JavaStar API Reference

Revision 1.1.4 October 1997

Contents

API Overview 1

This book describes the classes and methods of the JavaStar public library. If you are familiar with Java programming, you can use these classes to extend the power of your scripts.

This chapter presents the groups that the categories are collected into, and the meaning of deprecated methods.

Topics:

Figure 1-1 shows the hierarchy of these classes in the library.

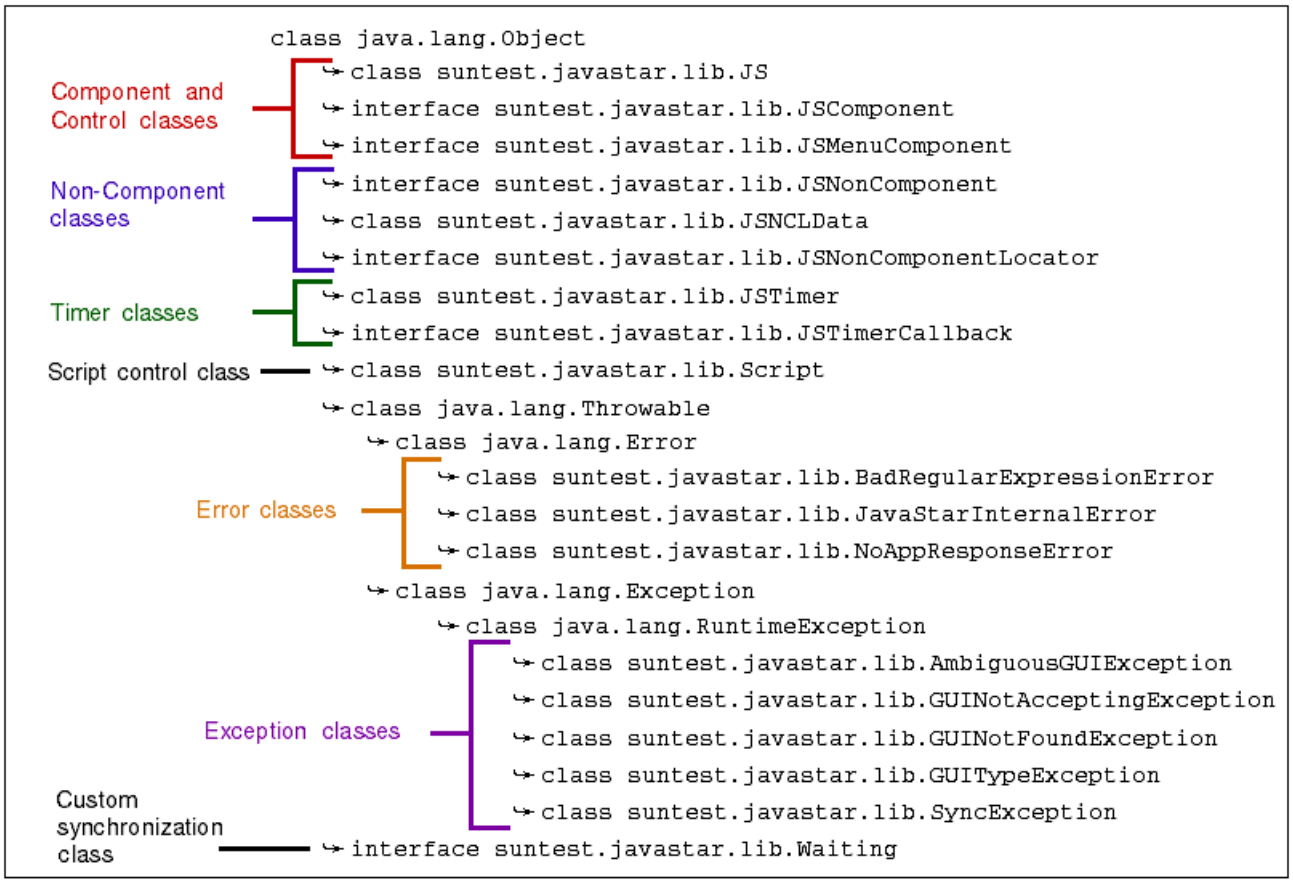

Figure 1-1 Hierarchy of the JavaStar class library

Class Categories

The JavaStar public library provides:

- Component and Control Classes
- Non-Component Classes
- Timer Classes
- Script Control Class
- Error Classes
- Exception Classes
- Waiting—Custom Synchronization

Component and Control Classes

The `JS` class, `JSComponent` interface, and `JSMenuComponent` interface comprise the component and control classes. The `JS` class contains component and script control classes—you'll probably use this class often if you customize scripts. The `JSComponent` and `JSMenuComponent` interfaces are implemented privately. Many `JS` methods return objects of these types.

Non-Component Classes

The `JSNonComponent` interface, `JSNonComponentLocator` interface, and `JSNCLData` class are described in Non-Component Classes. You use these when you are testing an application that uses components that do not extend `java.awt.Component`.

Timer Classes

The Timer classes (`JSTimer` and `JSTimerCallback`) allow you to create your own timers and register callbacks.

Script Control Class

The `Script` class defines the structure of JavaStar scripts. This information is included so you can better understand the way scripts work.

Error Classes

These classes define JavaStar-specific errors: `BadRegularExpressionError`, `JavaStarInternalError`, `NoAppResponseError`.

Exception Classes

These are the JavaStar-specific exceptions that the API can throw. Even if you're not using the API to customize scripts, this chapter can help you understand the exceptions thrown by scripts you generated using the record feature.

Waiting—Custom Synchronization

The `Waiting` interface is where you define custom synchronization conditions. You call objects of this class using `JS.waitFor()`.

Deprecated Methods

Several classes of the JavaStar API contain methods that have been deprecated with this release. A *deprecated* method is obsolete, but is currently included to support existing scripts.

If you compile a script that uses a deprecated method, the compiler issues a warning. You're not prevented from using the methods, but since they will be removed in a later release, you should avoid creating new scripts that use them, and consider updating older scripts, as well.

Each class description in this documentation groups deprecated methods together. The method description suggests the class you should use instead, where a replacement is offered.

Component and Control Classes 2

The JavaStar public library includes a concrete class and two interfaces for manipulating and comparing components, as well as adding script control.

- JS
- JSComponent
- JSMenuComponent

With the exception of some methods of the `JS` class, the members of these three classes can only be used with Components and MenuComponents—objects that derive from `java.awt.Component`. For manipulation of non-Components—objects from toolkits that do not derive from `java.awt.Component`—refer to JSNonComponent in the chapter "Non-Component Classes."

JS

The `suntest.javastar.lib.JS` class contains static functions for playback scripts to use. The constructor for this class is private.

This section describes `JS` by:

- Inheritance
- Syntax
- Variables
- Methods

Inheritance

```
java.lang.object
➥ suntest.javastar.lib.JS
```

Syntax

```
public class JS
extends Object
```

Variables

Variables	Description
public static PrintStream log	This PrintStream will be attached to the on-screen terminal and the error-logging file when a script is playing.

Methods

This section provides two ways to view methods. You can use:

- Methods by Name
- Methods by Category

If you are viewing this documentation on-line, you can click on any method name to jump to the full description, syntax, and parameter specification.

Methods by Name

Table 2-1 JS methods by name

Method	Category	Description
applet (String, int)	Find methods	Finds an applet matching the type and position you specify.
check(boolean)	Comparison methods	Checks the boolean you pass and reports `true` or `false` based on the comparison.
check(boolean, String)	Comparison methods	Checks the boolean variable you pass and reports `true` or `false` based on the comparison. Includes the *String* when writing to the log.
check(Script, boolean)	Comparison methods	Checks the boolean and reports `true` or `false` based on the comparison.
check(Script, boolean, String)	Comparison methods	Checks the boolean passed and reports `true` or `false` based on the comparison. Includes the *String* when writing to the log.
delay(long)	Script control / convenience methods	Requests a delay in milliseconds.

Table 2-1 JS methods by name

Method	Category	Description
deliverEventToHidden(boolean)	Script control / convenience methods	Allows events to be sent to hidden or disabled Components.
dialog(String, String)	Find methods	Finds all dialog windows exactly matching the type and title you specify.
dialogRE(String, String)	Find methods	Finds all dialog windows matching the type and title you specify. The title supports regular expressions.
dialogRX(String, String)	Deprecated methods	Finds all dialog windows matching the type and title you specify.
find(String, String)	Find methods	Finds all frames exactly matching the type and title you specify.
findRE(String, String)	Find methods	Finds all frames exactly matching the type and title you specify. The title supports regular expressions.
findRX(String, String)	Deprecated methods	Finds all frames exactly matching the type and title you specify. The title supports regular expressions.
flushEventQueue()	Comparison methods	Flushes any pending playback events and flushes the system event queue.
frame(String)	Find methods	Finds all frames exactly matching the title you provide.
frameRE(String)	Find methods	Finds all frames with a title matching the regular expression you specify.
frameRX(String)	Deprecated methods	Finds all frames with a title matching the regular expression you specify.
getProperty(String)	Get / Set methods	Reads the JavaStar test property of name and returns the value for the key.
getTimeout()	Get / Set methods	Returns the current timeout value (in seconds).
getTypingRate()	Get / Set methods	Returns the current typing rate (milliseconds per character).
goldenDirectory(String, String)	Deprecated methods	Returns the name of the gold file directory for the specified script.
lookup(String)	Find methods	Finds all named components matching the name you provide.

Table 2-1 JS methods by name

Method	Category	Description
mlookup(String)	Find methods	Finds all `MenuComponents` matching the name you provide.
note(String)	Script control / convenience methods	Enters a note into the log.
pause()	Script control / convenience methods	Inserts a breakpoint into your script.
playbackEnd(String, boolean)	Deprecated methods	Terminates script playback.
playbackEnd(String, Throwable)	Deprecated methods	Terminates playback of a script and throws an exception.
playbackInit(String)	Deprecated methods	Performs initialization to allow playback.
postEvent(AWTEvent)	Script control / convenience methods	Posts an event to the system EventQueue.
processPlayerArgs(String)	Deprecated methods	Adjusts the arguments to a playback script.
setProperty(String, String)	Get / Set methods	Sets the JavaStar test property.
setTimeout(int)	Get / Set methods	Increases the timeout value in seconds.
setTypingRate(int)	Get / Set methods	Sets the typing rate.
startApplication(Script)	Deprecated methods	Starts the application under test.
verifyAnyField(Script, boolean, boolean, Object String, Object, String)	Comparison methods	Reads a data member on the contained Component and compares this against the expected value.
verifyAnyMethod(Script, boolean, boolean, Object, String, Object, String)	Comparison methods	Calls the method name() of thisObj and compares against the expected value.
waitFor(Waiting)	Comparison methods	Executes a custom synchronization. Used in combination with the user-implemented `Waiting` interface.
wrap(Component)	Script control / convenience methods	Wraps the given Component in a `JSComponent`.

Methods by Category

Table 2-2 JS methods by category

Category	Method	Description
Comparison methods These methods compare the state of components. Comparison methods include verify and synchronization methods, a custom synchronization method, and simple check methods.	check(boolean)	Checks the boolean you pass and reports `true` or `false` based on the comparison.
	check(boolean, String)	Checks the boolean variable you pass and reports `true` or `false` based on the comparison. Includes the *String* when writing to the log.
	check(Script, boolean)	Checks the boolean and reports `true` or `false` based on the comparison.
	check(Script, boolean, String)	Checks the boolean passed and reports `true` or `false` based on the comparison. Includes the *String* when writing to the log.
	verifyAnyField(Script, boolean, boolean, Object String, Object, String)	Reads a data member on the contained Component and compares this against the expected value.
	verifyAnyMethod(Script, boolean, boolean, Object, String, Object, String)	Calls the method name() of thisObj and compares against the expected value.
	waitFor(Waiting)	Executes a custom synchronization. Used in combination with the user-implemented `Waiting` interface.

Table 2-2 JS methods by category

Category	Method	Description
Deprecated methods These methods will be phased out of JavaStar in the future, but are included in this release to support scripts written with earlier versions of the JavaStar library. See Deprecated Methods in the API Overview for more details.	dialogRX(String, String)	Finds all dialog windows matching the type and title you specify.
	findRX(String, String)	Finds all frames exactly matching the type and title you specify. The title supports regular expressions.
	frameRX(String)	Finds all frames with a title matching the regular expression you specify.
	goldenDirectory(String, String)	Returns the name of the gold file directory for the specified script.
	playbackEnd(String, boolean)	Terminates script playback.
	playbackEnd(String, Throwable)	Terminates playback of a script and throws an exception.
	playbackInit(String)	Performs initialization to allow playback.
	processPlayerArgs(String)	Adjusts the arguments to a playback script.
	startApplication(Script)	Starts the application under test.

Table 2-2 JS methods by category

Category	Method	Description
Find methods These methods locate and return `JSComponent` or `JSMenuComponent`. according to your specifications.	applet (String, int)	Finds an applet matching the type and position you specify.
	dialog(String, String)	Finds all dialog windows exactly matching the type and title you specify.
	dialogRE(String, String)	Finds all dialog windows matching the type and title you specify. The title supports regular expressions.
	find(String, String)	Finds all frames exactly matching the type and title you specify.
	findRE(String, String)	Finds all frames exactly matching the type and title you specify. The title supports regular expressions.
	frame(String)	Finds all frames exactly matching the title you provide.
	frameRE(String)	Finds all frames with a title matching the regular expression you specify.
	lookup(String)	Finds all named components matching the name you provide.
	mlookup(String)	Finds all `MenuComponents` matching the name you provide.
Get/Set methods These methods provide a way to get and set the values of certain data members.	getProperty(String)	Reads the JavaStar test property of name and returns the value for the key.
	getTimeout()	Returns the current timeout value (in seconds).
	getTypingRate()	Returns the current typing rate (milliseconds per character).
	setProperty(String, String)	Sets the JavaStar test property.
	setTimeout(int)	Increases the timeout value in seconds.
	setTypingRate(int)	Sets the typing rate.

