JCCKit Configuration Manual

Version 1.1

Author:
Franz-Josef Elmer

14th December 2004

Contents

Chapter 1

Concept

1.1 Introduction

JCCKit is highly configurable. Most classes have constructors based on an instance of `ConfigParameters`. Configuration parameters are hierarchically organized sets of key-value pairs. The hierarchy allows to create a complex object where inside the constructor less complex objects are created based on sub-trees of the hierarchy of configuration parameters.

The key is the name of the parameter and the value is a concrete value of that parameter. The key is a string without whitespaces and '/'. It is a unique identifier. The value is either an atomic value (like a string or a number) or a composite one (e.g. a `ConfigParameters` object).

Configuration parameters are programmatically specified by using an implementation of `ConfigData`. In most cases one will use `PorpertiesBasedConfigData` wrapping a `Properties` object.[1] Here is an example of typical usage:

```
Properties props = new Properties();
props.put("plot/coordinateSystem/xAxis/minimum", "-20");
props.put("plot/coordinateSystem/xAxis/maximum", "20");
ConfigParameters config
       = new ConfigParameters(new PropertiesBasedConfigData(props));
GraphicsPlotCanvas canvas = new GraphicsPlotCanvas(config);
```

A `Properties` object is a flat set of key-value pairs where both, key and value, are strings. In order to map hierarchically organized key-value pairs onto such a flat set the keys in a `Properties` object are build in accordance of the following convention:

- The `ConfigParameters` key is preceeded by the keys of its ancestors.
- The ancestor keys are separated by slashes (i.e. '/').

[1] For ⇒Plot Applets `AppletBasedConfigData` is used. It is similar to `PorpertiesBasedConfigData`: The key-value pairs are the applet parameters where the key is specified by the name attribute. The difference between both types of `ConfigDatas` is only the different syntax of applet parameters and `.properties` files. Otherwise the meaning of they keys and values are identical.

- The left-most key denotes the child of the root `ConfigParameters` object.

Example:

```
plot/coordinateSystem/origin
```

This denotes the parameter named `origin` which is a parameter of the composite parameter `coordinateSystem`. And `coordinateSystem` is the child of `plot` which is the child of the root `ConfigParameters` object.

In the next section the sophisticated concept of parameter inheritance is explained. It allows to avoid duplicated definitions in complex configurations. Section 1.3 explains the various types of parameters. Chapter 2 is the main part of this manual. It explains all configuration parameters. Finally, an index of all parameter keys allows to find quickly the meaning of certain parameters in existing configurations.

On the JCCKit Home page http://jcckit.sourceforge.net various examples for the usage of the configuration parameters can be found. Especially the study of the static examples is recommended: Just click on such an example. A new browser window pops up showing the example as a ⇒Plot Applet. The corresponding parameters can be found by making the HTML source of this popup visible.

1.2 Parameter Inheritance

`ConfigParameters` based on `FlatConfigData` (like `PorpertiesBasedConfigData` and `AppletBasedConfigData`) allow inheritance of subtrees in the parameter hierarchy. This is done by the following syntax:

<parameters>/ = <default parameters>/

This means, that the subtree <parameters> inherits all parameters from the subtree <default parameters>. Note the '/' at the end of key and value.

Example:

```
plot/curveFactory/definitions = def1 def2
plot/curveFactory/def1/ = default/
plot/curveFactory/def2/ = default/
default/symbolFactory/className = jcckit.plot.BarFactory
default/symbolFactory/size = 0.07
default/symbolFactory/attributes/lineColor = 0
```

is equivalent to

```
plot/curveFactory/definitions = def1 def2
plot/curveFactory/def1/symbolFactory/className = jcckit.plot.BarFactory
plot/curveFactory/def1/symbolFactory/size = 0.07
plot/curveFactory/def1/symbolFactory/attributes/lineColor = 0
plot/curveFactory/def2/symbolFactory/className = jcckit.plot.BarFactory
plot/curveFactory/def2/symbolFactory/size = 0.07
plot/curveFactory/def2/symbolFactory/attributes/lineColor = 0
```


Parameters can inherit from a parameter who also inherit from another parameter[2].

