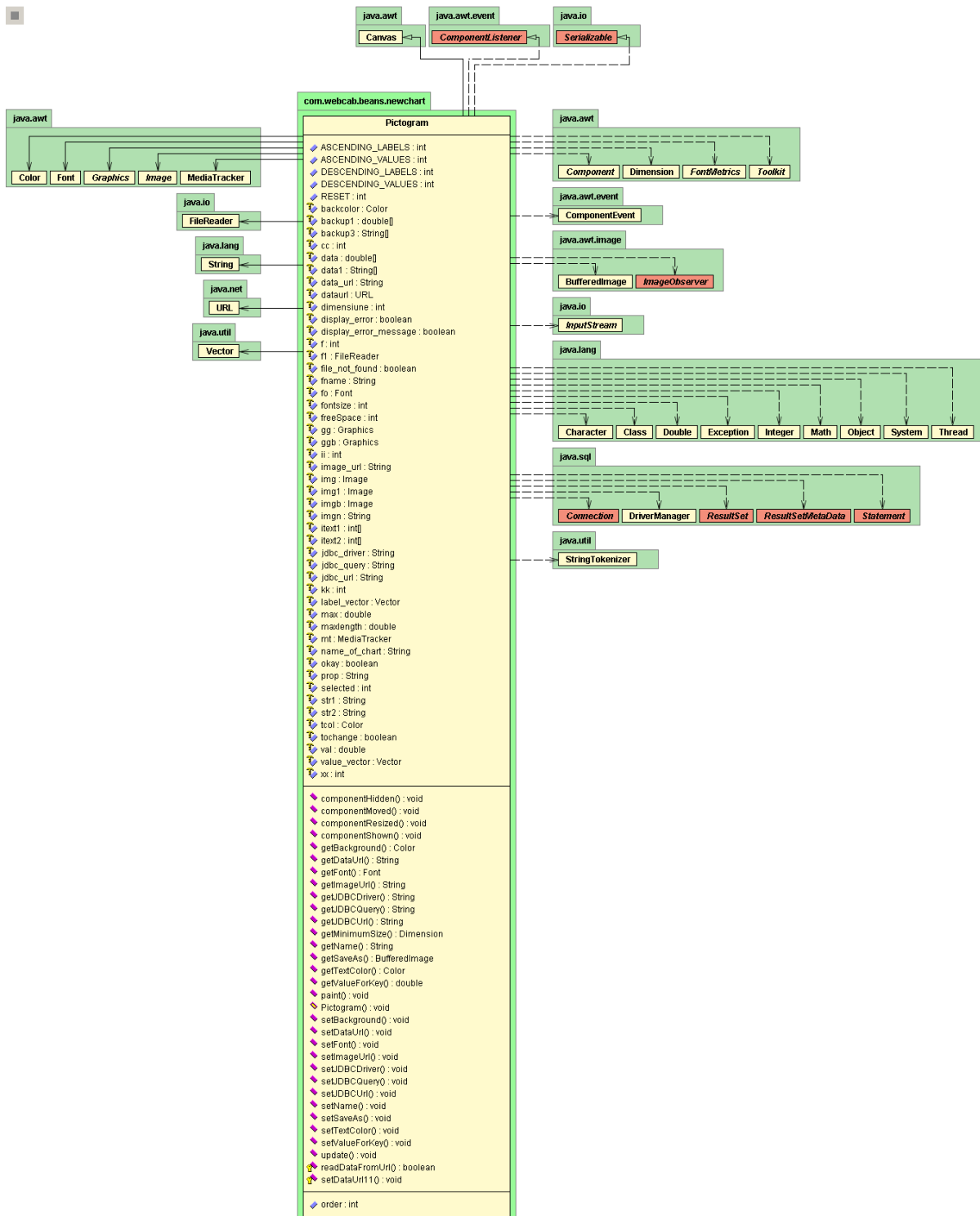
Zoom to at least 360% in order to be able to read the names of the fields, methods, component names and packages.



55

5.4 Pictogram UML Diagram

Zoom to at least 360% in order to be able to read the names of the fields, methods, component names and packages.



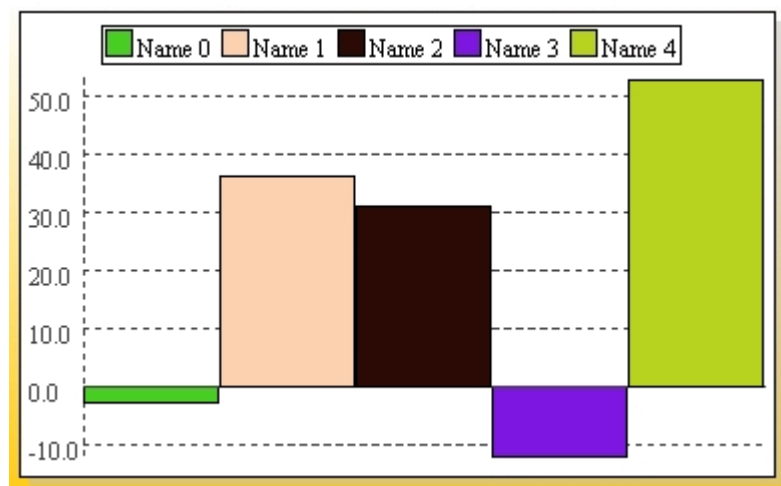
Chapter 6

JGraph Screen Shots

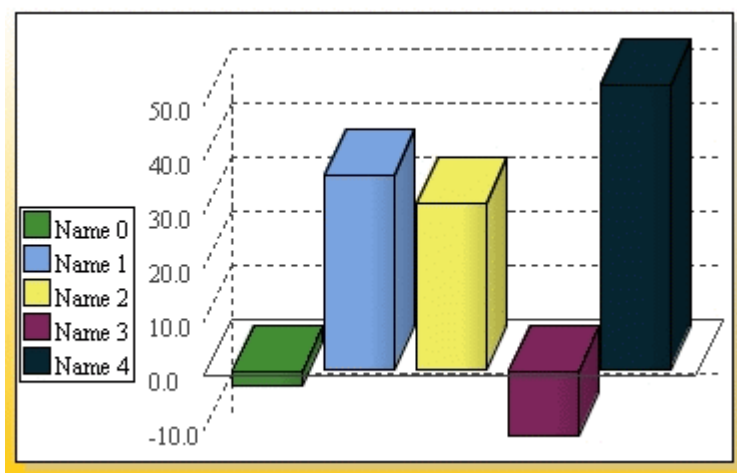
Within this chapter you can find some new features of JGraph v2.0 by looking at the below Screen Shots.