Table 2-2 JS methods by category

Category	Method	Description
Script control/ convenience methods These methods provide ways you can control your script—for example, inserting breakpoints or boolean checks.	delay(long)	Requests a delay in milliseconds.
	deliverEventToHidden(boolean)	Allows events to be sent to hidden or disabled Components.
	flushEventQueue()	Flushes any pending playback events and flushes the system event queue.
	note(String)	Enters a note into the log.
	pause()	Inserts a breakpoint into your script.
	postEvent(AWTEvent)	Posts an event to the system EventQueue.
	wrap(Component)	Wraps the given Component in a `JSComponent`.

delay(long)

Syntax

```
public static void delay(long ms)
```

Category

Script control and convenience methods

Description

Request a delay of *ms* milliseconds. This value may be ignored or scaled at runtime.

Parameters

- *ms* – the number of milliseconds to pause

deliverEventToHidden(boolean)

Syntax

```
public static void deliverEventToHidden(boolean deliver)
```

Category

Script control and convenience methods

Description

If called with `true`, this method allows events to be sent to hidden or disabled Components for the duration of the script or until called with `false`.

Parameters

- *deliver* - if `true`, allow events to hidden; if `false`, do not.

flushEventQueue()

Syntax

```
public static void flushEventQueue()
```

Category

Script control and convenience methods

Description

Flushes any pending playback events and flushes the system event queue.

note(String)

Syntax

```
public static void note(String c)
```

Category

Script control and convenience methods

Description

Enters the String *c* in the log. Use to provide additional debugging information.

Parameters

- *c* – the note to enter in the log

pause()

Syntax

```
public static void pause()
```

Category

Script control and convenience methods

Description

Forces the script to pause during playback. Use this to insert a breakpoint into your test.

postEvent(AWTEvent)

Syntax

```
public static void postEvent(AWTEvent evt)
```

Category

Script control and convenience methods

Description

Posts an event to the system EventQueue. `postEvent()` is shorthand for `Toolkit.getDefaultToolkit().getSystemEventQueue().postEvent(evt);`.

Parameter

- *evt* - the event to be posted.

wrap(Component)

Syntax

```
public static JSComponent wrap(Component comp)
```

Category

Script control and convenience methods

Description

Wraps the given Component in a `JSComponent`. Returns a `JSComponent` containing only *comp* (`getUnique()==comp`).

Parameters

- *comp* – a Component to be wrapped

applet (String, int)

Syntax

```
public static JSComponent applet(String type, int position)
```

Category

Find methods

Description

Finds an Applet matching type exactly; normally position is 0; If there is more than one applet of the same type, the position will disambiguate them.

Parameters

- *type* – the full class name of the Applet
- *position* – the construction-order ordinal for disambiguation.

dialog(String, String)

Syntax

```
public static JSComponent dialog(String type, String title)
```

Category

Find methods

Description

Finds all dialogs matching the type and title exactly. Returns a `JSComponent` containing all matches.

Parameters

- *type* – the full class name of the Dialog
- *title* – the exact title of the Dialog

dialogRE(String, String)

Syntax

```
public static JSComponent dialogRE(String type, String expression)
```

Category

Find methods

Description

Finds all dialogs matching type exactly and title matching regular expression. Returns a `JSComponent` containing all matches. For syntax, see Current Syntax in the chapter "Syntax for Regular Expressions."

Parameters

- *type* - the full class name of the Dialog
- *expression* - regular expression for title of the Dialog

find(String, String)

Syntax

```
public static JSComponent find(String type, String title)
```

Category

Find methods

Description

Finds all frames matching type and title exactly. Returns a `JSComponent` containing all matches.

Parameters

- *type* – the full class name of the Frame
- *title* – the exact title of the Frame

findRE(String, String)

```
public static JSComponent findRE(String type, String expression)
```

Category

Find methods

Description

Finds all frames matching *type* exactly and *title* matching regular expression. Returns a `JSComponent` containing all matches. For syntax, see Current Syntax in the chapter "Syntax for Regular Expressions."

Parameters

- *type* - the full class name of the Frame
- *expression* - regular expression for title of the Frame

frame(String)

Syntax

```
public static JSComponent frame(String title)
```

Category

Find methods

Description

Finds all frames matching *title* exactly. Returns a `JSComponent` containing all matches.

Parameters

- *title* – the exact title of the Frame

frameRE(String)

Syntax

```
public static JSComponent frameRE(String expression)
```

Category

Find methods

Description

Finds all frames with title matching regular expression. Returns a `JSComponent` containing all matches. For syntax, see Current Syntax in the chapter "Syntax for Regular Expressions."

Parameters

- *expression* - regular expression for title of the Frame

lookup(String)

Syntax

```
public static JSComponent lookup(String name)
```

Category

Find methods

Description

Finds all Named components with a matching name (return of the `getName()` method). Returns a `JSComponent` containing all matches.

Use `'.'` as a scope operator. `JS.lookup("A.B")` is equivalent to `JS.lookup("A").lookup("B")`. `'#'` must be used to escape `'.'`

Parameters

- *name* – the name to search for

mlookup(String)

Syntax

```
public static JSMenuComponent mlookup(String name)
```

Category

Find methods

Description

Finds all MenuComponents with a matching *name* (return of the `getName()` method). Returns a `JSMenuComponent` containing all matches.

Use `'.'` as a scope operator. `JS.mlookup("A.B")` is equivalent to `JS.mlookup("A").mlookup("B")`. Returns a `JSMenuComponent` containing all matches. `'#'` must be used to escape `'.'`

Parameters

- *name* – the name to search for

getProperty(String)

Syntax

```
public static String getProperty(String name)
```

Category

Get / Set methods

Description

Reads the JavaStar test property of *name* and returns the value for the key. If the property is not defined, this returns null.

Parameters

- *name* – the key to search for

setProperty(String, String)

Syntax

```
public static void setProperty(String name, String value)
```

Category

Get / Set methods

Description

Sets the JavaStar test property of *name* to *value.*

Parameters

- *name* – the key
- *value* – the value to associate with the key; null will un-set

getTimeout()

Syntax

```
public static int getTimeout()
```

Category

Get / Set methods

Description

Returns the current timeout value (in seconds).

setTimeout(int)

Syntax

```
public static void setTimeout(int secs)
```

Category

Get / Set methods

Description

Increases the timeout value to *secs*. If the setting requested is less than the current timeout value, this has no effect.

Parameters

- *secs* – requested value (in seconds)

getTypingRate()

Syntax

```
public static int getTypingRate()
```

Category

Get / Set methods

Description

Returns the current typing rate (milliseconds per character).

setTypingRate(int)

Syntax

```
public static void setTypingRate(int ms)
```

Category

Get / Set methods

Description

Sets the current typing rate to *ms*.

Parameters

- *ms* – requested typing rate (in milliseconds per character)

check(boolean)

Syntax

```
public static boolean check(boolean b)
```

Category

Comparison methods.

Description

Checks the boolean variable you pass and reports `true` or `false` to the log based on the comparison. Includes the *String* when writing to the log. Returns the value of *b*. Use this to insert non-GUI checks into your test.

Parameters

- *b* – the value required to be true

check(boolean, String)

Syntax

```
public static boolean check(boolean b, String s)
```

Category

Comparison methods

Description

Checks the boolean and reports `true` or `false` to the log (and includes the String *s*). Returns the value of *b*. Use this to insert non-GUI checks into your test.

Parameters

- *b* – the value required to be true
- *s* – a String to report

check(Script, boolean)

Syntax

```
public static boolean check(Script sc, boolean b)
```

Category

Comparison methods

Description

Checks the boolean and reports `true` or `false` to the log based on the comparison. Returns the value of *b*. Use this to insert non-GUI checks into your test.

Parameters

- *sc* – the script where you are performing the comparison
- *b* – the value required to be true

check(Script, boolean, String)

Syntax

```
public static boolean check(Script sc, boolean b, String s)
```

Category

Comparison methods

Description

Checks the boolean and reports true or false to the log (including the String *s*) based on the comparison results. Returns the value of *b*. Use this to insert non-GUI checks into your test.

Parameters

- *sc* – the script where you are performing the comparison
- *b* – the value required to be true
- *s* – a String to report

verifyAnyField(Script, boolean, boolean, Object String, Object, String)

Syntax

```
public static boolean verifyAnyField(Script sc, boolean sync,
boolean retry, Object thisObj, String name, Object expected, String
purpose)
```

Category

Comparison methods

Description

Reads the data member *name* on the contained Component and compares this against the expected value. Returns `true` or `false` depending on the comparison results.

Parameters

- *sc* – the script where you are performing the comparison
- *sync* – if set to true, this throws a SyncException on failure
- *retry* – if set to true, this method continues checking until timeout
- *thisObj* – the Object containing the data member to check
- *name* – the name of the non-static data member to verify
- *expected* – the expected value of the method
- *purpose* – the purpose of the verify operation

verifyAnyMethod(Script, boolean, boolean, Object, String, Object, String)

Syntax

```
public static boolean verifyAnyMethod(Script sc, boolean sync,
boolean retry, Object thisObj, String name, Object expected, String
purpose)
```

Category

Comparison methods

Description

Calls the method *name()* of *thisObj* and compares against the expected value. Returns `true` or `false` depending on the comparison results.

Parameters

- *sc* – the script where you are performing the comparison
- *sync* – if set to true, this throws an SyncException on failure
- *retry* – if set to true, this method continues checking until timeout
- *thisObj* – the Object containing the method to check
- *name* – the name of the method to verify (must not take parameters)
- *expected* – the expected value of the method
- *purpose* – the purpose of the verify operation

waitFor(Waiting)

Syntax

```
public static void waitFor(Waiting wait)
```

Category

Comparison methods

Description

Use this method to perform custom synchronizations—`waitFor()` waits for the `Waiting` object's `getState()` function to return `true`, or for the timeout to expire. You must implement the `Waiting` class to define the wait condition. If the timeout expires, this method throws a `GUINotAcceptingException`.

Parameters

- *wait* – the object whose state must be `true`

dialogRX(String, String)

Syntax

```
public static JSComponent dialogRX(String type, String expression)
```

Category

Deprecated methods

Description

Replaced by dialogRE(String, String).

Finds all dialogs matching type exactly and with the title matching regular expression (see the chapter "Syntax for Regular Expressions."). Returns a `JSComponent` containing all matches.

Parameters

- *type* – the full class name of the Dialog
- *expression* – regular expression for title of the Dialog

findRX(String, String)

Syntax

```
public static JSComponent findRX(String type, String expression)
```

Category

Deprecated methods

Description

Replaced by findRE(String, String).

Finds all frames matching *type* exactly and *title* matching regular expression (see the chapter "Syntax for Regular Expressions."). Returns a `JSComponent` containing all matches.

Parameters

- *type* – the full class name of the Frame
- *expression* – regular expression for title of the Frame

frameRX(String)

Syntax

```
public static JSComponent frameRX(String expression)
```

Category

Deprecated methods

Description

Replaced by frameRE(String).

Finds all frames with *title* matching regular expression (see the chapter "Syntax for Regular Expressions."). Returns a `JSComponent` containing all matches.

Parameters

- *expression* – regular expression for title of the Frame

goldenDirectory(String, String)

Syntax

```
public static String goldDirectory(String name, String
defaultValue)
```

Category

Deprecated methods

Description

This method is no longer useful.

Returns the name of the gold file directory for the specified *script*. If there is no setting for the directory, this method sets playback to use *defaultValue*.

Parameters

- *name* – the name of the playback script
- *defaultValue* – the value to use if there is no preference

playbackEnd(String, boolean)

Syntax

```
public static void playbackEnd(String name, boolean norm)
```

Category

Deprecated methods

Description

This method is no longer useful.

Terminates a playback script. If `processPlayerArgs()` included "`-exit`", this calls `System.exit()` and does not return. This is used by a standalone Script's main method.

Parameters

- *name* – name of script
- *norm* – normal or abnormal termination

playbackEnd(String, Throwable)

Syntax

```
public static void playbackEnd(String name, Throwable exc)
```

Category

Deprecated methods

Description

This method is no longer useful.

Terminates a playback script. If `processPlayerArgs()` included "`-exit`", this calls `System.exit()` and does not return. This is used by a standalone Script's main method.

Parameters

- *name* – name of script
- *exc* – exception which caused abnormal termination

playbackInit(String)

Syntax

```
public static void playbackInit(String name) throws IOException
```

Category

Deprecated methods

Description

This method is no longer useful.

Performs initialization to allow playback. Sets up the Record / Playback GUI and the log stream. This is used by the standalone Script's main method.

Parameters

- *name* – the name of the playback script

processPlayerArgs(String)

Syntax

```
public static String[] processPlayerArgs(String args[])
```

Category

Deprecated methods

Description

This method is no longer useful.

Adjusts the arguments to a playback script, accepting arguments of the form:

```
[-exit] [-scale double] [-timeout integer] [-log file] [-gold
directory] [-v] [-out]
```

This is used by the standalone Script's main method. Returns the arguments after stripping the leading arguments that JavaStar uses.

Parameters

- *args[]* – the arguments to the test program

startApplication(Script)

Syntax

```
public static void startApplication(Script sc)
```

Category

Deprecated methods

Description

This method is no longer useful.

Starts the application under test with the appropriate reporting. This will call the run method of the script.

Parameters

- *sc* – the script to run against the application

JSComponent

With the `suntest.javastar.lib.JSComponent` class, you can send events to components, or find related subcomponents—such as those it contains, associated dialogs, and so on. While you can't instantiate a `JSComponent`, many of the methods of the `JS` class return `JSComponents`. Also, if you have a component you want to wrap into a `JSComponent`, you can use wrap(Component).

This section describes `JSComponent` by:

- Syntax
- Methods

Syntax

```
public interface JSComponent
extends Object
```

Methods

This section provides two ways to view methods. You can use:

- Methods by Name
- Methods by Category

If you are viewing this documentation on-line, you can click on any method name to jump to the full description, syntax, and parameter specification.