Example: In the example from above `attributes` should inherit from some `defaultAttributes` with an equivalent result:

```
plot/curveFactory/definitions = def1 def2
plot/curveFactory/def1/ = default/
plot/curveFactory/def2/ = default/
default/symbolFactory/className = jcckit.plot.BarFactory
default/symbolFactory/size = 0.07
default/symbolFactory/attributes/ = defaultAttributes/
defaultAttributes/lineColor = 0
```

Inheritance means also extension and overriding.

1.2.1 Extension

One can add new parameters to a parameter container who inherits from another container. The following example shows this:

```
plot/curveFactory/definitions = def1 def2
plot/curveFactory/def1/ = default/
plot/curveFactory/def2/ = default/
plot/curveFactory/def2/withLine = false
default/symbolFactory/className = jcckit.plot.BarFactory
default/symbolFactory/size = 0.07
default/symbolFactory/attributes/lineColor = 0
```

is equivalent to

```
plot/curveFactory/definitions = def1 def2
plot/curveFactory/def1/symbolFactory/className = jcckit.plot.BarFactory
plot/curveFactory/def1/symbolFactory/size = 0.07
plot/curveFactory/def1/symbolFactory/attributes/lineColor = 0
plot/curveFactory/def2/symbolFactory/className = jcckit.plot.BarFactory
plot/curveFactory/def2/symbolFactory/size = 0.07
plot/curveFactory/def2/symbolFactory/attributes/lineColor = 0
plot/curveFactory/def2/withLine = false
```

1.2.2 Overriding

One can also redefine an inherited parameter as in the following example:

```
plot/curveFactory/definitions = def1 def2
plot/curveFactory/def1/ = default/
plot/curveFactory/def2/ = default/
plot/curveFactory/def2/size = 0.05
```

[2] This chain of inheritance is limited to twenty steps.


```
default/symbolFactory/className = jcckit.plot.BarFactory
default/symbolFactory/size = 0.07
default/symbolFactory/attributes/lineColor = 0
```

is equivalent to

```
plot/curveFactory/definitions = def1 def2
plot/curveFactory/def1/symbolFactory/className = jcckit.plot.BarFactory
plot/curveFactory/def1/symbolFactory/size = 0.07
plot/curveFactory/def1/symbolFactory/attributes/lineColor = 0
plot/curveFactory/def2/symbolFactory/className = jcckit.plot.BarFactory
plot/curveFactory/def2/symbolFactory/size = 0.05
plot/curveFactory/def2/symbolFactory/attributes/lineColor = 0
```

1.3 Parameter Types

A parameter is not just a simple textual value. A parameter has a certain *type*. The set of types can be divided into two groups: Atomic types and container types. Atomic types belong to only one parameter whereas container types group one or more parameters. For example, parameter `size` in the previous section is an atomic parameter and `attributes` is a container parameter.

1.3.1 Atomic Types

Type	Description of valid values
boolean	Only `true` or `false` are allowed.
integer	Signed integer number in decimal, octal, or hexadecimal notation. The different bases are recognized by the following beginning of a number (sign not included): **Decimal:** Begins with a non-zero decimal digit. **Octal:** Begins with ’0’. After this initial zero only ’0’-’7’ are allowed. **Hexadecimal:** Begins either with ’0X’, ’0x’, or ’#’. Only ’0’-’9’, ’A’-’F’, or ’a’-’f’ are valid characters.
double	Floating-point number.
double[]	Array of floating-point numbers. The numbers are separated by one or more spaces.
color	Color definition represented by an unsigned integer number. Thus, the same rules hold as for type *integer*. Bits 0-7, 8-15, and 16-23 denote the blue, green, and red color component, respectively.
string	Any character sequence. Leading and trailing spaces will be ignored.