6.1 The Chart Component

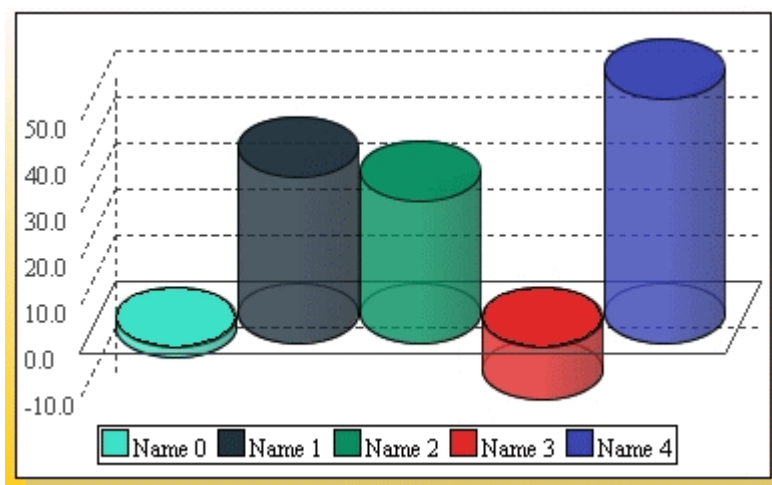
Several types of bar charts are available for displaying the different values from your database, also displays multiple data series simultaneously.



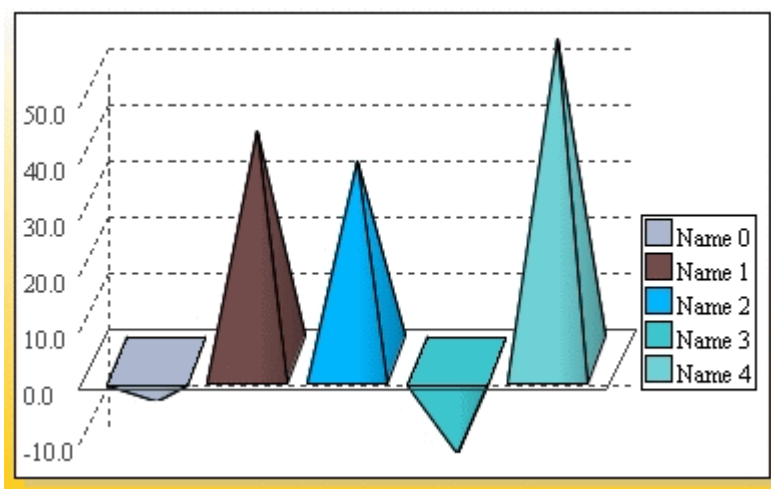
Illustrates the “Bar Chart 2D”



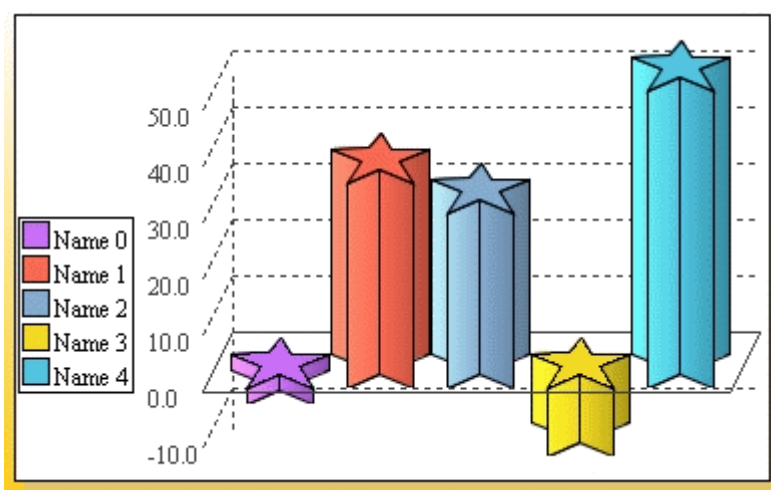
Illustrates the "Bar Chart 2D"



Illustrates the "Cylinder Chart"



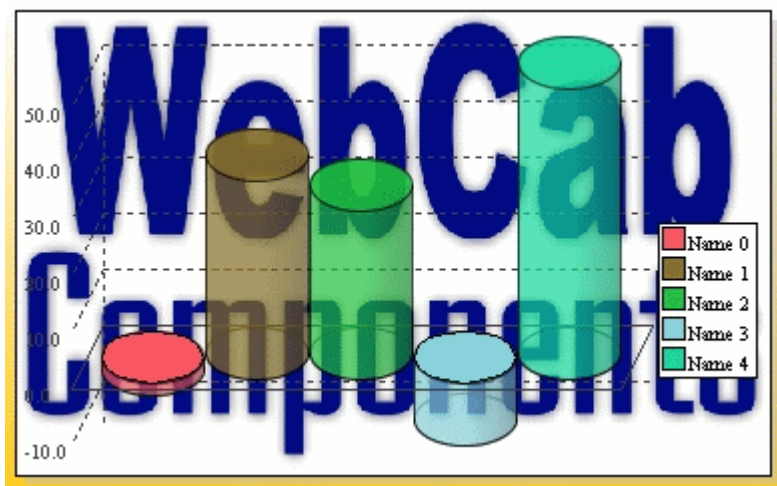
Illustrates the "Pyramid Chart"