Methods by Name

Table 2-3 `JSComponent` methods by name

Method	Category	Description
absolute(int)	Scrolling methods	Scrolls to the position you specify.
action()	Send event methods	Sends an `ACTION` event to a Button or Textfield.
action(String)	Send event methods	Sends an action event to Button, Choice, List, or TextField.
ageDifferentiation(int, int)	Find methods	Disambiguates based on relative construction time (approximately).
button(String)	Find methods	Finds contained buttons which have an exact label match.
buttonPress()	Send event methods	Sends `MOUSE_PRESSED`, `MOUSE_RELEASED`, `MOUSE_CLICKED`, and the ActionEvent. Valid only for Buttons.
buttonRE(String)	Find methods	Finds contained buttons which have labels matching this regular expression.
buttonRX(String)	Deprecated methods	Finds contained buttons which have labels matching this regular expression. Uses old syntax for regular expressions.
center(String, String)	Find methods	Returns the Center member that meets the specified type and label requirements.
child(int, String, String)	Find methods	Returns a child component that meets the specified type and label requirements.
choosefile(String, String)	Send event methods	FileDialog simulation event.
deiconify()	Send event methods	Sends a `WINDOW_DEICONIFY` event.
deselect()	Send event methods	Sends a `DESELECTED` ItemEvent to a Checkbox.
deselect(int)	Send event methods	Deselects the indexed entry of a List.
deselect(String)	Send event methods	Deselects an item in a List.
deselect(int, String)	Send event methods	Deselects the indexed entry of a List. Requires that the text of the item match.

Table 2-3 `JSComponent` methods by name

Method	Category	Description
destroy()	Send event methods	Sends a `WINDOW_CLOSING` event.
dialog(String)	Find methods	Returns dialogs of this Frame whose title matches title.
dialog(String, String)	Find methods	Returns dialogs of this Frame that meet specified type and title requirements.
dialogRE(String)	Find methods	Returns dialogs that have title matching regular expression.
dialogRE(String, String)	Find methods	Returns dialogs of this Frame of the specified type that have title matching regular expression.
dialogRX(String)	Deprecated methods	Returns dialogs that have a title matching this regular expression.
dialogRX(String, String)	Deprecated methods	Returns dialogs that meet specified type and title matching regular expression.
east(String, String)	Find methods	Returns the East member, requiring that it meet specified type and label requirements.
fkey(int, int)	Send event methods	Sends a `KEY_ACTION` KeyEvent.
fkeyUp(int, int)	Send event methods	Sends a `KEY_ACTION_RELEASE` KeyEvent.
getAll()	Query methods	Returns all Components pointed to by this `JSComponent`.
getAllValid()	Query methods	Returns all Components pointed to by this `JSComponent` which are capable of receiving events.
getAnyField(String)	Convenience methods	Returns the value of the data member name on the contained Component.
getAnyMethod(String)	Convenience methods	Returns the value of the method `name()` on the contained Component.
getUnique()	Query methods	Returns the Component pointed to by this `JSComponent`.
getValidUnique()	Query methods	Returns the only Component pointed to by this `JSComponent` which is capable of receiving events.

Table 2-3 `JSComponent` methods by name

Method	Category	Description
gotFocus()	Send event methods	Sends a `FOCUS_GAINED` FocusEvent.
hasMember(String)	Find methods	Returns any components of `JSComponent` that match the type you specify.
hasMember(String, String)	Find methods	Returns a `JSComponent` that contains the subset of this `JSComponent`'s Components which contain at least one member matching the specification.
hasMemberRE(String, String)	Find methods	Same as `hasMember()`, except this method accepts regular expressions for the Component title.
hasMemberRX(String, String)	Deprecated methods	Equivalent to `hasMemberRE()`, except that this method uses the old regular expression syntax.
iconify()	Send event methods	Sends a `WINDOW_ICONIFY` event.
isUnique()	Query methods	Returns true if there is exactly one Component contained by this `JSComponent`.
isValidUnique()	Query methods	Returns true if there is one valid match that can receive events.
key(int, int)	Send event methods	Sends a `KEY_PRESSED` KeyEvent.
key(int, int, int, int)	Send event methods	Typing simulation event.
keyPressed(int, char, int)	Send event methods	Sends a `KEY_PRESSED` KeyEvent.
keyReleased(int, char, int)	Send event methods	Sends a `KEY_RELEASED` KeyEvent.
keyTyped(char)	Send event methods	Sends a `KEY_TYPED` KeyEvent.
keyTyped(int, int)	Deprecated methods	Sends a `KEY_TYPED` KeyEvent. Replaced by `keyTyped(char)`.
keyUp(int, int)	Send event methods	Sends a `KEY_RELEASED` KeyEvent.
lineDown(int)	Scrolling methods	Scrolls down one line.
lineUp(int)	Scrolling methods	Scrolls up one line.

Table 2-3 `JSComponent` methods by name

Method	Category	Description
lookup(String)	Find methods	Finds any components with a matching name contained in the components.
lostFocus()	Send event methods	Sends a `FOCUS_LOST` FocusEvent.
member(String)	Find methods	Finds contained members which have an exact type match.
member(String, String)	Find methods	Finds contained members which have exact type and label matches.
member(String, int)	Find methods	Finds a member which have an exact type match.
member(String, String, int)	Find methods	Finds a member which has an exact type and label match. Uses a depth-first seach.
memberRE(String, String)	Find methods	Finds contained members which have a type and a label matching a regular expression.
memberRX(String, String)	Deprecated methods	Finds contained members which have a type and a label matching regular expression. Uses old regular expression syntax.
menubar()	Find methods	Returns the menubar. Valid only for Frames.
mouseClick(int, int, int)	Deprecated methods	Sends a `MOUSE_CLICKED` event.
mouseClicked(int, int, int)	Send event methods	Sends a `MOUSE_CLICKED` MouseEvent.
mouseDown(int, int, int)	Deprecated methods	Sends a `MOUSE_DOWN` event.
mouseDrag(int, int, int)	Deprecated methods	Sends a `MOUSE_DRAG` event.
mouseDragged(int, int, int)	Send event methods	Sends a `MOUSE_DRAGGED` MouseEvent.
mouseEnter(int, int)	Deprecated methods	Sends a `MOUSE_ENTER` event.
mouseEntered(int, int)	Send event methods	Sends a `MOUSE_ENTERED` MouseEvent.
mouseExit(int, int)	Deprecated methods	Sends a `MOUSE_EXIT` event.
mouseExited(int, int)	Send event methods	Sends a `MOUSE_EXITED` MouseEvent.

Table 2-3 `JSComponent` methods by name

Method	Category	Description
mouseMove(int, int, int)	Deprecated methods	Sends a `MOUSE_MOVE` event.
mouseMoved(int, int)	Send event methods	Sends a `MOUSE_MOVED` MouseEvent.
mousePressed(int, int, int)	Send event methods	Sends a `MOUSE_PRESSED` MouseEvent.
mouseReleased(int, int, int)	Send event methods	Sends a `MOUSE_RELEASED` MouseEvent.
mouseUp(int, int, int)	Deprecated methods	Sends a `MOUSE_UP` event.
multiClick(int, int, int, int)	Send event methods	Sends multiple groups of `MOUSE_PRESSED`, `MOUSE_RELEASED`, and `MOUSE_CLICKED` events.
north(String, String)	Find methods	Returns the North member, requiring that it meet specified type and label requirements.
orphan(String)	Deprecated methods	Finds Components whose ancestor is this, but who are not children of this, and that have an exact type match.
pageDown(int)	Scrolling methods	Scrolls down a page.
pageUp(int)	Scrolling methods	Scrolls up a page.
popup(String)	Find methods	Returns all popups with the specified label.
popup(String, int)	Find methods	Returns all popups with the specified label, and includes a count.
popupTrigger(int, int)	Send event methods	Sends the PopupTrigger MouseEvent appropriate for the platform.
readString()	Convenience methods	Returns the String value of the component.
relativefile(String, String)	Send event methods	FileDialog simulation event,
select()	Send event methods	Sets the state of a Checkbox (sends a `SELECTED` ItemEvent).
select(int)	Send event methods	Selects the indexed entry of a List.
select(String)	Send event methods	Selects an item in a List.

Table 2-3 `JSComponent` methods by name

Method	Category	Description
select(int, String)	Send event methods	Selects the indexed entry of a List. Requires that the text of the item match.
south(String, String)	Find methods	Returns the South member, requiring that it meet specified type and label requirements.
synchronize(Script, String, String)	Comparison methods	Waits until the component matches the specified gold file. If timeout occurs before a match, SyncException occurs.
typeString(String)	Send event methods	Sends `KEY_PRESS` and `KEY_RELEASE` events for each character in the String, to type the entire string.
typeString(String, int, int)	Send event methods	Same as `typeString(String)`, except that you specify the start and end position for the string.
verify(Script, boolean, String)	Comparison methods	Verify whether or not the component is capable of receiving events (see isValidUnique).
verify(Script, String, String)	Comparison methods	Verify that the component has the specified string value.
verifyWithFile(Script, String, String)	Comparison methods	Verify the component against the specified gold file.
verifyAnyField(Script, boolean, boolean, String, Object, String)	Comparison methods	Reads the data member name on the contained Component and compares this against the expected value.
verifyAnyMethod(Script, boolean, boolean, String, Object, String)	Comparison methods	Calls the method name() on the contained Component and compares it against the expected value.
waitFor(boolean, String)	Comparison methods	Waits until a Component pointed to by this `JSComponent` is correctly enabled or disabled.
waitFor(String)	Comparison methods	Waits until a Component pointed to by this `JSComponent` is capable of receiving events.
west(String, String)	Find methods	Waits until a Component pointed to by this `JSComponent` contains str.
west(String, String)	Find methods	Returns the West member, requiring that it meet specified type and label requirements.
windowMoved(int, int, int, int)	Send event methods	Window simulation event that reshapes the Window as specified.

Methods by Category

Table 2-4 `JSComponent` Methods by Category

Category	Method	Description
Comparison methods These methods compare component state to expected values or verify the state of attributes.	synchronize(Script, String, String)	Waits until the component matches the specified gold file. If timeout occurs before a match, SyncException occurs.
	verify(Script, boolean, String)	Verify whether or not the component is capable of receiving events (see isValidUnique).
	verify(Script, String, String)	Verify that the component has the specified string value.
	verifyWithFile(Script, String, String)	Verify the component against the specified gold file.
	verifyAnyField(Script, boolean, boolean, String, Object, String)	Reads the data member name on the contained Component and compares this against the expected value.
	verifyAnyMethod(Script, boolean, boolean, String, Object, String)	Calls the method name() on the contained Component and compares it against the expected value.
	waitFor(boolean, String)	Waits until a Component pointed to by this `JSComponent` is correctly enabled or disabled.
	waitFor(String)	Waits until a Component pointed to by this `JSComponent` is capable of receiving events.
Convenience methods Convenience methods are a grouping of miscellaneous methods that perform functions on Components.	getAnyField(String)	Returns the value of the data member name on the contained Component.
	readString()	Returns the String value of the component.
	getAnyMethod(String)	Returns the value of the method `name()` on the contained Component.

Table 2-4 `JSComponent` Methods by Category

Category	Method	Description
Deprecated methods These methods will be phased out of JavaStar in the future, but are included in this release to support scripts written with earlier versions of the JavaStar library.	buttonRX(String)	Finds contained buttons which have labels matching this regular expression. Uses old syntax for regular expressions.
	dialogRX(String)	Returns dialogs that have a title matching this regular expression.
	dialogRX(String, String)	Returns dialogs that meet specified type and title matching regular expression.
	hasMemberRX(String, String)	Equivalent to `hasMemberRE()`, except that this method uses the old regular expression syntax.
	keyTyped(int, int)	Sends a `KEY_TYPED` KeyEvent. Replaced by `keyTyped(char)`.
	memberRX(String, String)	Finds contained members which have a type and a label matching regular expression. Uses old regular expression syntax.
	mouseClick(int, int, int)	Sends a `MOUSE_CLICKED` event.
	mouseDown(int, int, int)	Sends a `MOUSE_DOWN` event.
	mouseDrag(int, int, int)	Sends a `MOUSE_DRAG` event.
	mouseEnter(int, int)	Sends a `MOUSE_ENTER` event.
	mouseExit(int, int)	Sends a `MOUSE_EXIT` event.
	mouseMove(int, int, int)	Sends a `MOUSE_MOVE` event.
	mouseUp(int, int, int)	Sends a `MOUSE_UP` event.
	orphan(String)	Finds Components whose ancestor is this, but who are not children of this, and that have an exact type match.

Table 2-4 `JSComponent` Methods by Category

Category	Method	Description
Find methods Find methods locate a component based on its label, position, or other attributes. Most find methods return a `JSComponent` or `JSMenuComponent`.	ageDifferentiation(int, int)	Disambiguates based on relative construction time (approximately).
	button(String)	Finds contained buttons which have an exact label match.
	buttonRE(String)	Finds contained buttons which have labels matching this regular expression.
	center(String, String)	Returns the Center member that meets the specified type and label requirements.
	child(int, String, String)	Returns a child component that meets the specified type and label requirements.
	dialog(String)	Returns dialogs of this Frame whose title matches title.
	dialog(String, String)	Returns dialogs of this Frame that meet specified type and title requirements.
	dialogRE(String)	Returns dialogs that have title matching regular expression.
	dialogRE(String, String)	Returns dialogs of this Frame of the specified type that have title matching regular expression.
	east(String, String)	Returns the East member, requiring that it meet specified type and label requirements.
	hasMember(String)	Returns any components of `JSComponent` that match the type you specify.
	hasMember(String, String)	Returns a `JSComponent` that contains the subset of this `JSComponent`'s Components which contain at least one member matching the specification.
	hasMemberRE(String, String)	Same as `hasMember()`, except this method accepts regular expressions for the Component title.
	lookup(String)	Finds any components with a matching name contained in the components.
	member(String)	Finds contained members which have an exact type match.
	member(String, String)	Finds contained members which have exact type and label matches.
	member(String, String, int)	Finds a member which has an exact type and label match. Uses a depth-first seach.

Table 2-4 `JSComponent` Methods by Category

Category	Method	Description
Query methods These methods return `Components` pointed to by a `JSComponent`.	getAll()	Returns all Components pointed to by this `JSComponent`.
	getAllValid()	Returns all Components pointed to by this `JSComponent` which are capable of receiving events.
	getUnique()	Returns the Component pointed to by this `JSComponent`.
	getValidUnique()	Returns the only Component pointed to by this `JSComponent` which is capable of receiving events.
	isUnique()	Returns true if there is exactly one Component contained by this `JSComponent`.
	isValidUnique()	Returns true if there is one valid match that can receive events.
Scrolling methods These methods scroll a TextComponent to the position you specify.	absolute(int)	Scrolls to the position you specify.
	pageDown(int)	Scrolls down a page.
	pageUp(int)	Scrolls up a page.
	lineDown(int)	Scrolls down one line.
	lineUp(int)	Scrolls up one line.