1.3.2 Container Types

Type	Description of valid values

Set	A set of arbitrary many parameters (key-value pairs). In the case of `PorpertiesBasedConfigData` and `AppletBasedConfigData` all flat key-value pairs belong to the same parameter of type *Set* when their full names start with the full name of that parameter plus '/'.
	Example: Parameter `defaultCoordinateSystem` contains
	`defaultCoordinateSystem/ticLabelFormat = %d`
	`defaultCoordinateSystem/ticLabelAttributes/fontSize = 0.03`
	`defaultCoordinateSystem/axisLabelAttributes/fontSize = 0.04`
	`defaultCoordinateSystem/axisLabelAttributes/fontStyle = bold`

Set[]	Sequence of parameters of type *Set*. A so-called *sequence parameter* of type *string* contains a sequence of space-separated names of parameters of type *Set*. Even though names can be chosen arbitrarily they must be unique in the set.
	Example: Sequence parameter `definitions` defines two *Sets*.
	`plot/curveFactory/definitions = cf1 cf2`
	`plot/curveFactory/cf1/withLine = true`
	`plot/curveFactory/cf1/symbolFactory/size = 0.05`
	`plot/curveFactory/cf2/symbolFactory/size = 0.05`

Object	A parameter of type *Set* which contains a parameter of type *string* named `className` which specifies fully-qualified class name of a the object.
	Example: Parameter `default/symbolFactory` contains
	`default/symbolFactory/className = jcckit.plot.BarFactory`
	`default/symbolFactory/stacked = true`
	`default/symbolFactory/size = 0.07`
	`default/symbolFactory/horizontalBars = true`
	`default/symbolFactory/attributes/lineColor = #ff8040`
	Usually, the allowed classes have to implement a certain interface or to subclass a certain (abstract) class.

Chapter 2

Dictionary

The configuration dictionary explains all possible configuration parameters. It is organized by alphabetically ordered entries. An entry belongs to a certain parameter container.

There are four different types of dictionary entries recognizable by their different structures:

Set: Defines a parameter of type *Set*. It has the following elements:

- Parameter type.
- Synopsis.
- Table of all parameters of the set. The table has the following columns:
  - *Name & Default Value*: The unique name (i.e., key) of the parameter and optionally a '=' followed by the default value or a description of it.
  - *Type*: One of the types described in Sec. 1.3.
  - *Mandatory?*: Shows whether this parameter is mandatory or not. Missing or misspelled mandatory parameter causes an exception.
  - *Description*: The meaning in case of an atomic parameter. For a container parameter a reference is made to another dictionary entry.

Object: Defines a parameter of type *Object*. It has the same elements as a dictionary entry of type **Set**. In addition it has the mandatory parameter `className` which denotes the fully-qualified class name of the object.

Class Type: Defines all parameters of type *Object* which can be used at all places where a reference is made to such an entry. It has the following elements:

- Class type. That is, the fully-qualified class name either of a superclass or an interface which all concrete classes have to subclass or implement, respectively.
- Concrete classes: List of dictionary entries of type **Object**.
- Synopsis.

2.1 Axis Parameters

Parameter type: *Object*

Class parameter: `className = jcckit.plot.AxisParameters`

Synopsis: Axis parameters of an axis of a ⇒Cartesian Coordinate System.

NOTE: The default values for x- and y-axis may differ. They are noted as *x-axis default* / *y-axis default*.