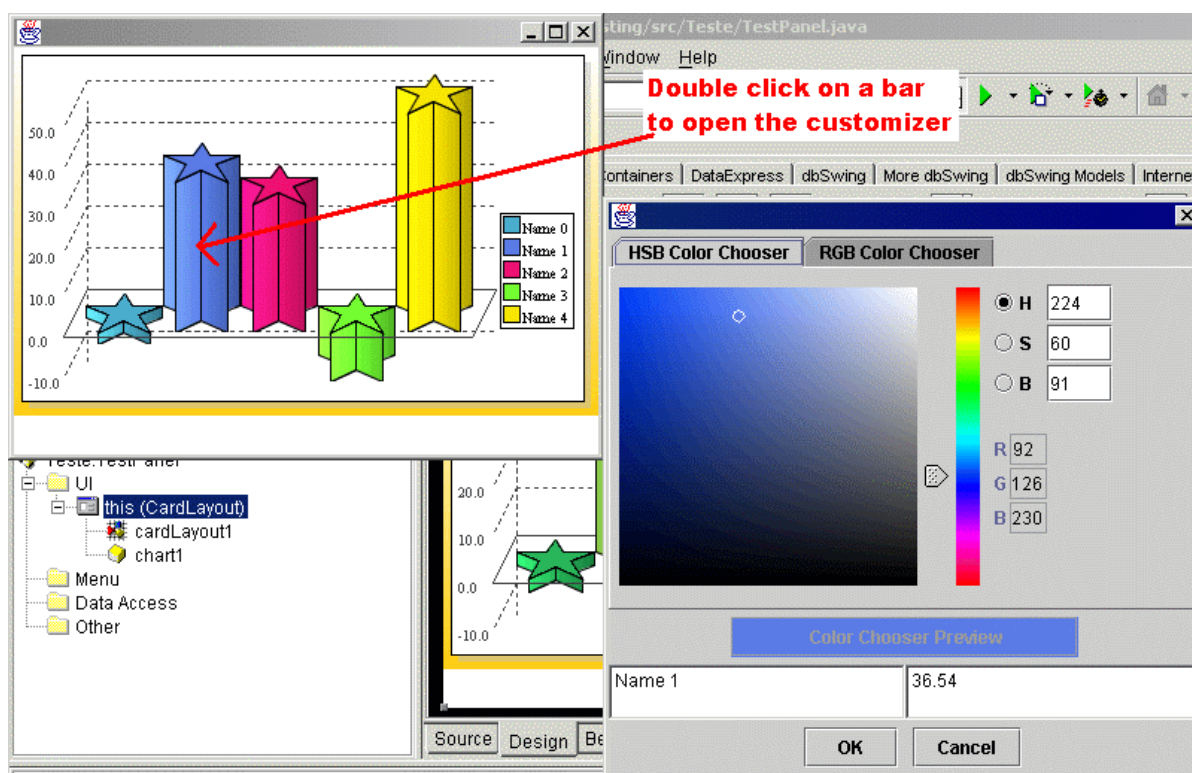
Illustrates the "Star Chart"

This new implemented feature simplifies the work with the databases: JDBC compliant
- connect via JDBC drivers to any JDBC compliant DBMS.

The background of the chart can be changed. It can be either a solid color or an image (JPEG, GIF or PNG) loaded via a URL connection.

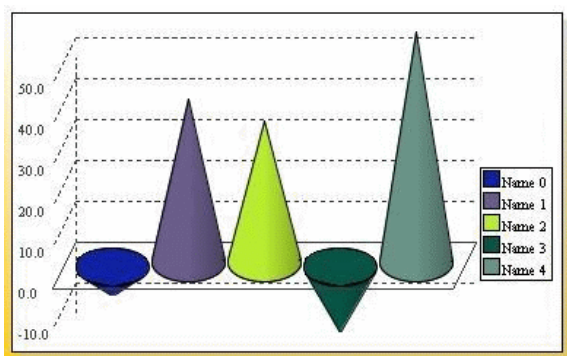


A "Cylinder Chart" using an image as background

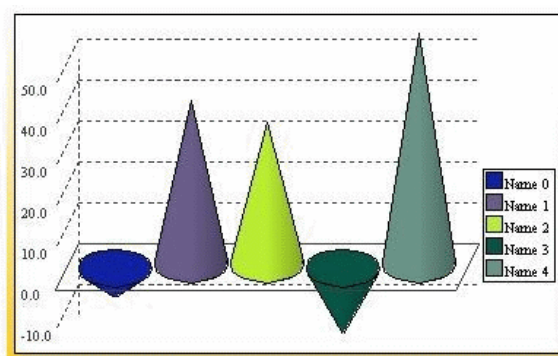


Shows how to open the "Chart Customizer"

Anti-aliasing capabilities can be used when drawing the charts. This makes all the curves and inclined lines look smoother. You may add light effects to 3D perspective charts for a better view.

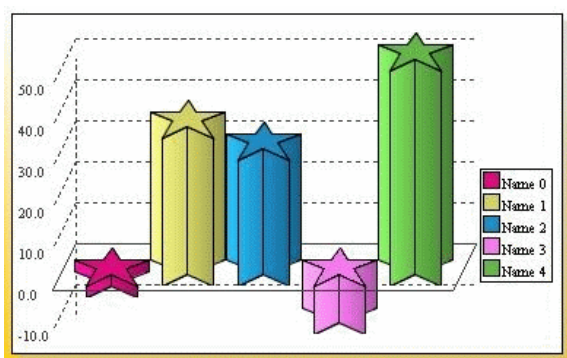


With "Antialias"



Without "Antialias"

Anti-aliasing



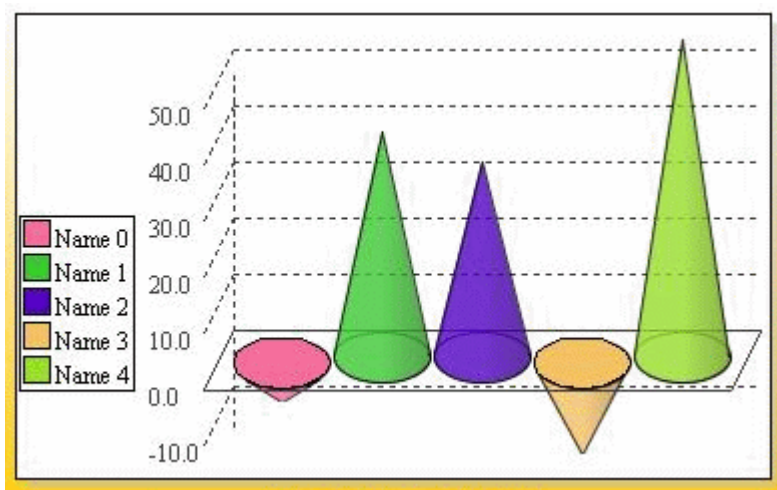
With "Light Effect"