Table 2-4 `JSComponent` Methods by Category

Category	Method	Description
Send event methods These methods send events to GUI components.	action()	Sends an `ACTION` event to a Button or Textfield.
	action(String)	Sends an action event to Button, Choice, List, or TextField.
	buttonPress()	Sends `MOUSE_PRESSED`, `MOUSE_RELEASED`, `MOUSE_CLICKED`, and the ActionEvent. Valid only for Buttons.
	choosefile(String, String)	FileDialog simulation event.
	deiconify()	Sends a `WINDOW_DEICONIFY` event.
	deselect()	Sends a `DESELECTED` ItemEvent to a Checkbox.
	deselect(int)	Deselects the indexed entry of a List.
	deselect(String)	Deselects an item in a List.
	deselect(int, String)	Deselects the indexed entry of a List. Requires that the text of the item match.
	destroy()	Sends a `WINDOW_CLOSING` event.
	fkey(int, int)	Sends a `KEY_ACTION` KeyEvent.

Table 2-4 `JSComponent` Methods by Category

Category	Method	Description
Send event methods, continued	fkeyUp(int, int)	Sends a `KEY_ACTION_RELEASE` KeyEvent.
	gotFocus()	Sends a `FOCUS_GAINED` FocusEvent.
	iconify()	Sends a `WINDOW_ICONIFY` event.
	key(int, int)	Sends a `KEY_PRESSED` KeyEvent.
	key(int, int, int, int)	Typing simulation event.
	keyPressed(int, char, int)	Sends a `KEY_PRESSED` KeyEvent.
	keyReleased(int, char, int)	Sends a `KEY_RELEASED` KeyEvent.
	keyTyped(char)	Sends a `KEY_TYPED` KeyEvent.
	keyUp(int, int)	Sends a `KEY_RELEASED` KeyEvent.
	lostFocus()	Sends a `FOCUS_LOST` FocusEvent.
	mouseClicked(int, int, int)	Sends a `MOUSE_CLICKED` MouseEvent.
	mouseDragged(int, int, int)	Sends a `MOUSE_DRAGGED` MouseEvent.
	mouseEntered(int, int)	Sends a `MOUSE_ENTERED` MouseEvent.
	mouseExited(int, int)	Sends a `MOUSE_EXITED` MouseEvent.
	mouseMoved(int, int)	Sends a `MOUSE_MOVED` MouseEvent.
	mousePressed(int, int, int)	Sends a `MOUSE_PRESSED` MouseEvent.
	mouseReleased(int, int, int)	Sends a `MOUSE_RELEASED` MouseEvent.
	multiClick(int, int, int, int)	Sends multiple groups of `MOUSE_PRESSED`, `MOUSE_RELEASED`, and `MOUSE_CLICKED` events.
	popupTrigger(int, int)	Sends the PopupTrigger MouseEvent appropriate for the platform.
	relativefile(String, String)	FileDialog simulation event,
	select()	Sets the state of a Checkbox (sends a `SELECTED` ItemEvent).
	select(int)	Selects the indexed entry of a List.
	select(String)	Selects an item in a List.
	select(int, String)	Selects the indexed entry of a List. Requires that the text of the item match.
	typeString(String)	Sends `KEY_PRESS` and `KEY_RELEASE` events for each character in the String, to type the entire string.

Table 2-4 `JSComponent` Methods by Category

Category	Method	Description
Send event methods, continued	windowMoved(int, int, int, int)	Window simulation event that reshapes the Window as specified.

action()

Syntax

```
public void action()
```

Category

Send event method

Description

Valid only for Buttons and TextFields: sends an `ACTION` event with the appropriate String arg (the current button label or textfield text).

action(String)

Syntax

```
public void action(String text)
```

Category

Send event method

Description

Sends an action event to Button, Choice, List, TextField.

Parameters

- *text* – the label or the title of the object you want to send the event to

buttonPress()

Syntax

```
public void buttonPress()
```

Category

Send event method

Description

Valid only for Buttons. Sends `MOUSE_PRESSED`, `MOUSE_RELEASED`, `MOUSE_CLICKED`, and the ActionEvent. For non-Buttons, sends only the three mouse events, at the center of the Component.

choosefile(String, String)

Syntax

```
public void choosefile(String dir, String file)
```

Category

Send event method

Description

FileDialog simulation event. Valid only for FileDialogs.

Parameters

- *dir* – the directory chosen
- *file* – the file chosen (may be null if this is a cancel event)

deiconify()

Syntax

```
public void deiconify()
```

Category

Send event method

Description

Sends a `WINDOW_DEICONIFY` event.

deselect()

Syntax

```
public void deselect()
```

Category

Send event method

Description

Sets the state of a Checkbox (sends a `DESELECTED` ItemEvent).

deselect(int)

Syntax

```
public void deselect(int index)
```

Category

Send event method

Description

Deselects the indexed entry of a List.

Parameters

- *index* – the index to deselect

deselect(String)

Syntax

```
public void deselect(String item)
```

Category

Send event method

Description

Deselects an item in a List.

Parameters

- *item* – the value of the item to deselect

deselect(int, String)

Syntax

```
public void deselect(int index, String item)
```

Category

Send event method

Description

Deselects the indexed entry of a List. Requires that the text of the item match.

Parameters

- *index* – the index to deselect
- *item* – the value of the item

destroy()

Syntax

```
public void destroy()
```

Category

Send event method

Description

Sends a `WINDOW_CLOSING` event.

fkey(int, int)

Syntax

```
public void fkey(int keycode, int modifiers)
```

Category

Send event method

Description

Sends a `KEY_ACTION` KeyEvent.

Parameters

- *keycode* – the key that was pressed
- *modifiers* – the state of the modifier keys

fkeyUp(int, int)

Syntax

```
public void fkeyUp(int keycode, int modifiers)
```

Category

Send event method

Description

Sends a `KEY_ACTION_RELEASE` KeyEvent.

Parameters

- *keycode* – the key that was pressed
- *modifiers* – the state of the modifier keys

gotFocus()

Syntax

```
public void gotFocus()
```

Category

Send event method

Description

Sends a `FOCUS_GAINED` FocusEvent.

iconify()

Syntax

```
public void iconify()
```

Category

Send event method

Description

Sends a `WINDOW_ICONIFY` event.

key(int, int)

Syntax

```
public void key(int keychar, int modifiers)
```

Category

Send event method

Description

Sends a `KEY_PRESSED` KeyEvent.

Parameters

- *keychar* – the char value of the key that was pressed
- *modifiers* – the state of the modifier keys

key(int, int, int, int)

Syntax

```
public void key(int key, int modifiers, int selectionStart, int selectionEnd)
```

Category

Send event method

Description

Typing simulation event—valid only for TextComponents.

Parameters

- *key* – the key that was pressed
- *modifiers* – the state of the modifier keys
- *selectionStart* – the selection-start position before typing
- *selectionEnd* – the selection-end position before typing

keyPressed(int, char, int)

Syntax

```
public void keyPressed(int keyCode, char keyChar, int modifiers)
```

Category

Send event method

Description

Sends a KEY_PRESSED KeyEvent.

Parameters

- *keyCode* – the keycode for the key
- *keyChar* – the char value of the key that was pressed
- *modifiers* – the state of the modifier keys

keyReleased(int, char, int)

Syntax

```
public void keyReleased(int keyCode, char keyChar, int modifiers)
```

Category

Send event method

Description

Sends a KEY_RELEASED KeyEvent.

Parameters

- *keyCode* – the keycode for the key
- *keyChar* – the char value of the key that was pressed
- *modifiers* – the state of the modifier keys

keyTyped(char)

Syntax

```
public void keyTyped(char keyChar)
```

Category

Send event method

Description

Sends a KEY_TYPED KeyEvent.

Parameters

- *keyChar* – the char value of key that was pressed

keyUp(int, int)

Syntax

```
public void keyUp(int keychar, int modifiers)
```

Category

Send event method

Description

Sends a `KEY_RELEASED` KeyEvent.

Parameters

- *keychar* – the char value of the key that was pressed
- *modifiers* – the state of the modifier keys

lostFocus()

Syntax

```
public void lostFocus()
```

Category

Send event method

Description

Sends a `FOCUS_LOST` FocusEvent.

mouseClicked(int, int, int)

Syntax

```
public void mouseClicked(int x, int y, int modifiers)
```

Category

Send event method

Category

Send event method

Description

Sends a `MOUSE_CLICKED` MouseEvent.

Parameters

- *x* - the x-coordinate
- *y* - the y-coordinate
- *modifiers* - the state of the modifier keys

mouseDragged(int, int, int)

Syntax

```
public void mouseDragged(int x, int y, int modifiers)
```

Category

Send event method

Description

Sends a `MOUSE_DRAGGED` MouseEvent.

Parameters

- *x* - the x-coordinate
- *y* - the y-coordinate
- *modifiers* - the state of the modifier keys

mouseEntered(int, int)

Syntax

```
public void mouseEntered(int x, int y)
```

Category

Send event method

Description

Sends a `MOUSE_ENTERED` MouseEvent.

Parameters

- *x* - the x-coordinate
- *y* - the y-coordinate

mouseExited(int, int)

Syntax

```
pubic void mouseExited(int x, int y)
```

Category

Send event method

Description

Sends a `MOUSE_EXITED` MouseEvent.

Parameters

- *x* - the x-coordinate
- *y* - the y-coordinate

mouseMoved(int, int)

Syntax

```
pubic void mouseMoved(int x, int y, int modifiers)
```

Category

Send event method

Description

Sends a `MOUSE_MOVED` MouseEvent.

Parameters

- *x* - the x-coordinate
- *y* - the y-coordinate
- *modifiers* - the state of the modifier keys

mousePressed(int, int, int)

Syntax

```
pubic void mousePressed(int x, int y, int modifiers)
```

Category

Send event method

Description

Sends a `MOUSE_PRESSED` MouseEvent.

Parameters

- *x* - the x-coordinate
- *y* - the y-coordinate
- *modifiers* - the state of the modifier keys

mouseReleased(int, int, int)

Syntax

```
pubic void mouseReleased(int x, int y, int modifiers)
```

Category

Send event method

Description

Sends a `MOUSE_RELEASED` MouseEvent.

Parameters

- *x* - the x-coordinate
- *y* - the y-coordinate
- *modifiers* - the state of the modifier keys

multiClick(int, int, int, int)

Syntax

```
public void multiClick(int x, int y, int modifiers, int count)
```

Category

Send event method

Description

Simulation event. This method sends multiple groups of `MOUSE_PRESSED`, `MOUSE_RELEASED`, and `MOUSE_CLICKED` events.

You can use this method to simulate a double-click—for example, `multiClick(x, y, 0, 2)`. For Buttons, this method sends an `ACTION` even after each `CLICKED`.

Parameters

- *x* – the x-coordinate
- *y* – the y-coordinate
- *modifiers* – the state of the modifier keys
- *count* – the number of clicks (one or more)

popupTrigger(int, int)

Syntax

```
pubic void popupTrigger(int x, int y)
```

Category

Send event method

Description

Sends the PopupTrigger MouseEvent appropriate for the platform. (On Solaris, a `MOUSE_RELEASED`; on Windows, a `MOUSE_PRESSED`).

Parameters

- *x* - the x-coordinate
- *y* - the y-coordinate

select()

Syntax

```
public void select()
```

Category

Send event method

Description

Sets the state of a Checkbox (sends a `SELECTED` ItemEvent).

select(int)

Syntax

```
public void select(int index)
```

Category

Send event method

Description

Selects the indexed entry of a List.

Parameters

- *index* – the index to select

select(String)

Syntax

```
public void select(String item)
```

Category

Send event method

Description

Selects an item in a List.

Parameters

- *item* – the value of the item to select

select(int, String)

Syntax

```
public void select(int index, String item)
```

Category

Send event method

Description

Selects the indexed entry of a List. Requires that the text of the item match.

Parameters

- *index* – the index to select
- *item* – the value of the item

typeString(String)

Syntax

```
public void typeString(String data)
```

Category

Send event method

Description

Sends KEY_PRESS and KEY_RELEASE events for each character, to type the entire string. For TextComponents, this will select the entire contents and overwrites it.

Parameters

- *data* – the string to type

typeString(String, int, int)

Syntax

```
public void typeString(String data, int selectionStart, int
selectionEnd)
```

Category

Send event method

Description

Valid only for TextComponents: sends KEY_PRESS and KEY_RELEASE events for each character, to type the entire string into the TextComponent.

Parameters

- *data* – the string to simulate typing
- *selectionStart* – the selection-start position before the typing
- *selectionEnd* – the selection-end position before after the typing

relativefile(String, String)

Syntax

```
public relativefile(String rdir, String fil)
```

Category

Send event method

Description

FileDialog simulation event - valid only for FileDialogs.

Parameters

- *rdir* – the directory chosen as relative path to output directory.
- *fil* – the file chosen (may be null if a cancel event)

windowMoved(int, int, int, int)

Syntax

```
public void windowMoved(int x, int y, int width, int height)
```

Category

Send event method

Description

Window simulation event that sizes and positions the Window as specified. Valid only for Window Components.

Parameters

- *x* – the new x position
- *y* – the new y position
- *width* – the new width
- *height* – the new height

ageDifferentiation(int, int)

Syntax

```
public JSComponent ageDifferentiation(int position, int total)
```

Category

Find method

Description

Disambiguates Components based on their relative construction time.

Parameters

- *position* – position in `getAll()` list
- *total* – expected length of `getAll()` list

button(String)

Syntax

```
public JSComponent button(String val)
```

Category

Find method

Description

Finds contained buttons which have an exact label match.

Parameters

- *val* – exact label of Button

buttonRE(String)

Syntax

```
pubic JSComponent buttonRE(String val)
```

Category

Find method

Description

Finds contained buttons which have labels matching this regular expression. See Current Syntax in the chapter "Syntax for Regular Expressions."