Name & Default Value	Type	Mandatory?	Description
`logScale = false`	*boolean*	no	If `true` the axis will be logarithmic. Otherwise the axis is linear.
`minimum = 0`	*double*	no	The corresponding data value of one end of the axis.
`maximum = 1`	*double*	no	The corresponding data value of the other end of the axis.
`axisLength = 0.8 / 0.45`	*double*	no	Length of the axis in device-independent units.
`axisAttributes` = *default values of Shape Attributes*	*Set*	no	⇒Shape Attributes of the axis box.
`axisLabel = x / y`	*string*	no	Axis label.
`axisLabelPosition = 0 -0.05 / -0.1 0`	*double[]*	no	Position of the axis label relative to the center of the axis.
`axisLabelAttributes` = *default values of Basic Attributes*	*Set*	no	⇒Basic Graphic Attributes of the axis label.
`automaticTicCalculation = true`	*boolean*	no	If `true` the axis tics will be calculated automatically.
`minimumTic` = *result from automatic calculation*	*double*	no	The corresponding data value of the tic nearest the minimum end of the axis. Will be ignored if `automaticTicCalculation = true`.
`maximumTic` = *result from automatic calculation*	*double*	no	The corresponding data value of the tic nearest the maximum end of the axis. Will be ignored if `automaticTicCalculation = true`.
`numberOfTics` = *result from automatic calculation*	*double*	no	Number of tics. The tics between the minimum and maximum tic are spaced equidistantly. Will be ignored if `automaticTicCalculation = true`.
`ticLength = 0.01`	*double*	no	Length of the tics in device-independent units. Negative/positive values mean tics inside/outside the axis box.

Name & Default Value	Type	Mandatory?	Description
`ticAttributes` = *default values of Shape Attributes*	*Set*	no	⇒Shape Attributes of tics.
`ticLabelFormat` = `%1.1f`	*string / Object*	no	Either a printf-like number format definition or ⇒Tic Label Map. Note, that an empty string means that tic labels are dropped.
`ticLabelPosition` = `0 -0.01/-0.01 0`	*double[]*	no	Position of the anchor of the tic label relative to the tic position on the axis.
`ticLabelAttributes` = *default values of Basic Graphic Attributes*	*Set*	no	⇒Basic Graphic Attributes of the tic labels.
`grid` = `false`	*boolean*	no	If `true` grid lines will be drawn through the axis tics.
`gridAttributes` = *default values of Shape Attributes*	*Set*	no	⇒Shape Attributes of grid lines.

2.2 Bar

Parameter type: *Object*

Class parameter: `className = jcckit.plot.BarFactory`

Synopsis: ⇒Symbol Factory for bars. For the usage see the bar-chart examples on the JCCKit home page http://jcckit.sourceforge.net.

Name & Default Value	Type	Mandatory?	Description
`size` = `0.01`	*double*	no	Size of the symbol in device-independent units.
`attributes`	*Object*	no	⇒Graphic Attributes of the symbol.
`stacked` = `false`	*boolean*	no	If `true` the bars of several curves will be stacked.
`horizontalBars` = `false`	*boolean*	no	If `true` horizontal bars will be drawn. Otherwise vertical bars are drawn.

2.3 Basic Graphic Attributes

Parameter type: *Object*

Class parameter: `className = jcckit.graphic.BasicGraphicAttributes`

Synopsis: Basic attributes of elementary graphic objects and texts in a ⇒Plot.

Name & Default Value	Type	Mandatory?	Description
`fillColor` = *no filling*	*color*	no	The color a shape will be filled up.
`lineColor` = *foreground color*	*color*	no	The line color.
`lineThickness` = `0`	*double*	no	The line thickness in device-independent units.
`linePattern` = *solid line*	*double[]*	no	The line pattern.
`textColor` = *foreground color*	*color*	no	The text color.
`fontName` = `SansSerif`	*string*	no	The name of the text font. Possible values are `Serif`, `SansSerif`, and `Monospaced`.
`fontStyle` = `normal`	*string*	no	The font style. Possible values are `normal`, `bold`, `italic`, and `bold italic`.
`fontSize` = *default* **device-dependent** *size*	*color*	no	The font size in units of the device-independent units.
`horizontalAnchor` = `left`	*string*	no	Anchor for horizontal text position. Possible values are `left`, `center`, and `right`.
`verticalAnchor` = `center`	*string*	no	Anchor for vertical text position. Possible values are `top`, `center`, and `bottom`.
`orientationAngle` = `0`	*double*	no	The orientation angle of the text (in degree). Zero means normal orientation whereas a positive value means a rotation in counter-clockwise direction.