Without "Light Effect"

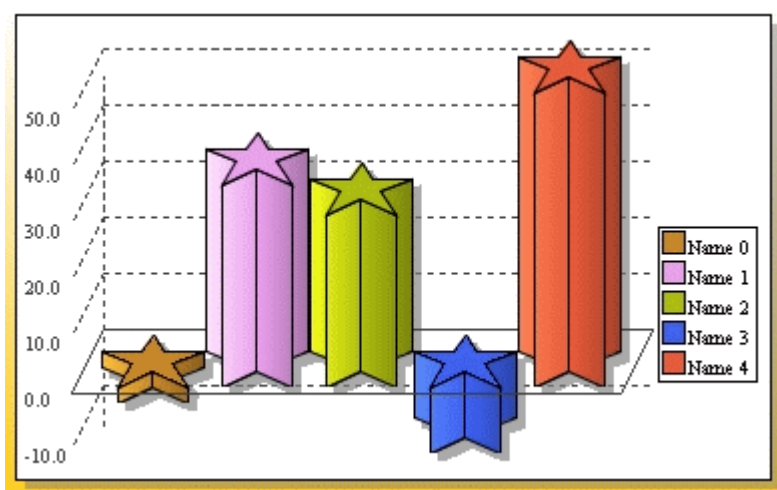
Light effect

The chart can be smoothly varied between different levels of transparency, from being opaque to totally transparent.

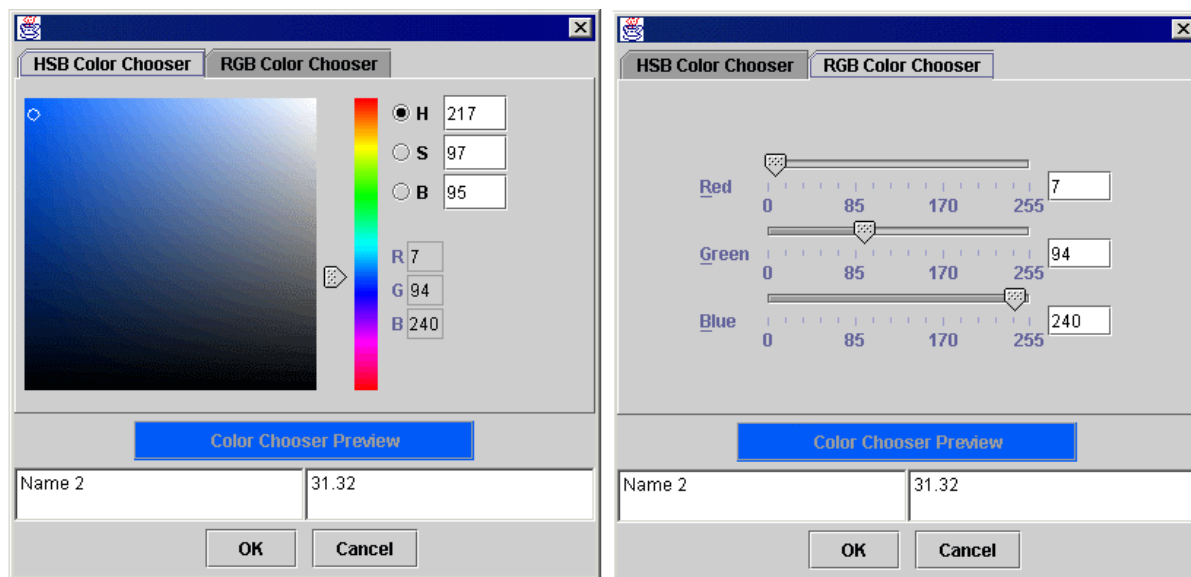


A "Cone Chart" with transparency

Other new feature is the possibility to add shadows to the charts.



"Star Chart" with shadows



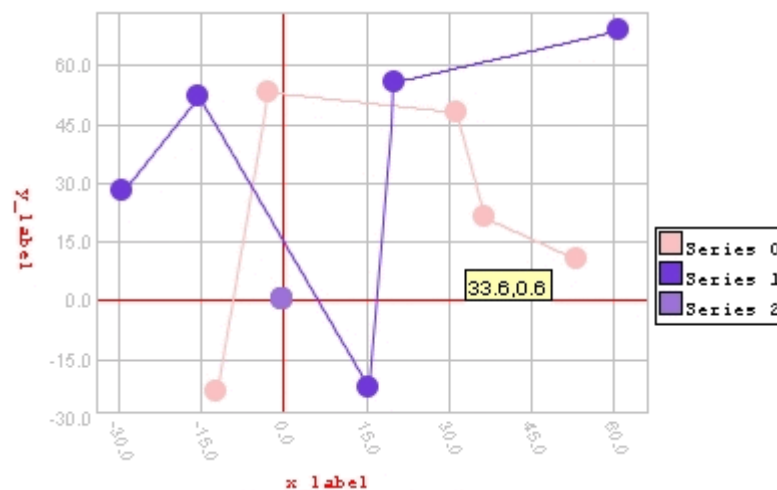
Available customizers

Choose from a large number of fonts and colors inside the Chart component to personalize your charts.

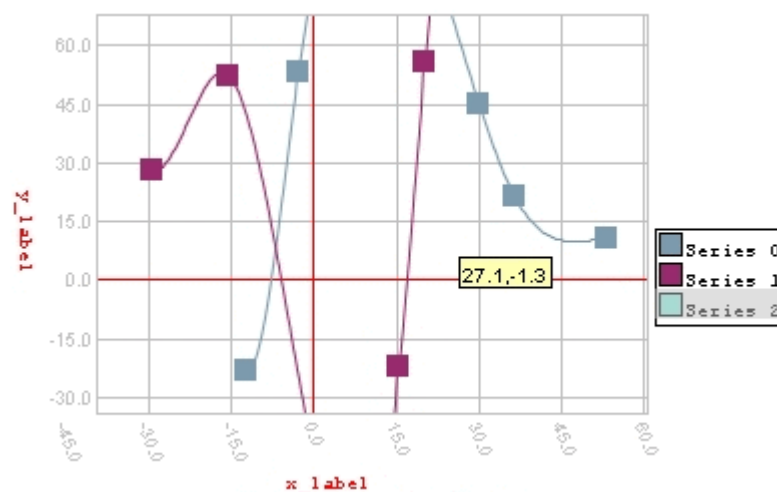
For easy value reading a grid can be used. Finally the chart can be exported as a JPEG with selectable image quality.