Parameters

- *val* - Regular expression label of Button

center(String, String)

Syntax

```
public JSComponent center(String type, String label)
```

Category

Find method

Description

Returns the Center member that meets the specified type and label requirements. Valid only for Containers with BorderLayouts.

Parameters

- *type* – full class name
- *label* – exact label of Button (Use "" for non-Buttons)

child(int, String, String)

Syntax

```
public JSComponent child(int i, String type, String label)
```

Category

Find method

Description

Returns a child component that meets the specified type and label requirements. Valid only for Containers.

Parameters

- *i* – which member (position within Container's list.)
- *type* – full class name
- *label* – exact label of Button (Use "" for non-Buttons)

dialog(String)

Syntax

```
public JSComponent dialog(String title)
```

Category

Find method

Description

Returns dialogs of this Frame whose title matches *title*. Valid only for Frames.

Parameters

- *title* – exact title of Dialog

dialog(String, String)

Syntax

```
public JSComponent dialog(String type, String title)
```

Category

Find method

Description

Returns dialogs of this Frame that meet specified type and title requirements. Valid only for Frames.

Parameters

- *type* – full class name
- *title* – exact title of Dialog

dialogRE(String)

Syntax

```
pubic JSComponent dialogRE(String expression)
```

Category

Find method

Description

Returns dialogs that have title matching regular expression. See Current Syntax in the chapter "Syntax for Regular Expressions." Valid only for Frames.

Parameters

- *expression* - regular expression title of Dialog

dialogRE(String, String)

Syntax

```
pubic JSComponent dialogRE(String type, String expression)
```

Category

Find method

Description

Returns dialogs of this Frame of the specified type that have title matching regular expression. See Current Syntax in the chapter "Syntax for Regular Expressions." Valid only for Frames.

Parameters

- *type* - full class name
- *expression* - regular expression title of Dialog

east(String, String)

Syntax

```
public JSComponent east(String type, String label)
```

Category

Find method

Description

Returns the East member, requiring that it meet specified type and label requirements. Valid only for Containers with BorderLayouts.

Parameters

- *type* – full class name
- *label* – exact label of Button (use "" for non-Buttons)

hasMember(String)

Syntax

```
public JSComponent hasMember(String type)
```

Category

Find method

Description

Disambiguates – finds the subset of this `JSComponent`'s Component that contian at least one member of the *type* you specify. Returns a `JSComponent` containing this subset.

Parameters

- *type* – full class name of the component

hasMember(String, String)

Syntax

```
public JSComponent hasMember(String type, String val)
```

Category

Find method

Description

Disambiguates – looks for a member of a this JSComponent that contains at least one member matching the *type* and *val*. Returns a JSComponent containing this subset.

Parameters

- *type* – full class name of the component
- *val* – exact label of Button

hasMemberRE(String, String)

Syntax

```
pubic JSComponent hasMemberRE(String type, String expression)
```

Category

Find method

Description

Disambiguates – looks for a member of a this JSComponent that contains at least one member matching this *type* and have a label matching this regular *expression*. (See Current Syntax in the chapter "Syntax for Regular Expressions.") Returns a JSComponent containing this subset.

lookup(String)

Syntax

```
public JSComponent lookup(String name)
```

Category

Find method

Description

Finds any components with a matching *name* (return of the `getName()` method) contained in the components returned by the `getAll()` method.

This method uses '.' as a scope separator. For example, *name* of `A.B` refers to a component named `B` contained in a component named `A`, where `A` is contained within the current component. `'#' must be used to escape '.'`

Parameters

- *name* – the name of a component

member(String)

Syntax

```
public JSComponent member(String type)
```

Category

Find method

Description

Finds contained members which have an exact type match.

Parameters

- *type* – full class name of Component

member(String, String)

Syntax

```
public JSComponent member(String type, String val)
```

Category

Find method

Description

Finds contained members which have exact type and label matches.

Parameters

- *type* – full class name of Component
- *val* – exact label of Button or Label or Checkbox

member(String, String, int)

Syntax

```
public JSComponent member(String type, String val, int position);
```

Category

Find method

Description

Finds a member which has an exact type and label match. Uses a depth-first search of the tree, return the *position*'th Component that matches.

Parameters

- *type* - full class name of Component
- *val* – exact label of Button or Label or Checkbox
- *position* - search-order position (0 or higher)

member(String, int)

Syntax

```
pubic JSComponent member(String type, int position)
```

Category

Find method

Description

Finds a member which have an exact type match. With a depth-first search of the tree, returns the position of the Component matching the given *type.*

Parameters

- *type* - full class name of Component
- *position* - search-order position (0 or higher)

memberRE(String, String)

Syntax

```
pubic JSComponent memberRE(String type, String expression)
```

Category

Find method

Description

Finds contained members which have a type matching type and a label matching this regular expression. See Current Syntax in the chapter "Syntax for Regular Expressions."

Parameters

- *type* - full class name
- *val* – Regular expression label of Button or Label or Checkbox

menubar()

Syntax

```
public JSMenuComponent menubar()
```

Category

Find method

Description

Returns the menubar. Valid only for Frames.

north(String, String)

Syntax

```
public JSComponent north(String type, String label)
```

Category

Find method

Description

Returns the North member, requiring that it meet specified type and label requirements. Valid only for Containers with BorderLayouts.

Parameters

- *type* – full class name
- *label* – exact label of Button (use "" for non-Buttons)

popup(String)

Syntax

```
public JSMenuComponent popup(String label)
```

Category

Find method

Description

Returns all popups with the specified label.

Parameters

- *label* – the label of the popup to find

popup(String, int)

Syntax

```
public JSMenuComponent popup(String label, int count)
```

Category

Find method

Description

Returns all popups with the specified label.

Parameters

- *label* – the label of the popup to find
- *count* – number of popups returned.

south(String, String)

Syntax

```
public JSComponent south(String type, String label)
```

Category

Find method

Description

Returns the South member, requiring that it meet specified type and label requirements. Valid only for Containers with BorderLayouts.

Parameters

- *type* – full class name
- *label* – exact label of Button (use "" for non-Buttons)

west(String, String)

Syntax

```
public JSComponent west(String type, String label)
```

Category

Find method

Description

Returns the West member, requiring that it meet specified type and label requirements. Valid only for Containers with BorderLayouts.

Parameters

- *type* – full class name
- *label* – exact label of Button (use "" for non-Buttons)

getAll()

Syntax

```
public Component[] getAll()
```

Category

Query method

Description

Returns all Components pointed to by this `JSComponent`.

getAllValid()

Syntax

```
public Component[] getAllValid()
```

Category

Query method

Description

Returns all Components pointed to by this `JSComponent` which are capable of receiving events.

getUnique()

Syntax

```
public Component getUnique()
```

Description

Returns the Component pointed to by this `JSComponent`. This requires that there is exactly one match, throwing an exception otherwise.

getValidUnique()

Syntax

```
public Component getValidUnique()
```

Category

Query method

Description

Returns the only Component pointed to by this `JSComponent` which is capable of receiving events. If there are no components capable of receiving events, or if more than one is capable of receiving, this method throws an exception.

isUnique()

Syntax

```
public boolean isUnique()
```

Category

Query method

Description

Returns `true` if there is exactly one Component contained by this `JSComponent`.

isValidUnique()

Syntax

```
public boolean isValidUnique()
```

Category

Query method

Description

Returns true if there is one valid match that can receive events.

synchronize(Script, String, String)

Syntax

`public void synchronize(Script` *sc*`, String` *filename*`, String` *purpose*`)`

Category

Comparison method

Description

Waits until the component matches the specified gold file. If timeout occurs before a match, `SyncException` occurs.

Parameters

- *sc* – the the script where the comparison occurs
- *filename* – name of gold file
- *purpose* – the purpose of this comparison

verify(Script, boolean, String)

Syntax

```
public boolean verify(Script sc, boolean en, String purpose)
```

Category

Comparison method

Description

Verify whether or not the component is capable of receiving events (see `isValidUnique`). Returns `true` if the verify succeeds.

Parameters

- *sc* – the script where you are performing the verify
- *en* – `true` means this component must be capable of receiving events; `false` means it must not be capable
- *purpose* – the purpose of this comparison

verify(Script, String, String)

Syntax

```
public boolean verify(Script sc, String value, String purpose)
```

Category

Comparison method

Description

Verify that the component has the specified string value. Returns `true` if verify succeeds.

Parameters

- *sc* – the script where you are performing the comparison
- *value* – expected string value
- *purpose* – the purpose of this comparison

verifyAnyField(Script, boolean, boolean, String, Object, String)

Syntax

```
public boolean verifyAnyField(Script sc, boolean sync, boolean
retry, String name, Object expected, String purpose)
```

Category

Comparison method

Description

Reads the data member *name* on the contained Component and compares this against the expected value.

Parameters

- *sc* – the script where you are performing the comparison
- *sync* – if set to true, this throws a SyncException on failure
- *retry* – if set to true, this method continues checking until timeout
- *name* – the name of the non-static data member to verify
- *expected* – the expected value of the method
- *purpose* – the purpose of the verify operation

verifyAnyMethod(Script, boolean, boolean, String, Object, String)

Syntax

```
public boolean verifyAnyMethod(Script sc, boolean sync, boolean
retry, String name, Object expected, String purpose)
```

Category

Comparison method

Description

Calls the method *name()* on the contained Component and compares it against the expected value. Returns `true` or `false` depending on the comparison results.

Parameters

- *sc* – the script where you are performing the comparison
- *sync* – if set to true, this throws an SyncException on failure
- *retry* – if set to true, this method continues checking until timeout
- *name* – the name of the method to verify (must not take parameters)
- *expected* – the expected value of the method
- *purpose* – the purpose of the verify operation

verifyWithFile(Script, String, String)

Syntax

```
public boolean verifyWithFile(Script sc, String filename, String
purpose)
```

Category

Comparison method

Description

Verify the component against the specified gold file. Returns `true` if the verify succeeds.

Parameters

- *sc* – the script where you are performing the comparison

- *filename* – name of gold file
- *purpose* – the purpose of this comparison

waitFor(boolean, String)

Syntax

```
public void waitFor(boolean enabled, String purpose)
```

Category

Comparison method

Description

Waits until a Component pointed to by this `JSComponent` is correctly enabled or disabled. If a timeout occurs, this method throws a GUINotAcceptingException.

Parameters

- *enabled* – if true, the Component must be enabled to pass; if false, the component must be disabled
- *purpose* – the purpose of this synchronization

waitFor(String)

Syntax

```
public void waitFor(String purpose)
```

Category

Comparison method

Description

Waits until a Component pointed to by this `JSComponent` is capable of receiving events. If timeout occurs, this method throw a `GUINotAcceptingException`.

Parameters

- *purpose* – the purpose of this synchronization

waitFor(String, String)

Syntax

```
public void waitFor(String str, String purpose)
```

Category

Comparison method

Description

Waits until a Component pointed to by this `JSComponent` contains *str*. Valid for Frames, Dialogs, Labels, Buttons, TextAreas, and TextFields. Timeout will cause `GUINotAcceptingException`.

Parameters

- *str* – the label, title, or contents of the Component
- *purpose* – the purpose of this synchronization

absolute(int)

Syntax

```
public void absolute(int position)
```

Category

Scrolling method

Description

Scrolls to *position* and sends a `TRACK` AdjustmentEvent.

Parameters

- *position* – the position to which to scroll

lineDown(int)

Syntax

```
public void lineDown(int position)
```

Category

Scrolling method

Description

Scrolls to *position* and sends a `UNIT_DECREMENT` AdjustmentEvent.

Parameters

- *position* – the position to which to scroll

lineUp(int)

Syntax

```
public void lineUp(int position)
```

Category

Scrolling method

Description

Scrolls to *position* and sends a `UNIT_INCREMENT` AdjustmentEvent.

Parameters

- *position* – the position to which to scroll

pageDown(int)

Syntax

```
public void pageDown(int position)
```

Category

Scrolling method

Description

Scrolls to *position* and sends a `BLOCK_DECREMENT` AdjustmentEvent.

Parameters

- *position* – the position to which to scroll

pageUp(int)

Syntax

```
public void pageUp(int position)
```

Category

Scrolling method

Description

Scrolls to *position* and sends a `BLOCK_INCREMENT` AdjustmentEvent.

Parameters

- *position* – the position to which to scroll

getAnyField(String)

Syntax

```
public Object getAnyField(String name) throws
NoSuchFieldException, IllegalAccessException
```

Category

Convenience method

Description

This convenience function returns the value of the data member *name* on the contained Component.

Parameters

- *name* – the name of the non-static data member

getAnyMethod(String)

Syntax

```
public Object getAnyMethod(String name) throws
noSuchMethodException, IllegalAccessException,
InvocationTargetException
```

Category

Convenience method

Description

This convenience function returns the value of the method *name()* on the contained Component.

Parameters

- *name* – the name of the method to check (must not take parameters)

readString()

Syntax

```
public String readString()
```

Category

Convenience method

Description

Returns the String value of the component. The value returned is based on the component type:

buttonRX(String)

Syntax

```
public JSComponent buttonRX(String val)
```

Category

Deprecated method

Description

Replaced by buttonRE(String).

Finds contained buttons which have labels matching this regular expression (see Current Syntax in the chapter "Syntax for Regular Expressions.").

Parameters

- *val* – Regular expression label of Button

orphan(String)

Syntax

```
public JSComponent orphan(String type)
```

Category

Deprecated method

Description

This method is obsolete; there is no replacement.

Finds Components whose ancestor is this, but who are not children of this, and that have an exact type match.

> ***Note –*** This may be generated by a record. Usually this indicates unusual usage of the AWT (such as overriding `getComponents()` or `getParent()` methods).

Parameters

- *type* – full class name of Component

dialogRX(String)

Syntax

```
public JSComponent dialogRX(String expression)
```

Category

Deprecated method

Description

Replaced by dialogRE(String).

Returns dialogs that have a title matching this regular expression (see Current Syntax in the chapter "Syntax for Regular Expressions"). Valid only for Frames.

Parameters

- *expression* – regular expression title of Dialog

dialogRX(String, String)

Syntax

```
public JSComponent dialogRX(String type, String expression)
```

Category

Deprecated method

Description

Replaced by dialogRE(String, String).