2.4 Cartesian Coordinate System

Parameter type: *Object*

Class parameter: `className = jcckit.plot.CartesianCoordinateSystem`[1]

Synopsis: Definition of a Cartesian coordinate system for a ⇒Plot.

[1] Parameter `className` could be omitted if used in conjunction with ⇒Plot

Name & Default Value	Type	Mandatory?	Description
`origin = 0.15 0.1`	*double[]*	no	Position (in device-independent units) of the lower-left corner of the axis box.
`xAxis`	*Set*	no	Definition of ⇒Axis Parameters for the x axis.
`yAxis`	*Set*	no	Definition of ⇒Axis Parameters for the y axis.

2.5 Circle

Parameter type: *Object*

Class parameter: `className = jcckit.plot.CircleSymbolFactory`

Synopsis: ⇒Symbol Factory for circles.

Name & Default Value	Type	Mandatory?	Description
`size = 0.01`	*double*	no	Size of the symbol in device-independent units.
`attributes`	*Object*	no	⇒Graphic Attributes of the symbol.

2.6 Coordinate System

Class type: `jcckit.plot.CoordinateSystem`

Concrete classes: ⇒Cartesian Coordinate System

Synopsis: Definition of a coordinate system of a ⇒Plot.

2.7 Curve

Parameter type: *Object*

Class parameter: `className = jcckit.plot.SimpleCurve`[2]

Synopsis: Properties of a curve of a ⇒Plot.

[2] Parameter `className` could be omitted if used in conjunction with ⇒Simple Curve Factory

Name & Default Value	Type	Mandatory?	Description
`symbolFactory`	*Object*	no	⇒Symbol Factory for curve symbols. If undefined no symbols will be drawn.
`withLine = true`	*boolean*	no	If `true` the points of the curve are connected by straight lines.
`softClipping = true`	*boolean*	no	If `true` no explicit clipping takes place but the symbol is not drawn if the corresponding curve point is outside the axis box. If `false` the symbol is drawn in any case but it may be clipped by the axis box. Soft-clipping should be set to `false` if the symbols are not located around the curve point (like for bars). This is especially the case when the factory evaluates ⇒Position Hints.
`lineAttributes` = *default Shape Attributes with colors selected from a set of six.*	*Object*	no	⇒Graphic Attributes of the curve.
`initialHintForNextPoint` = *no initial hint*	*Object*	no	Initial ⇒Hint for drawing the first point of the curve.

2.8 Curve Factory

Class type: `jcckit.plot.CurveFactory`

Concrete classes: ⇒Simple Curve Factory

Synopsis: Definition of the properties of a curve in a ⇒Plot.

2.9 Data Curve

Parameter type: *Set*

Synopsis: Sequence of data points defining a two-dimensional curve.

Name & Default Value	Type	Mandatory?	Description
`title` = *undefined*	*string*	no	Title of the curve. Will be used in the ⇒Legend.
`x`	*double[]*	yes	x-coordinates of the curve points.

Name & Default Value	Type	Mandatory?	Description
y	*double[]*	yes	y-coordinates of the curve points.

2.10 Data Plot

Parameter type: *Set*

Synopsis: Sequence of ⇒Data Curves.

Name & Default Value	Type	Mandatory?	Description
curves	*Set[]*	yes	*Sequence parameter* of curve definitions. That is, for each name in the sequence a ⇒Data Curve element in this set has to appear.