6.2 The Graph Component

You can select different type of interpolation methods to represent your data.

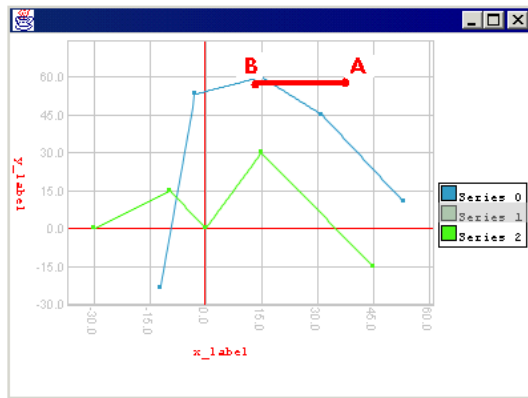


Graph with “No Interpolation” and round points

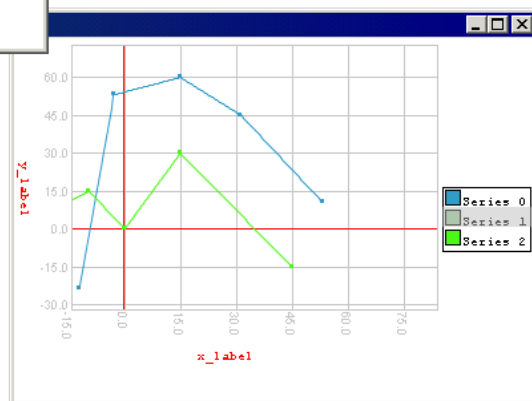


Graph with “Cubic Splines Interpolation” and square points

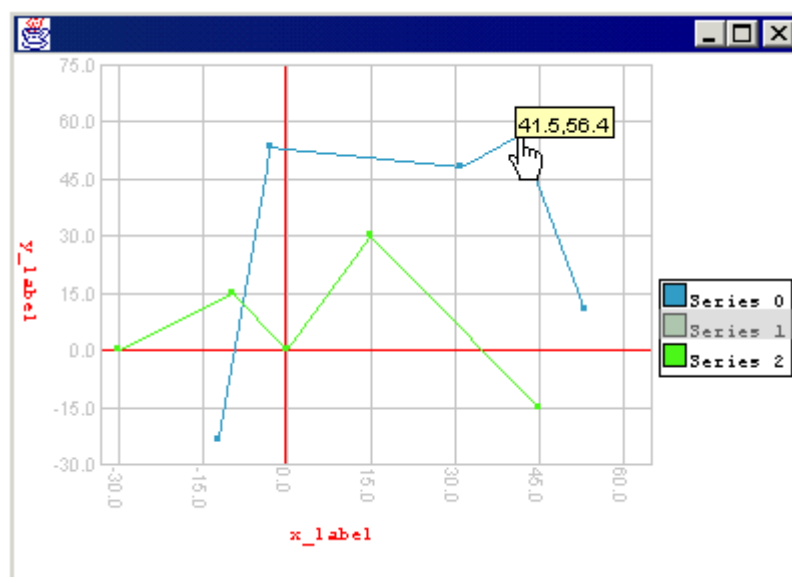
The user can change the data set by dragging the interpolation points. The graph will change in real time to represent the new values. The whole graph can be dragged using the mouse. To observe a certain part of the chart the user can choose between two powerful techniques, zooming and scaling.



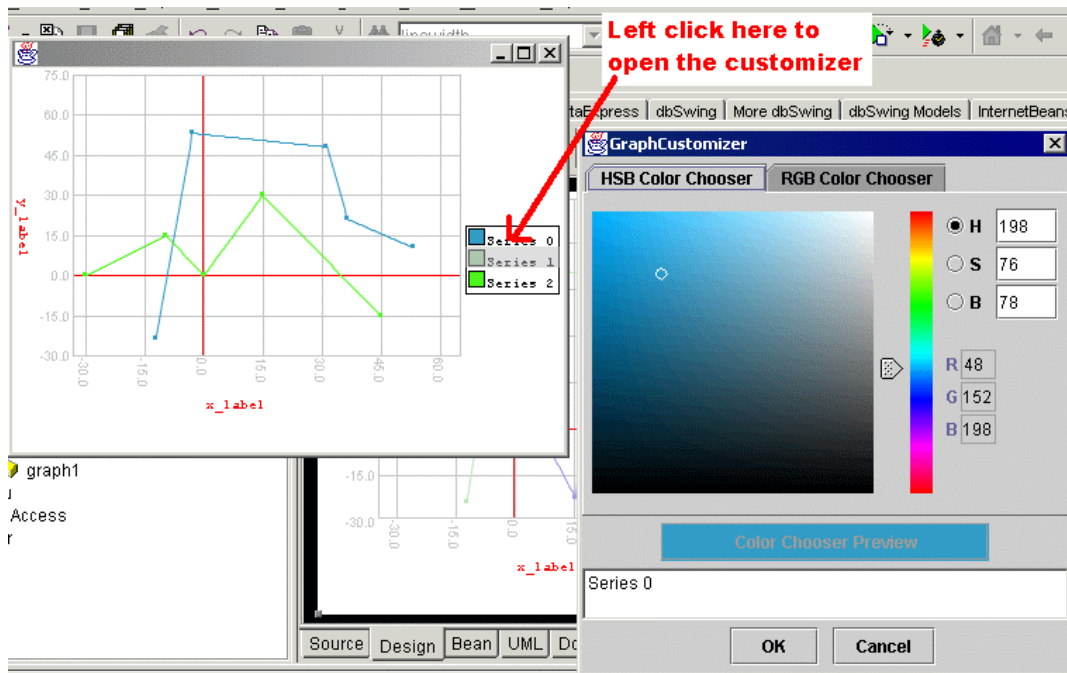
Dragging the mouse pointer from A to B will move the whole graph.