Returns dialogs that meet specified type and title matching regular expression (see Current Syntax in the chapter "Syntax for Regular Expressions"). Valid only for Frames.

Parameters

- *type* – full class name
- *expression* – regular expression title of Dialog

hasMemberRX(String, String)

Syntax

```
public JSComponent hasMemberRX(String type, String expression)
```

Category

Deprecated method

Description

Replaced by hasMemberRE(String, String).

Disambiguates – returns a `JSComponent` that contains the subset of this `JSComponent`'s Components which contain at least one member which have type & label matching regular expression (see Current Syntax in the chapter "Syntax for Regular Expressions").

Parameters

- *type* – full class name
- *expression* – regular expression label of Button

keyTyped(int, int)

Syntax

```
pubic void keyTyped(int keyChar, int modifiers)
```

Category

Deprecated method

Description

Replaced by keyTyped(char).

Backward-compatibility only - not recommended. Sends a `KEY_TYPED` KeyEvent.

Parameters

- *keyChar* - the char value of key that was pressed
- *modifiers* - the state of the modifier keys

memberRX(String, String)

Syntax

```
public JSComponent memberRX(String type, String expression)
```

Category

Deprecated method

Description

Replaced by memberRE(String, String).

Finds contained members which have a type and a label matching regular expression (see the chapter "Syntax for Regular Expressions").

Parameters

- *type* – full class name
- *expression* – regular expression label of Button

mouseClick(int, int, int)

Syntax

```
public void mouseClick(int x, int y, int modifiers)
```

Category

Deprecated method

Description

Replaced by mouseClicked(int, int, int).

Sends a `MOUSE_CLICKED` event.

Parameters

- *x* – the x-coordinate
- *y* – the y-coordinate
- *modifiers* – the state of the modifier keys

mouseDown(int, int, int)

Syntax

```
public void mouseDown(int x, int y, int modifiers)
```

Category

Deprecated method

Description

Replaced by mousePressed(int, int, int).

Sends a `MOUSE_DOWN` event.

Parameters

- *x* – the x-coordinate
- *y* – the y-coordinate
- *modifiers* – the state of the modifier keys

mouseDrag(int, int, int)

Syntax

```
public void mouseDrag(int x, int y, int modifiers)
```

Category

Deprecated method

Description

Replaced by mouseDragged(int, int, int).

Sends a `MOUSE_DRAG` event.

Parameters

- *x* – the x-coordinate
- *y* – the y-coordinate
- *modifiers* – the state of the modifier keys

mouseEnter(int, int)

Syntax

```
public void mouseEnter(int x, int y)
```

Category

Deprecated method

Description

Replaced by mouseEntered(int, int).

Sends a `MOUSE_ENTER` event.

Parameters

- *x* – the x-coordinate
- *y* – the y-coordinate

mouseExit(int, int)

Syntax

```
public void mouseExit(int x, int y)
```

Category

Deprecated method

Description

Replaced by mouseExited(int, int).

Sends a `MOUSE_EXIT` event.

Parameters

- *x* – the x-coordinate
- *y* – the y-coordinate

mouseMove(int, int, int)

Syntax

```
public void mouseMove(int x, int y, int modifiers)
```

Category

Deprecated method

Description

Replaced by mouseMoved(int, int).

Sends a `MOUSE_MOVE` event.

Parameters

- *x* – the x-coordinate
- *y* – the y-coordinate
- *modifiers* – the state of the modifier keys

mouseUp(int, int, int)

Syntax

```
public void mouseUp(int x, int y, int modifiers)
```

Category

Deprecated method

Description

Replaced by mouseReleased(int, int, int).

Sends a `MOUSE_UP` event.

Parameters

- *x* – the x-coordinate
- *y* – the y-coordinate
- *modifiers* – the state of the modifier keys

JSMenuComponent

The `suntest.javastar.lib.JSMenuComponent` interface is constructed with one or more `java.awt.MenuComponent` objects, allowing simple tests of its contents. As an interface, you cannot instantiate `JSMenuComponent`, but you can obtain a `JSMenuComponent` and `JSComponent.menubar()` using `JS.mlookup()`.

This section describes `JSMenuComponent` by:

- Syntax
- Methods

Class

```
suntest.javastar.lib.JSMenuComponent
```

Syntax

```
public interface JSMenuComponent
extends Object
```

Methods

This section provides two ways to view methods. You can use:

- Methods by Name
- Methods by Category

If you are viewing this documentation on-line, you can click on any method name to jump to the full description, syntax, and parameter specification.

Methods by Name

Table 2-5 `JSMenuComponent` methods by name

Method	Category	Description
action()	Send event methods	Sends an action event referencing this MenuItem.
deselect()	Send event methods	Sets the state of a CheckboxMenuItem to false (sends a `DESELECTED` ItemEvent).
getAll()	Query methods	Returns all MenuComponents pointed to by this `JSMenuComponent`.
getUnique()	Query methods	Returns the MenuComponent pointed to by this `JSMenuComponent`. This requires that there is exactly one match.
getValidUnique()	Query methods	Returns the only MenuComponent pointed to by this `JSMenuComponent` which is capable of receiving events.
isUnique()	Query methods	Returns true if there is exactly one JSMenuComponent contained by this.
isValidUnique()	Query methods	Returns `true` if there is one valid match that can receive events.
item(String)	Find methods	Returns the MenuItem(s) whose label matches label.

Table 2-5 `JSMenuComponent` methods by name

Method	Category	Description
item(int, String)	Find methods	Returns the specified MenuItem, requiring its label to match label.
menu(String)	Find methods	Returns the menu(s) whose label matches the String you provide.
menu(int, String)	Find methods	Returns the specified menu, requiring its label match the String you provide.
mlookup(String)	Find methods	Find contained Named JSMenuComponents whose name matches the String you provide.
nested(String)	Find methods	Returns the nested Menu(s) whose label matches the String you provide.
nested(int, String)	Find methods	Returns the specified nested Menu, requiring its label to match the String you provide.
select()	Send event methods	Sets the state of a CheckboxMenuItem to true (sends a `SELECTED` ItemEvent).

Methods by Category

Table 2-6 `JSMenuComponent` methods by category

Category	Method	Description
Find methods Find methods return menus, `JSMenuComponents`, and menu items that match the criteria you specify.	item(String)	Returns the MenuItem(s) whose label matches label.
	item(int, String)	Returns the specified MenuItem, requiring its label to match label.
	menu(String)	Returns the menu(s) whose label matches the String you provide.
	menu(int, String)	Returns the specified menu, requiring its label match the String you provide.
	mlookup(String)	Find contained Named `JSMenuComponents` whose name matches the String you provide.
	nested(String)	Returns the nested Menu(s) whose label matches the String you provide.
	nested(int, String)	Returns the specified nested Menu, requiring its label to match the String you provide.

Table 2-6 `JSMenuComponent` methods by category

Category	Method	Description
Query methods	getAll()	Returns all `MenuComponents` pointed to by this `JSMenuComponent`.
These methods query the state of a `JSMenuComponent`, receiving either Components pointed to by the `JSMenuComponent`, or a boolean response.	getUnique()	Returns the `MenuComponent` pointed to by this `JSMenuComponent`. This requires that there is exactly one match.
	getValidUnique()	Returns the only `MenuComponent` pointed to by this `JSMenuComponent` which is capable of receiving events.
	isUnique()	Returns true if there is exactly one `JSMenuComponent` contained by this.
	isValidUnique()	Returns `true` if there is one valid match that can receive events.
Send event methods	action()	Sends an action event referencing this MenuItem.
These methods send events to GUI components.	deselect()	Sets the state of a CheckboxMenuItem to false (sends a DESELECTED ItemEvent).
	select()	Sets the state of a CheckboxMenuItem to true (sends a `SELECTED` ItemEvent).

action()

Syntax

```
public void action()
```

Category

Send event method

Description

Sends an action event referencing this MenuItem. Only applicable when the contained MenuComponent is a MenuItem.

deselect()

Syntax

```
public void deselect()
```

Category

Send event method

Description

Sets the state of a CheckboxMenuItem to false (sends a `DESELECTED` ItemEvent). Only Applicable when contained MenuComponent is CheckboxMenuItem.

getAll()

Syntax

```
public MenuComponent[] getAll()
```

Category

Query method

Description

Returns all MenuComponents pointed to by this `JSMenuComponent`.

getUnique()

Syntax

```
public MenuComponent getUnique()
```

Category

Query method

Description

Returns the MenuComponent pointed to by this `JSMenuComponent`. This requires that there is exactly one match, and throws an exception if there is not.

getValidUnique()

Syntax

```
public MenuComponent getValidUnique()
```

Category

Query method

Description

Returns the only MenuComponent pointed to by this `JSMenuComponent` which is capable of receiving events. If there is no component capable of receiving, or if more than one component can receive, this method throws an exception.

isUnique()

Syntax

```
public boolean isUnique()
```

Category

Query method

Description

Returns `true` if there is exactly one MenuComponent contained by this.

isValidUnique()

Syntax

```
public boolean isValidUnique()
```

Category

Query method

Description

Returns `true` if there is one valid match that can receive events.

item(String)

Syntax

```
public JSMenuComponent item(String label)
```

Category

Find method

Description

Returns the MenuItem(s) whose label matches label. Only applicable when contained MenuComponent is Menu. Note that this function will not return nested Menus—see nested(String).

Parameters

- *position* – Position in Menu
- *label* – Label of MenuItem

item(int, String)

Syntax

```
public JSMenuComponent item(int position, String label)
```

Category

Find method

Description

Returns the specified MenuItem, requiring its label to match label. Only applicable when contained MenuComponent is Menu. Note that this function will not return nested Menus; see nested(int, String).

Parameter

- *position* – Position in Menu
- *label* – Label of MenuItem

menu(String)

Syntax

```
public JSMenuComponent menu(String label)
```

Category

Find method

Description

Returns the menu(s) whose label match label. Only applicable when contained MenuComponent is MenuBar.

Parameters

- *label* – Label of Menu

menu(int, String)

Syntax

```
public JSMenuComponent menu(int position, String label)
```

Category

Find method

Description

Returns the specified menu, requiring its label match label. Only applicable when contained MenuComponent is MenuBar.

Parameters

- *position* – Position in MenuBar
- *label* – Label of Menu

mlookup(String)

Syntax

```
public JSMenuComponent mlookup(String name)
```

Category

Find method

Description

Find contained Named MenuComponents whose name matches name.

nested(String)

Syntax

```
public JSMenuComponent nested(String label)
```

Category

Find method

Description

Returns the nested Menu(s) whose label matches label. Only applicable when contained MenuComponent is Menu.

Parameters

- *position* – Position in Menu
- *label* – Label of Menu

nested(int, String)

Syntax

```
public JSMenuComponent nested(int position, String label)
```

Category

Find method

Description

Returns the specified nested Menu, requiring its label to match label. Only applicable when contained MenuComponent is Menu.

Parameters

- *position* – Position in Menu
- *label* – Label of Menu

select()

Syntax

```
public void select()
```

Category

Send event method

Description

Sets the state of a CheckboxMenuItem to true (sends a SELECTED ItemEvent). Only applicable when contained MenuComponent is CheckboxMenuItem.

Non-Component Classes 3

The Non-component classes of the JavaStar library provide support for Java toolkits that do not extend `java.awt.Component` to create components.

- JSNonComponentLocator
  Use this class to implement a locator for your toolkit.
- JSNCLData
  Objects of this class contain the data structure of a non-component.
- JSNonComponent
  This class contains non-component versions of some of the methods offered by `JSComponent`—for example, sending mouse events, performing simple method and data member comparisons, and querying object state.

JavaStar already provides locators for the Marimba™ Bongo™ toolkit and Netscape Internet Foundation Classes. If you are using either of these, you only need to refer to the `JSNonComponent` class, and `JSNCLData`, as needed.

JSNonComponentLocator

`JSNonComponentLocator` is a class that you implement to identify non-component objects inside Components. You only need to implement this class if you are building your user interface using a toolkit that is not based on the Java AWT model (and the components do not derive from `java.awt.Component`).

JavaStar already provides locators for the Bongo™ toolkit (by Marimba™) and the Netscape Internet Foundation Classes. You can use the source for these locators as an example of how to implement this class—look in the `\javastar\contrib\locators` directory.

> ***Note –*** To read more about JavaStar locators, see the chapter "Locators for Non-Components" in the *JavaStar User's Manual*.

Class

```
suntest.javastar.lib.JSNonComponentLocator
```

Syntax

```
public interface JSNonComponentLocator
```

Methods

Table 3-1 `suntest.javastar.lib.JSNonComponentLocator` Methods

Method	Description
findObject(Component, AWTevent)	Called during recording or inspection—translates a Component and `AWTevent` into a `JSNCLData` object containing locator information for a non-component.
getNamedObjectData(Component, String)	Called during playback—translates a string from the test code into a `JSNCLData` object that includes the screen coordinates where JavaStar can execute the event specified by the test.

findObject(Component, AWTevent)

Syntax

```
public JSNCLData findObject(Component c, AWTEvent e)
```

Description

Called during recording or inspection—when sent a component location and an `AWTevent`, this method returns a `JSNCLData` object containing a locator string to serve as the name for the non-component. If `findObject()` returns null, this indicates that the locator does not know of any non-component appropriate for this event.

Note – If all locators return null, the event will be recorded on the component itself.

Parameters

- *c* – the screen location of the non-component to be located
- *e* – the `AWTEvent` to send to the Component

getNamedObjectData(Component, String)

Syntax

```
public JSNCLData getNamedObjectData(Component c, String wname)
```

Description

When passed a Component and a non-component name (*wname*), this method returns a `JSNCLData` object containing the screen location of the component. If `getNamedObjectData()` returns `null`, this method should throw `GUINotFoundException` or `AmbiguousGUIException`, as appropriate.

Parameters

- *c* – Component that needs to be located.
- *wname* – the string that defines the non-component name

JSNCLData

An immutable data structure returned by `JSNonComponentLocator.findObject()`. The methods of the `NonComponentLocator` class return `JSNCLData` objects.