2.11 Error Bar

Parameter type: *Object*

Class parameter: `className = jcckit.plot.ErrorBarFactory`

Synopsis: ⇒Symbol Factory for horizontal and/or vertical error bars. For the usage see the error bar example on the JCCKit home page http://jcckit.sourceforge.net.

Name & Default Value	Type	Mandatory?	Description
size = 0	*double*	no	Width of the error bars in device-independent units.
attributes	*Object*	no	⇒Graphic Attributes of the error bars.
symbolFactory	*Object*	no	⇒Symbol Factory for the curve symbol without bars. If undefined only the error bars are drawn.

2.12 Graphic Attributes

Class type: `jcckit.graphic.GraphicAttributes`

Concrete classes: ⇒Shape Attributes, ⇒Basic Graphic Attributes

Synopsis: Attributes of graphical elements of a ⇒Plot.

2.13 Graphics Plot Canvas

Parameter type: *Set*

Synopsis: Convenient definition of a plot canvas for AWT (`jcckit.GraphicsPlotCanvas`, `jcckit.GraphicsPlotCanvas2`) and Swing (`jcckit.GraphicsPlotCanvas2`, `jcckit.Graphics2DPlotCanvas`).

Name & Default Value	Type	Mandatory?	Description
`background` = *depends on the environment*	*color*	no	Background color.
`foreground` = *depends on the environment*	*color*	no	Foreground color.
`doubleBuffering` = `true`	*boolean*	no	If `true` double buffering will be used. That is, the plot will be drawn first into a buffer. When drawing of the plot is finished the buffer is drawn onto the screen. Double buffering is needed for smooth animations. For off-screen creation it has to be switched off.
`horizontalAnchor` = `center`	*string*	no	Horizontal position of the paper relative to the view rectangle. Possible values are `left`, `center`, and `right`
`verticalAnchor` = `center`	*string*	no	Vertical position of the paper relative to the view rectangle. Possible values are `top`, `center`, and `bottom`.
`paper` = `0 0 1 0.6`	*double[]*	no	Rectangle defining the paper. The first two values determine the x- and y- coordinates (in device-independent units) of the lower-left corner. The last two values determine the upper-right corner.
`plot`	*Set*	no	⇒Plot.
`antiAliasing` = `true`	*boolean*	no	If `true` anti-aliasing rendering will be performed. Works only for `jcckit.Graphics2DPlotCanvas`.

2.14 Hint

Class type: `jcckit.plot.Hint`

Concrete classes: ⇒Position Hint, ⇒Shape Attributes Hint

Synopsis: Mechanism for providing additional data to draw a curve point in a ⇒Plot. For example, drawing bar charts needs ⇒Position Hints. For more details see the chapter 1.3 of the JCCKit User Guide (http://jcckit.sourceforge.net/UserGuide/curves.html).

2.15 Legend

Parameter type: *Set*

Synopsis: Legend of a ⇒Plot.

Note, that all lengths, distances, and points are specified in device independent units.

Name & Default Value	Type	Mandatory?	Description
`upperRightCorner = 0.94 0.54`	*double[]*	no	Position of the upper-right corner of the legend box.
`boxWidth = 0.2`	*double*	no	Width of the legend box.
`boxHeight = 0.1`	*double*	no	Height of the legend box.
`boxAttributes` = *white background and black border*	*Object*	no	⇒Shape Attributes of the legend box.
`title = Legend`	*string*	no	Title.
`titleDistance = 0.005`	*double*	no	Distance of the title from the top of the legend box.
`titleAttributes` = *default text attributes, anchor: top, center*	*Object*	no	⇒Basic Graphic Attributes of the legend title.
`leftDistance = 0.01`	*double*	no	Horizontal distance between the line part of the legend symbol and the left border of the legend box.
`bottomDistance = 0.02`	*double*	no	Distance between the last row and the bottom of the legend box.
`topDistance = 0.04`	*double*	no	Distance between the first row and the top of the legend box.
`lineLength = 0.035`	*double*	no	Length of the line part of the legend symbol.
`symbolSize = 0.01`	*double*	no	Size of the symbol part of the legend symbol.