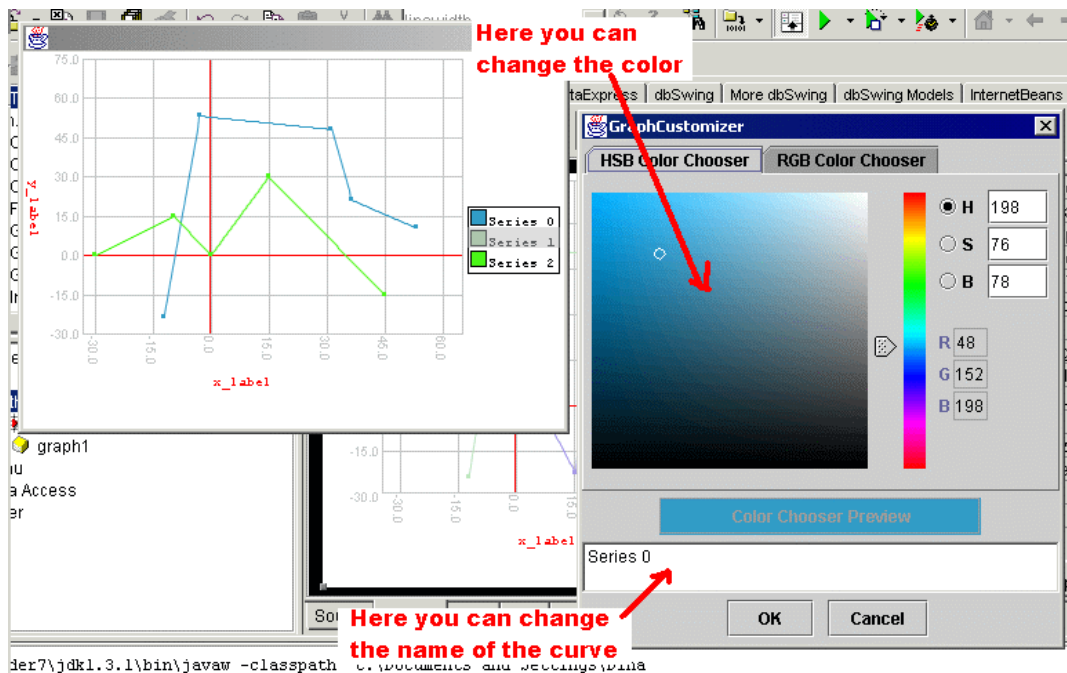
Dragging example



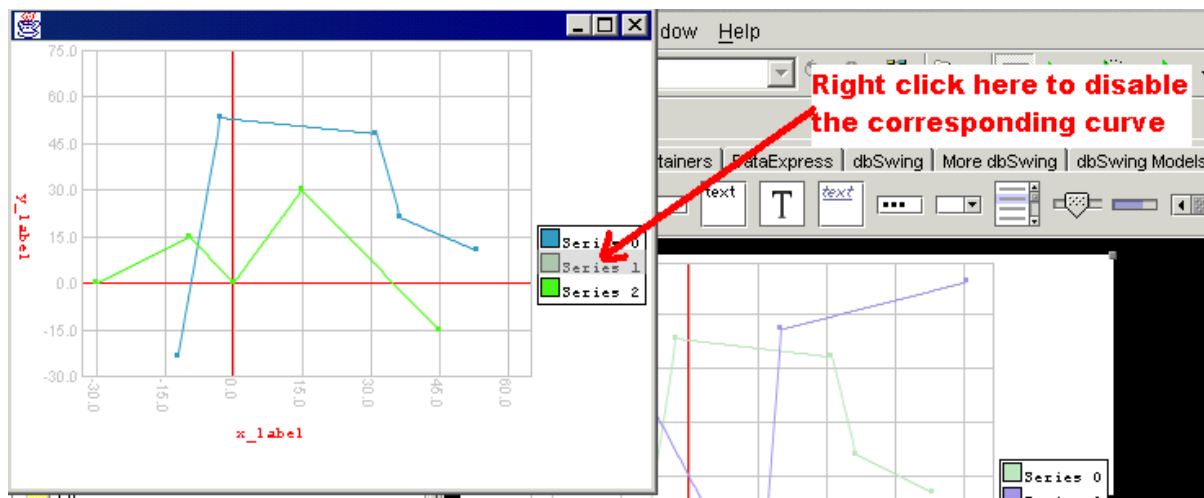
JGraph can display multiple data series, the user having the possibility to disable one curve or more. The color and labels of the curves can be changed using a fully featured customizer.



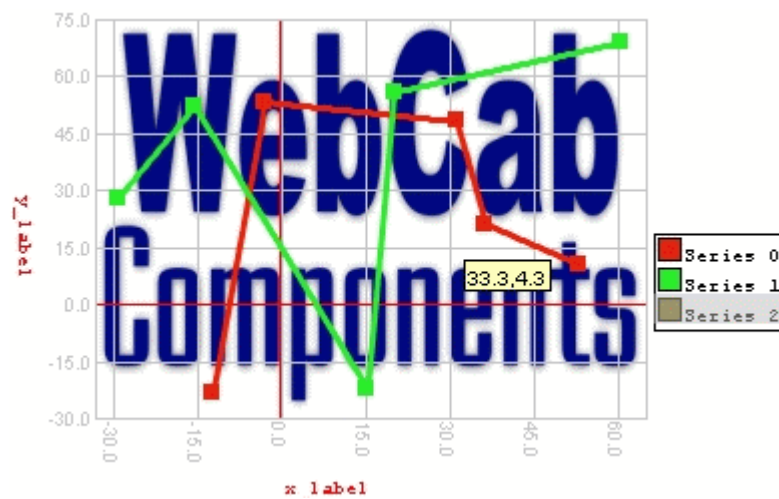
How to open the "Graph Customizer"