Class

```
suntest.javastar.lib.JSNCLData
```

Inheritance

```
java.lang.Object
➥ suntest.javastar.lib.JSNCLData
```

Syntax

```
public final class JSNCLData
extends Object
```

Variables

Table 3-2 Data members of `JSNCLData`

Variable	Declaration	Description
Name	`public final String` *Name*	The non-component's name
P	`public final Point` *P*	The reference point of the noncomponent (usually the upper-left corner) in the containing Component's coordinate system.
Ref	`public final Object` *Ref*	An object representing the noncomponent, possibly null, which will be used by `JSNonComponent.verifyAnyMethod()` method

JSNonComponent

Similar to the `JSComponent` interface. Objects of this type represent some sub-part of a Component of the user's GUI and allow sending only mouse events. The mouse events are the same as for `JSComponent`, except they are adjusted by the offfset given by the `JSNonComponentLocator` (`getOffset()`) then sent to the containing Component.

Class

```
suntest.javastar.lib.JSNonComponent
```

Syntax

```
public interface JSNonComponent
```

Methods

This section provides two ways to view methods. You can use:

- Methods by Name
- Methods by Category

If you are viewing this documentation on-line, you can click on any method name to jump to the full description, syntax, and parameter specification.

Methods by Name

Table 3-3 `JSNonComponent` methods by name

Method	Category	Description
getAnyField(String)	Comparison methods	Returns the value of the field name specified on a non-component object.
getAnyField(String)	Comparison methods	Returns the value of a method name specified on a non-component object.
getOffset()	Get methods	Returns the offset given by the locator of this `JSNonComponent`.
getReference()	Get methods	Returns the reference given by the locator of this `JSNonComponent`.
mouseClicked(int, int, int)	Send event methods	Equivalent of `JSComponent.mouseClicked`, but for a non-component.
mouseDragged(int, int, int)	Send event methods	Equivalent of `JSComponent.mouseDragged`, but for a non-component.
mouseMoved(int, int, int)	Send event methods	Equivalent of `JSComponent.mouseMoved`, but for a non-component.
mousePressed(int, int, int)	Send event methods	Equivalent of `JSComponent.mousePressed`, but for a non-component.
mouseReleased(int, int, int)	Send event methods	Equivalent of `JSComponent.mouseReleased`, but for a non-component.

Table 3-3 `JSNonComponent` methods by name

Method	Category	Description
multiClick(int, int, int, int)	Send event methods	Equivalent of `JSComponent.multiClick`, but for a non-component.
verifyAnyField(Script, Boolean, Boolean, String, Object, String)	Comparison methods	Reads the field `name()` on an object and compares it against expected value.
verifyAnyMethod(Script, Boolean, Boolean, String, Object, String)	Comparison methods	Calls the method `name()` on an object and compares against the expected value.

Methods by Category

Table 3-4 `JSNonComponent` methods by category

Category	Method	Description
Comparison methods	getAnyField(String)	Returns the value of the field name specified on a non-component object.
These methods compare the state of components.	getAnyMethod(String)	Returns the value of a method name specified on a non-component object.
	verifyAnyField(Script, Boolean, Boolean, String, Object, String)	Reads the field `name()` on an object and compares it against expected value.
	verifyAnyMethod(Script, Boolean, Boolean, String, Object, String)	Calls the method `name()` on an object and compares against the expected value.
Get methods	getOffset()	Returns the offset given by the locator of this `JSNonComponent`.
These methods read specific non-component values.	getReference()	Returns the reference given by the locator of this `JSNonComponent`.

Table 3-4 `JSNonComponent` methods by category

Category	Method	Description
Send event methods These methods send events to GUI components.	mouseClicked(int, int, int)	Equivalent of `JSComponent.mouseClicked`, but for a non-component.
	mouseDragged(int, int, int)	Equivalent of `JSComponent.mouseDragged`, but for a non-component.
	mouseMoved(int, int, int)	Equivalent of `JSComponent.mouseMoved`, but for a non-component.
	mousePressed(int, int, int)	Equivalent of `JSComponent.mousePressed`, but for a non-component.
	mouseReleased(int, int, int)	Equivalent of `JSComponent.mouseReleased`, but for a non-component.
	multiClick(int, int, int, int)	Equivalent of `JSComponent.multiClick`, but for a non-component.

verifyAnyField(Script, Boolean, Boolean, String, Object, String)

Syntax

```
public boolean verifyAnyField(Script sc, boolean sync, boolean
retry, String name, Object expected, String purpose)
```

Category

Comparison method

Description

Reads the field 'name' on the `getReference()` object, and compares it against expected value. Returns `true` or `false` depending on the comparison results.

Parameters

- *sc* – script for which to report results
- *sync* – if true, throw a SyncException on failure
- *retry* – if true, continue checking until timeout
- *name* – the name of the non-static field
- *expected* – the expected value of the method
- *purpose* – purpose of this verify

verifyAnyMethod(Script, Boolean, Boolean, String, Object, String)

Syntax

```
public boolean verifyAnyMethod(Script sc, boolean sync, boolean
retry, String name, Object expected, String purpose)
```

Category

Comparison method

Description

Calls the method `name()` on the `getReference()` object and compares against the expected value. Returns `true` or `false` depending on the comparison results.

Parameters

- *sc* – script for which to report results
- *sync* – if true, throw a SyncException on failure
- *retry* – if true, continue checking until timeout
- *name* – the name of the parameter-less method
- *expected* – the expected value of the method
- *purpose* – purpose of this verify

getOffset()

Syntax

```
public Point getOffset()
```

Category

Get method

Description

Returns the offset given by the locator of this `JSNonComponent`.

getReference()

Syntax

```
public Object getReference()
```

Category

Get method

Description

Returns the reference given by the locator of this `JSNonComponent`. Could be null.

getAnyField(String)

Syntax

```
public Object getAnyField(String name) throws
NoSuchFieldException, IllegalAccessException
```

Category

Comparison method

Description

Returns the value of field *name* on the `getReference()` Object.

Parameters

- *name* – the name of the non-static field

getAnyMethod(String)

Syntax

```
public Object getAnyMethod(String name) throws
NoSuchMethodException, IllegalAccessException,
InvocationTargetException
```

Category

Comparison method

Description

Returns the value of method `name()` on the `getReference()` Object.

Parameters

- *name* – the name of the parameter-less method

mouseClicked(int, int, int)

Syntax

```
public void mouseClicked(int x, int y, int modifiers)
```

Category

Send event method

Description

Equivalent of `JSComponent.mouseClicked(int, int, int)` for a `JSNonComponent`.

Parameters

- *x* - the x-coordinate (relatative to the `JSNonComponent`)
- *y* - the y-coordinate (relatative to the `JSNonComponent`)
- *modifiers* - the state of the modifier keys

mouseDragged(int, int, int)

Syntax

```
public void mouseDragged(int x, int y, int modifiers)
```

Category

Send event method

Description

Equivalent of `JSComponent.mouseDragged(int, int, int)` for a `JSNonComponent`.

Parameters

- *x* - the x-coordinate (relatative to the `JSNonComponent`)
- *y* - the y-coordinate (relatative to the `JSNonComponent`)
- *modifiers* - the state of the modifier keys

mouseMoved(int, int, int)

Syntax

```
public void mouseMoved(int x, int y, int modifiers)
```

Category

Send event method

Description

Equivalent of `JSComponent.mouseMoved(int, int)` for a `JSNonComponent`.

Parameters

- *x* - the x-coordinate (relatative to the `JSNonComponent`)
- *y* - the y-coordinate (relatative to the `JSNonComponent`)
- *modifiers* - the state of the modifier keys

mousePressed(int, int, int)

Syntax

```
public void mousePressed(int x, int y, int modifiers)
```

Category

Send event method

Description

Equivalent of `JSComponent.mousePressed(int, int, int)` for a `JSNonComponent`

Parameters

- *x* - the x-coordinate (relatative to the `JSNonComponent`)
- *y* - the y-coordinate (relatative to the `JSNonComponent`)
- *modifiers* - the state of the modifier keys

mouseReleased(int, int, int)

Syntax

```
public void mouseReleased(int x, int y, int modifiers)
```

Category

Send event method

Description

`JSComponent.mouseReleased(int, int, int)` for a `JSNonComponent.`

Parameters

- *x* - the x-coordinate (relatative to the `JSNonComponent`)
- *y* - the y-coordinate (relatative to the `JSNonComponent`)
- *modifiers* - the state of the modifier keys

multiClick(int, int, int, int)

Syntax

```
public void multiClick(int x, int y, int modifiers, int count)
```

Category

Send event method

Description

Equivalent of `JSComponent.multiClick(int, int, int, int)` for a `JSNonComponent`. Sends multiple groups of `MOUSE_PRESSED`, `MOUSE_RELEASED`, and `MOUSE_CLICKED` events. For example, to simulate a double-click, use `multiClick(x, y, 0, 2)`.

Parameters

- *x* - the x-coordinate (relatative to the `JSNonComponent`)
- *y* - the y-coordinate (relatative to the `JSNonComponent`)
- *modifiers* - the state of the modifier keys
- *count* – the number of clicks (one or more)

3

Text Map Class 4

`JSTextMapping` is a user-implemented interface for associating specific text to a Component, for record / playback purposes..

Topics:

Class

```
suntest.javastar.lib.JSTextMapping
```

Syntax

```
public interface JSTextMapping
extends Object
```

Methods

Table 4-1 suntest.javastar.lib.JSTextMapping Methods

Method	Description
computeText(Component)	The method that computes the association between the Component and a String.

computeText(Component)

Syntax

```
public String computeText(Component c)
```

Description

Implement this method to return the string you want to associate with the Component passed in. This is useful for things that behave like buttons (meaning, they have stable text). A text map is not usually helpful for mutable text components (such as TextComponents).

This method returns text to associate with c. If no text should be associated, it should return null.

Parameters

- c – the Component for which text should be computed

Timer Classes 5

Timer classes provide you with a way to define your own timers rather than inserting them using the Timers button in Record mode.

- JSTimer
  Defines a timer with methods to start and stop, retrieve start and elapsed times, and register a `JSTimerCallback`.
- JSTimerCallback
  Contains methods that are called asynchronously when a `JSTimer` object starts or stops.

JSTimer

A class with static methods for dealing with named timers.

Class

```
suntest.javastar.lib.JSTimer
```

Inheritance

```
java.lang.Object
➥ suntest.javastar.lib.JSTimer
```

Syntax

```
public class JSTimer
extends Object
```

Methods

Table 5-1 suntest.javastar.lib.JSTimer Methods

Method	Description
getElapsedTime(String)	Returns the current elapsed time on the timer by the given name (represented by String).
getStartTime(String)	Returns the current start time on the timer by the given name (represented by String).
register(Object, Boolean)	Registers to receive callbacks.
start(String)	Starts a timer by the given name (represented by String).
stop(String)	Stops a timer by the given name (represented by String).

getElapsedTime(String)

Syntax

```
public static long getElapsedTime(String name)
```

Description

Returns the current elapsed time on the timer by the given *name*. If the timer is not already started, then -1 will be returned.

Parameters

- *name* – a string to uniquely identify the timer

getStartTime(String)

Syntax

```
public static long getStartTime(String name)
```

Description

Returns the current start time on the timer by the given *name*. If the timer is not already started, then -1 will be returned. Start time is the value of `System.currentTimeMillis()` at the last call to `JSTimer.start(name)`.

Parameters

- *name* – a string to uniquely identify the timer

register(Object, Boolean)

Syntax

```
public static void register(JSTimerCallback object, boolean negatives)
```

Description

Registers to receive callbacks.

Parameters

- *object* – object to call
- *negatives* – object wishes to receive calls even when elapsed is -1

start(String)

Syntax

```
public static long start(String name)
```

Description

Starts a timer by the given *name*. Normally returns -1. If the timer is already started, then it will be stopped and restarted, and the elapsed time will be returned (and reported).

Parameters

- *name* – a string to uniquely identify the timer

stop(String)

Syntax

```
public static long stop(String name)
```

Description

Stops a timer by the given *name*. Normally returns (and reports) the elapsed time (in ms) since it was started. If the timer is not already started, then -1 will be returned.

Parameters

- *name* – a string to uniquely identify the timer

JSTimerCallback

Use `JSTimerCallback` to receive callbacks after `JSTimer.start()` and `JSTimer.stop()` methods execute. Objects of this must be registered with the `JSTimer.register()` method in order to function.

Class

```
suntest.javastar.lib.JSTimerCallback
```

Syntax

```
public interface JSTimerCallback
```

Methods

Table 5-2 suntest.javastar.lib.JSTimerCallback Methods

Method	Description
timerStarted(String, long, long)	This method is called asynchronously after a call to `JSTimer.start()`.
timerStopped(String, long, long)	This method is called asynchronously after a call to `JSTimer.stop()`.

timerStarted(String, long, long)

Syntax

```
public void timerStarted(String name, long timeStarted, long elapsed)
```

Description

Once you register JSTimerCallback with an `JSTimer.register()`, this method is called asynchronously after a call to `JSTimer.start()`.

Parameters

- *name* – the name of the timer (argument to `JSTimer.start()`).
- *timeStarted* – the time at which the call occurred.
- *elapsed* – the return value of `JSTimer.start()`.

timerStopped(String, long, long)

Syntax

```
public void timerStopped(String name, long timeStopped, long elapsed)
```

Description

Once you register JSTimerCallback with an `JSTimer.register()`, this method is called asynchronously after a call to `JSTimer.stop()`.

Parameters

- *name* – the name of the timer (argument to `JSTimer.stop()`).
- *timeStopped* – the time at which the call occurred.
- *elapsed* – the return value of `JSTimer.stop()`.

Script 6

JavaStar uses this class to define the structure for test scripts. When you record a test, JavaStar automatically creates a `Script` object, implementing each method according to the choices you make while recording.

This class is included for educational purposes. While you can generate tests manually, you should only modify the contents of the `play()` method within a script. Modifying other methods can lead to unpredictable results.