Name & Default Value	Type	Mandatory?	Description
`curveTitleDistance = 0.005`	*double*	no	Horizontal distance between the line part of the legend symbol and the curve title.
`curveTitleAttributes` = *default text attributes*	*Object*	no	⇒Basic Graphic Attributes of curve titles printed in the legend.

2.16 Plot

Parameter type: *Set*

Synopsis: A plot is specified by its coordinate system, axes, axis labeling, curve attributes, etc.

Name & Default Value	Type	Mandatory?	Description
`coordinateSystem = ⇒Cartesian Coordinate System`	*Object*	no	⇒Coordinate System.
`curveFactory = ⇒Simple Curve Factory`	*Object*	no	⇒Curve Factory.
`initialHintForNextCurve` = *no initial hint*	*Object*	no	⇒Hint.
`legend` = *default values*	*Set*	no	⇒Legend.
`legendVisible = true`	*boolean*	no	If `true` a legend will be drawn on top of the plot.

2.17 Plot Applet

Parameter type: *Set*

Synopsis: An AWT ⇒Graphics Plot Canvas embedded in an Applet. The applet parameters are taken as the configuration parameters.

Name & Default Value	Type	Mandatory?	Description
`background` = *depends on the environment*	*color*	no	Background color.

Name & Default Value	Type	Mandatory?	Description
`foreground` = *depends on the environment*	*color*	no	Foreground color.
`doubleBuffering` = `true`	*boolean*	no	If `true` double buffering will be used. That is, the plot will be drawn first into a buffer. When drawing of the plot is finished the buffer is drawn onto the screen. Double buffering is needed for smooth animations. For off-screen creation it has to be switched off.
`horizontalAnchor` = `center`	*string*	no	Horizontal position of the paper relative to the view rectangle. Possible values are `left`, `center`, and `right`
`verticalAnchor` = `center`	*string*	no	Vertical position of the paper relative to the view rectangle. Possible values are `top`, `center`, and `bottom`.
`paper` = `0 0 1 0.6`	*double[]*	no	Rectangle defining the paper. The first two values determine the x- and y- coordinates (in device-independent units) of the lower-left corner. The last two values determine the upper-right corner.
`plot`	*Set*	no	⇒Plot.
`waitingMessage` = `Please wait, applet data are loading...`	*string*	no	Message presented after the applet has been loaded but before construction of the plot canvas is finished.
`data`	*Set*	yes/no	⇒Data Plot. This is a mandatory parameter if `dataProperties` is absent.
`dataProperties`	*string*	yes/no	File name relative to the applet's document base. It should denote a `.properties` file with the configuration parameters for ⇒Data Plot. This is a mandatory parameter if `data` is absent
`renderer` = `jcckit.renderer.GraphicsRenderer`	*string*	no	Fully qualified class name of a class extending the default renderer.

2.18 Position Hint

Parameter type: *Object*

Class parameter: `className = jcckit.plot.PositionHint`

Synopsis: Hint needed for bar charts or error bars in a ⇒Plot. For the correct usage it is recommended to study the examples on the JCCKit home page http://jcckit.sourceforge.net.

Name & Default Value	Type	Mandatory?	Description
`position`	*double[]*	no	A position in device-independent units.
`origin = 0 0` *or* `position` *if defined*	*double[]*	no	Position of the origin of the data coordinate system in device-independent units.

2.19 Shape Attributes

Parameter type: *Object*

Class parameter: `className = jcckit.graphic.ShapeAttributes`

Synopsis: Basic attributes of elementary graphic objects in a ⇒Plot.