Graph customizer explained



How to disable curves within a graph

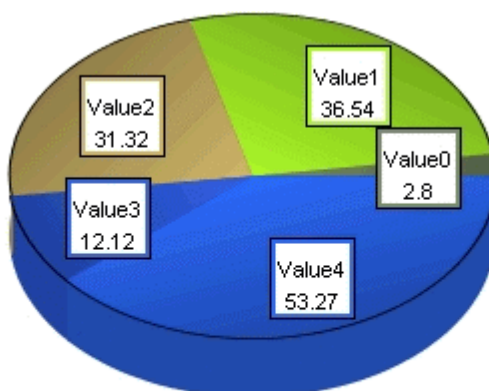


Graph with image on the background

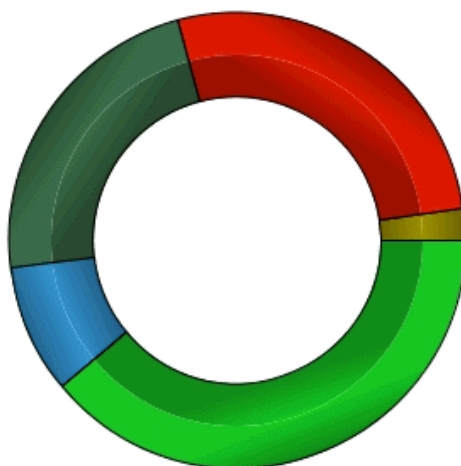
6.3 The Pie Component



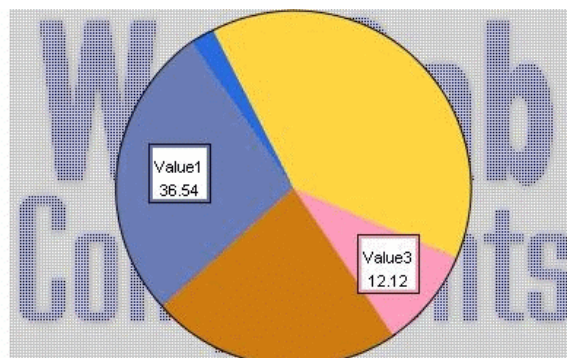
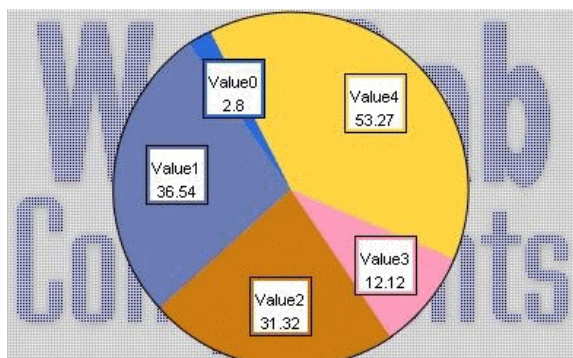
Pie 2D Chart



Pie 3D Chart

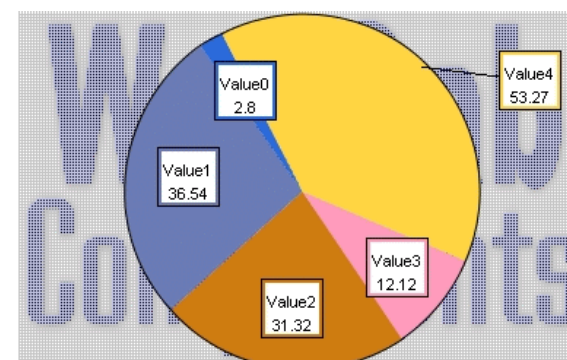
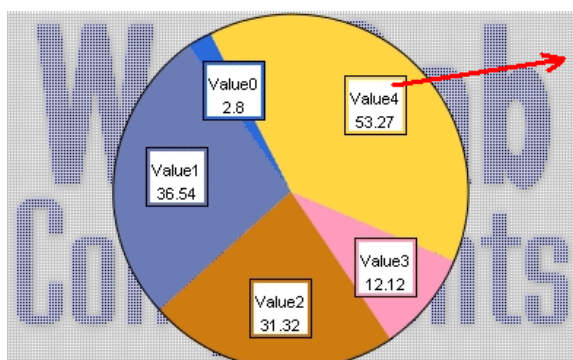


Donut Chart



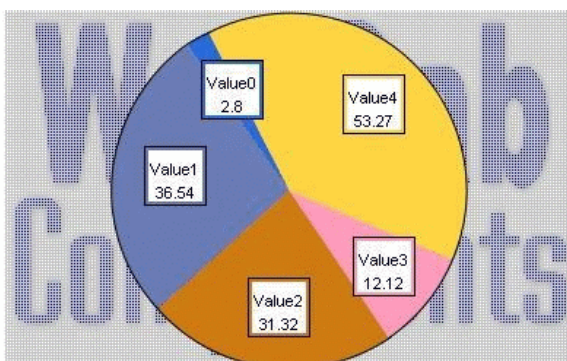
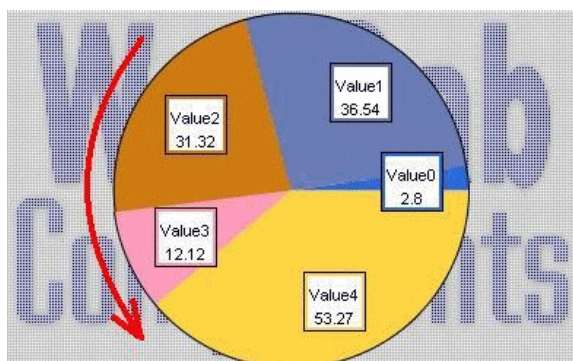
**Right clicking on a label will make the label disappear.
Right clicking on its corresponding component
will make the label visible again.**