Topics:

Class

```
suntest.javastar.lib.Script
```

Inheritance

```
java.lang.object
➥ suntest.javastar.lib.Script
```

Syntax

```
public class Script
extends Object
```

Methods

Table 6-1 suntest.javastar.lib.Script Methods

Method	Description
getAppArgs()	Returns arguments with which the app will be started. For internal use only.
getAppClass()	Returns the full class name of the app main For internal use only.
gold()	Returns the gold directory For internal use only.
play(String[])	The body of this method is created by a record or hand-written to exercise the GUI being tested.
run()	Starts application For internal use only.
setGold(String)	Sets the gold directory. For internal use only.

getAppArgs()

Syntax

```
public String[] getAppArgs()
```

Description

Returns arguments with which the app will be started. For internal use only.

getAppClass()

Syntax

```
public String getAppClass()
```

Description

Returns the full class name of the app main. For internal use only.

gold()

Syntax

```
public String gold()
```

Description

Returns the gold directory. For internal use only.

play(String[])

Syntax

```
public void play(String args[]) throws Throwable
```

Description

The body of this method is created by a record or hand-written to exercise the GUI being tested. Failure is indicated by a `Throwable` being thrown.

run()

Syntax

```
public void run()
```

Description

Starts application. For internal use only.

setGold(String)

Syntax

```
public void setGold(String g)
```

Description

Sets the gold directory. For internal use only.

Error Classes 7

This section describes the error classes of JavaStar:

- BadRegularExpressionError
- JavaStarInternalError
- NoAppResponseError

BadRegularExpressionError

Indicates that a regular expression given to a JavaStar API method is illegal. See the chapter "Syntax for Regular Expressions" for details on proper syntax.

Inheritance

```
java.lang.Object
➥ java.lang.Throwable
   ➥ java.lang.Error
      ➥ suntest.javastar.lib.NoAppResponseError
```

Syntax

```
public class BadRegularExpressionError
extends Error
```

JavaStarInternalError

Indicates a problem occurred within JavaStar. Note what was happening before the `JavaStarInternalError` occurred and contact SunTest with this information.

Class

```
suntest.javastar.lib.JavaStarInternalError
```

Inheritance

```
java.lang.object
➥ java.lang.Throwable
   ➥ java.lang.Error
      ➥ suntest.javastar.lib.JavaStarInternalError
```

Syntax

```
public class JavaStarInternalError
extends Error
```

Methods

printStackTrace(PrintStream)

Syntax

```
public void printStackTrace(PrintStream ps)
```

Description

Overrides `printStackTrace` in class `Throwable`.

Parameters

- *ps* – print device

NoAppResponseError

Indicates JavaStar is terminating a thread because it reached the user-set timeout limit between JavaStar events. This error references the `hangtime` value you define at the command line or within the JavaStar Options GUI.

Class

```
suntest.javastar.lib.NoAppResponseError
```

Inheritance

```
java.lang.object
➥ java.lang.Throwable
   ➥ java.lang.Error
      ➥ suntest.javastar.lib.NoAppResponseError
```

Syntax

```
public class NoAppResponseError
extends RuntimeException
```

Exception Classes 8

This chapter describes the exception classes for JavaStar:

AmbiguousGUIException

Indicates that the `JSComponent` contains multiple Components capable of receiving events. For example, if there are two buttons on a frame "X" with the same label "Quit", the following code will produce an `AmbiguousGUIException`:

```
JS.frame("X").button("Quit").action();
```

Class

```
suntest.javastar.lib.AmbiguousGUIException
```

Inheritance

```
java.lang.Object
➥ java.lang.Throwable
   ➥ java.lang.Exception
      ➥ java.lang.RuntimeException
         ➥ suntest.javastar.lib.AmbiguousGUIException
```

Syntax

```
public class AmbiguousGUIException
extends RuntimeException
```

GUINotAcceptingException

Indicates that the JSComponent does not contain any Component capable of receiving events. This can occur when a test attempts an action when all the components are disabled or are not visible.

Class

```
suntest.javastar.lib.GUINotAcceptingException
```

Inheritance

```
java.lang.Object
➥ java.lang.Throwable
   ➥ java.lang.Exception
      ➥ java.lang.RuntimeException
         ➥ suntest.javastar.lib.GUINotAcceptingException
```

Syntax

```
public class GUINotAcceptingException
extends RuntimeException
```

GUINotFoundException

Indicates that the JSComponent could not be resolved—no Components match its description.

Class

```
suntest.javastar.lib.GUINotFoundException
```

Inheritance

```
java.lang.Object
➥ java.lang.Throwable
   ➥ java.lang.Exception
      ➥ java.lang.RuntimeException
         ➥ suntest.javastar.lib.GUINotFoundException
```

Syntax

```
public class GUINotFoundException
extends RuntimeException
```

GUITypeException

Indicates that the Component of the `JSComponent` is of the wrong type for special operations. For example, if a test attempts text entry on a Button field, the test throws this exception.

This can occur when your test references a component of your application or applet by name, and later a component of a different type is changed to use this name instead.

Class

```
suntest.javastar.lib.GUITypeException
```

Inheritance

```
java.lang.Object
➥ java.lang.Throwable
   ➥ java.lang.Exception
      ➥ java.lang.RuntimeException
         ➥ suntest.javastar.lib.GUITypeException
```

Syntax

```
public class GUITypeException
extends RuntimeException
```

SyncException

Indicates a that a synchronized comparison did not achieve a match before the timeout.

Class

```
suntest.javastar.lib.GUITypeException
```

Inheritance

```
java.lang.Object
➥ java.lang.Throwable
   ➥ java.lang.Exception
      ➥ java.lang.RuntimeException
         ➥ suntest.javastar.lib.SyncException
```

Syntax

```
public class SyncException
extends RuntimeException
```

Waiting—Custom Synchronization 9

`Waiting` is a user-implemented interface that `JS.waitFor()` uses to provide custom synchronization. Implement this class to define the condition you want `JS.waitFor()` to synchronize to.

Topics:

Class

```
suntest.javastar.lib.Waiting
```

Syntax

```
public interface Waiting
extends Object
```

Methods

Table 9-1 suntest.javastar.lib.Waiting Methods

Method	Description
getState()	Returns true if the condition being waited for is now happening.

getState()

Syntax

```
public abstract boolean getState()
```

Description

Implement this function for the condition you want `JS.waitFor()` to synchronize to. Return `true` when the condition being waited for occurs.

Syntax for Regular Expressions 10

This release of JavaStar uses a different syntax for regular expressions—one that is perhaps more familiar to developers. The older syntax is still supported, but as it will be phased out in a future release, methods that support it are in a deprecated status.

Topics:

Current Syntax

The new regular expression language is used with JavaStar API methods that have names end with `RE`. For example, `JSComponent.hasMemberRE()` and `JS.dialogRE()` both use this syntax.

This language is a subset of Perl's regular expressions, similar to UNIX *egrep*.

Alternatives

A regular expression can consist of one or more alternatives. An *alternative* is a sequence of items. Alternatives are separated by `|` (the pipe symbol). An alternative matches if all the items match in the order they occur. The regular expression matches if any of the alternatives match.

Items

An *item* is either an assertion or a quantified atom. *Assertions* are:

Table 10-1 Assertions in regular expressions

Expression	Use
`\b`	Matches on word boundary, between `\w` and `\W` or between `\W` and `\w`.
`\B`	Matches on non-word boundary.

A *quantified atom* is an atom followed optionally by one of the following which indicate how many times the atom must or may occur. If no quantifier is given, the atom must occur exactly once.

Quantifiers are:

Table 10-2 Quantifiers in regular expressions

Expression	Use
*	0 or more times
+	1 or more times
?	0 or 1 time

An atom is:

- A regular expression in parentheses ( )
- A `.` matches any character except `\n`
- A list of characters in square brackets `[ ]` matches one of a class of characters. A caret `^` at the front of the list negates the class. Character ranges may be indicated with a hyphen `-` , such as `a-z`. You may also use any of `\d \w \s \n \r \t \f` . Backslash can be used to escape `- ] ^` to prevent their special meaning.

- Some backslashed letters have special meanings:

Table 10-3 Backslashed characters in regular expressions

Expression	Use
`\n`	Newline
`\r`	Carriage return
`\t`	Tab
`\f`	Form feed
`\d`	Digit, same as [0-9]
`\D`	Non-digit, same as [^0-9]
`\w`	Word character (alphanumeric), same as [0-9A-Z_a-z]
`\W`	Non-word charcter, same as [^0-9A-Z_a-z]
`\s`	Whitespace character, same as [\t\n\r\f]
`\S`	Non-whitespace character, same as [^ \t\n\r\f]

- Any other backslashed character matches that character.
- Any other character matches itself

Excluded Syntax

Regular expression syntax *not* included are:

Table 10-4 Excluded Perl regular expressions

Character	Use
`^` and `$`	These are not useful, since the library always match whole strings.
`{n,m}` `{n,}` `{n}`	These are not currently implemented.
`\1` , etc. (backreferences to substrings.)	Not implemented
`\0` `\033` `\x7f` `\cD`	These are not necessary. JavaStar Strings can contain any legal Java characters that do not have special meaning to the regular expression language, as well as those if prefixed by backslash. You can use Java unicode escapes `\u0000` - `\uFFFF`, interpreted by Java. That is "`\u0000`" not "`\\u0000`".

Previous Syntax

The RX methods of JS and JSComponent (for example, `buttonRX()` and `dialogRX()`) use regular expressions. Regular expressions use a syntax in which a few characters are special constructs and the rest are "ordinary." This section describes this syntax and how you can perform complex matches to complex criteria.

Table 10-5 Syntax for regular expressions

Term	Definition
Ordinary characters	A simple regular expression which matches that character and nothing else—for example, a simple alphabetical character such as `f` matches only itself and does not match any other characters or strings.
Special characters	`{`, `}`, `~`, `*`, `+`, `?`, `,`, `-`, `[`, `]`, and `\`. `\` is defined only as an escape character, to allow the use of special characters as literals.

Concatenating Regular Expressions

When you concatenate two regular expressions—for example, *A* and *B*—the result is a regular expression that matches a string if *A* matches some amount of the beginning of that string and *B* matches the rest of the string.

In a simple example, you can concatenate the regular expressions `a` and `t` to get the regular expression `at`. This expression only matches the string `at`.

To do something more complex, you need to use special characters, as this table shows:

Table 10-6 Special character use for string concatenation

Character	Use
`[ ... ]`	'`[`' begins a "character set", which is terminated by a '`]`'. In the simplest case, the characters between the two form the set. Thus, '`[ad]`' matches either one '`a`' or one '`d`', and '`[ad]*`' matches any string composed of just '`a`'s and '`d`'s (including the empty string). The expression '`c[ad]*r`' matches '`cr`', '`car`', '`cdr`', '`caddaar`', etc. You can include character ranges in a character set by writing two characters with a '`-`' between them. Thus, '`[a-z]`' matches any lower-case letter. Ranges may be intermixed freely with individual characters, as in '`[a-z$%.]`', which matches any lower- case letter or '`$`', '`%`', or period.
`~[ ... ]`	'`~[`' begins a "complement character set", which matches any character except the ones specified. Thus, '`~[a-z0-9A-Z]`' matches any character except letters and digits.
`{ ... }`	A grouping construct that serves two purposes: • To enclose a set of ',' alternatives. Thus, '`{foo,bar}x`' matches either '`foox`' or '`barx`'. • To enclose a complicated expression for the postfix '`*`', '`?`', or '`+`' to operate on. Thus, '`ba{na}*`' matches '`bananana`', etc., with any (zero or more) number of '`na`' strings.
*	'`*`' means any number (zero or more) of something. It may be used in three ways: • Standalone, to mean any number of characters. For example, '`ab*`' means any string starting with '`ab`' including '`ab`' itself. • Immediately following the '`]`' of a character set (regular or complemented). Thus, '`t~[e]*`' matches any string beginning with '`t`' and including no '`e`'s (such as '`t`', '`this`', or '`that`'). • Immediately following the '}' of a grouping. For example, '`{aa,cc}*`' would match '' '`aa`' '`cc`' '`aacc`' '`ccaa`' '`aaccaa`', etc.

Table 10-6 Special character use for string concatenation

Character	Use
+	'+' is just like '*', except that it requires one or more of something. It has the same three basic uses: • Standalone, to mean any number of characters, except the empty set. For example, '+ab+' means any string containing 'ab' except there must be at least one character before and after: 'cabs' or 'Alabama', but not 'cab' or 'abe'. • Immediately following the ']' of a character set (regular or complemented). Thus, '[a-z]+' matches any string containing only lower case letters. • Immediately following the '}' of a grouping. For example, '{aa,cc}+' would match 'aa' 'cc' 'aacc' 'ccaa' 'aaccaa', etc., but not the empty string.
?	'?' is like '+' and '*', except that it normally requires zero or one of something. It has the same three basic uses: • Standalone, to mean any single character. For example, 'x?y' will match 'xay' or 'xvy' but not 'xy' or 'xaay'. • Immediately following the ']' of a character set (regular or complemented). Thus, '[a-z]?' matches either the empty string or any single lower case letter. • Immediately following the '}' of a grouping. For example, '{aa,cc}?' would match only 'aa','cc', or the empty string.
-	'-' is only allowed unescaped inside a character set.
,	',' is only allowed unescaped inside a grouping.
~	'~' is used to mean "not". It has two uses: • Preceding '[' as the start of a complement character set. • Preceding any other character as a match for any character other than that. For example, 'b~at' matches any three letter string starting with 'b' and ending in 't' except 'bat'. This includes 'b t' 'bit' 'bot' 'b%t'.

These regular expressions use '\' as an escape. Java also uses '\' as an escape and this can be somewhat confusing. To put a '\' into a Java String, the literal will appear as '\\'. For example, in the Java code:

```
String rx1 = "[\\\\\\+\\-]";
```

```
String rx2 = "[\t\r\n ]+";
```

`rx1` actually contains: '`[\\\+\-]`' which will match a '\','+' or '-'.

`rx2` actually contains: '`[]+`' and will match any whitespace.

More Examples

'`[\+\-/\*]`' matches any of the four standard operators '+', '*', '/', '-'. Note that the three special characters had to be prefaced with the escape character for this expression.

'`[A-Za-z][a-z]*`' is a minimum for a well-formed English word (other than acronyms) by requiring a letter followed by zero or more lower case letters.

'`{OK,Done,Quit,Cancel,Continue}`' matches any of five frequently used button labels.

'`[A-Za-z_$][A-Za-z_$0-9]*`' matches any legal Java identifier (in normal ASCII—Java also allows UniCode characters).