Name & Default Value	Type	Mandatory?	Description
`fillColor` = *no filling*	*color*	no	The color a shape will be filled up.
`lineColor` = *foreground color*	*color*	no	The color of the line of a shape.
`lineThickness = 0`	*double*	no	The line thickness in device-independent units. This parameter will be ignored in a purely AWT context where `jcckit.renderer.GraphicsRenderer` is used.
`linePattern` = *solid line*	*double[]*	no	The line pattern. It is a sequence of lengths (in device-independent units) where the pen is up or down. The first element is the length where the pen is down. The next element is the length where the pen is up and so on. The pattern is cyclically repeated. This parameter will be ignored in a purely AWT context where `jcckit.renderer.GraphicsRenderer` is used.

2.20 Shape Attributes Hint

Parameter type: *Object*

Class parameter: `className = jcckit.plot.ShapeAttributesHint`

Synopsis: This hint allows to have gradually changed properties of curve symbols and curve line segments. For the correct usage it is recommended to study the Lorenz example on the JCCKit home page http://jcckit.sourceforge.net.

Name & Default Value	Type	Mandatory?	Description
`initialAttributes` = *default values of Shape Attributes*	*Set*	no	Initial values of shape attributes. Note, that default fill and line colors are undefined. In this case color increments have no effects.
`fillColorHSBIncrement = 0 0 0`	*double[]*	no	Increments of hue, saturation, and brightness of the symbol fill color.
`lineColorHSBIncrement = 0 0 0`	*double[]*	no	Increments of hue, saturation, and brightness of the line color.
`lineThicknessIncrement = 0`	*double*	no	Increment of the line thickness. This parameter will be ignored in a purely AWT context where `jcckit.renderer.GraphicsRenderer` is used.

2.21 Simple Curve Factory

Parameter type: *Object*

Class parameter: `className = jcckit.plot.SimpleCurveFactory`[3]

Synopsis: Definition of the properties of a curve of a ⇒Plot created by this factory.

[3] Parameter `className` could be omitted if used in conjunction with ⇒Plot

Name & Default Value	Type	Mandatory?	Description
`definitions`	*Set[]*	no	*Sequence parameter* of curve definitions. That is, for each name in the sequence a ⇒Curve element in this set has to appear. The elements will be cyclically reused if more curves are needed than this sequence has elements.

2.22 Square

Parameter type: *Object*

Class parameter: `className = jcckit.plot.SquareSymbolFactory`

Synopsis: ⇒Symbol Factory for square symbols.

Name & Default Value	Type	Mandatory?	Description
`size = 0.01`	*double*	no	Size of the symbol in device-independent units.
`attributes`	*Object*	no	⇒Graphic Attributes of the symbol.

2.23 Symbol Factory

Class type: `jcckit.plot.SymbolFactory`

Concrete classes: ⇒Square, ⇒Circle, ⇒Bar, ⇒Error Bar

Synopsis: Definition of factories creating symbols for ⇒Curve points in a ⇒Plot.

2.24 Tic Label Map

Parameter type: *Object*

Class parameter: `className = jcckit.plot.TicLabelMap`

Synopsis: Mapping of numerical tic labels onto arbitrary text labels.

Name & Default Value	Type	Mandatory?	Description

`map`	*string*	yes	Map description. It is a list of conditions separated by ';'. The conditions are tested from left to right until a condition is fulfilled. Its value will be the tic label. If no condition is fullfilled a '?' will be used. A condition description has one of the following forms:
			• *label*
			This condition is always fulfilled. It will return *label*. This is a kind of else condition which should be put at the end of the condition list.
			• *number* `=` *label*
			This conditions maps a particular number onto a label. In order to be equal with the specified number the tic value should not deviate more than 1 ppm (part per millions) from *number*.
			• a`:`b `=` *label*
			This condition maps an interval onto a label. The condition reads $a \leq$ tic label value $< b$
			Examples:
			`1=m;2=t;3=w;4=t;5=f;6=s;7=s`
			`0.5:1.5 = I; 1.5:2.5=II; the rest`

Index