How to activate/deactivate labels



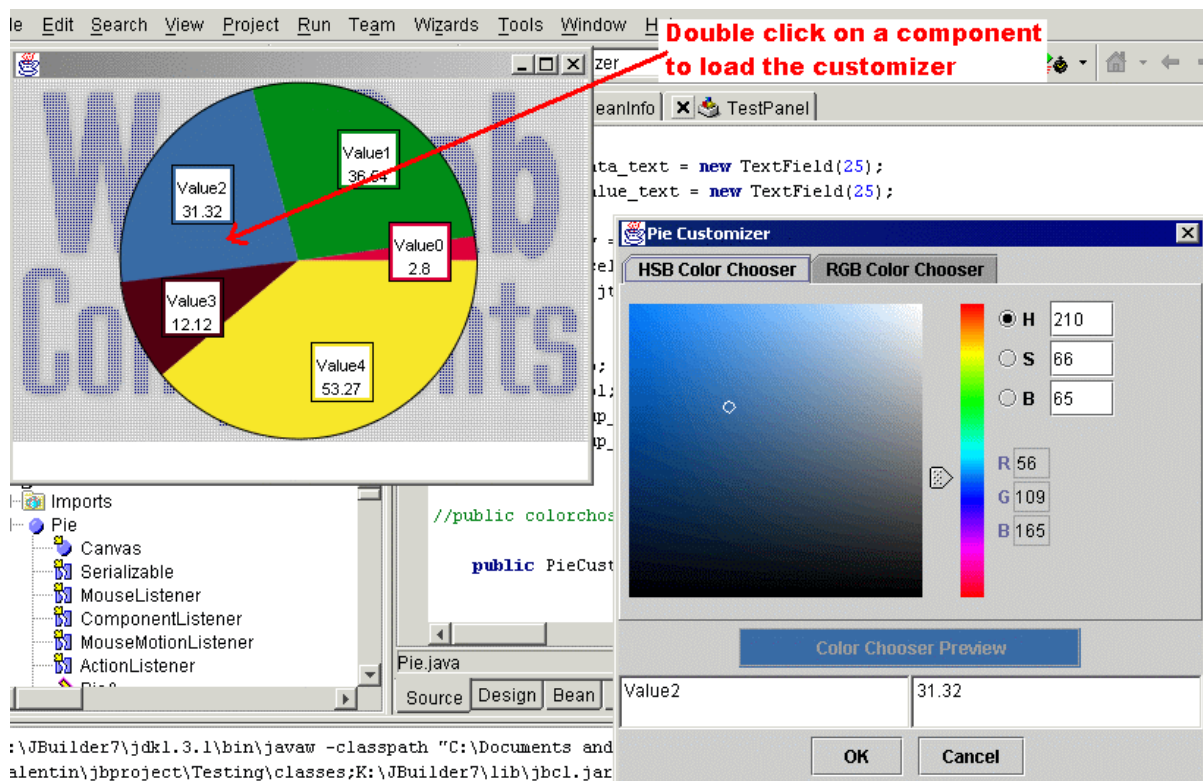
**Labels can be dragged with the mouse. To bring back a label to its default position,
hold the "Alt" key pressed and click on the label's corresponding component.
In this example the label was dragged along the red line**

How to move labels



**Pies can be rotated by holding the "Shift" key pressed and dragging
the mouse pointer around the center of the pie.**

How to rotate pies



Pie Customizer

6.4 The Pictogram Component



Pictogram example

Chapter 7

Guide to WebCab Components

7.1 The Company

WebCab is a privately owned British company that has built business solutions since its inception in 1999. We continue to refine and develop our Mathematical and Financial Framework which we have implemented on the J2SE, J2EE and .NET platforms.

7.2 Presentation of Products

We aim to provide good quality, useful information to help you decide which J2EE application best suits your development needs. WebCab is committed to honesty and realism when presenting our products. In order to achieve this a detailed, clear and factual style for presentation is adopted within our documentation and marketing material.

7.3 Supported Clients, IDEs, Containers and DBMSs

We support all major development, server and client side technologies. Each product contains detailed examples and advice concerning the integration of our enterprise applications within existing infrastructure. Our documentation provides the information that the developer needs to get their applications up and running as quickly and as easily as possible.

We detail exactly how the developer can use our J2EE Applications with the following technologies:

- Client containers - Internet Explorer, Netscape Navigator, Opera
- IDEs - JBuilder, Eclipse, BEA Studio, Sun ONE (formerly Forte For Java), IBM VisualAge, Oracle JDeveloper
- Client Side Technologies - Application Clients, Servlet, JSP, Applets
- EJB containers - IBM WebSphere, BEA WebLogic, Oracle 9iAS, Sun ONE AppServer, Borland AppServer, Sybase EAServer, Ironflare Orion, JBoss

- DBMS - Oracle, IBM DB2, SQL Server, Sybase, MySQL
- Platforms - Windows XP/2000/NT, Linux, Solaris, IBM AIX

For these technologies we include all the deployment descriptors, installation scripts and template examples which will help the developer quickly and easily assemble their multi-tier enterprise application.

7.4 Transparent Functionality

All technical and business intelligence incorporated within our products is described within the associated documentation. This allows the developer to see the nature of the methodology implemented within our proprietary algorithms.

7.5 Code Conventions

WebCab adheres to the ‘Code Conventions for the Java Programming Language’ as specified on April 20, 1999 by Sun Microsystems Inc. These conventions cover filenames, file organization, indentation, comments, declarations, statements, white space, naming conventions and programming practices. Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.

7.6 Company Culture and Activity

WebCab Components is focused solely on the production of high quality software modules within our Financial and Mathematical Framework. The WebCab team contains a wide range of expertise and experience from the academic, investment banking and software development worlds.

Within the company there exists significant internal competitive pressures with constant peer review and evaluation, resulting in higher quality products, which adhere to high professional standards and offer extended functionality.

7.7 Product Life cycle

We continuously add value to our products by evolving them according to new J2EE specifications, customer feedback and market demands. We give particular emphasis to incorporating our clients suggested modifications and additions into our product development cycle.

7.8 Support, Warranty and Upgrades

WebCab warrants that each component will perform substantially in accordance with the accompanying written material. We provide with all our products without charge sixty (60) days product support services including fixing bugs, compatibility issues and other technical support issues.

All maintenance updates (including service packs) will be distributed free of additional license cost. New version releases of this product will be offered to present licensee holders at a greatly reduced rate.

Dr Ben Fairfax

Founder and CEO

WebCab Components - From Developer, To Developer

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. .NET is a trademark of Microsoft Corporation in the U.S. and other countries