

### 8.3.1 Class diagrams

Various class diagrams have been organized according to the components to which they belong. The following section describes various classes in the following order:

- Browser
  - ♣ Network classes
  - ♣ Parser classes
  - ♣ Event Handler Classes
  - ♣ Caching classes
- Proxy server
  - ♣ iBOSClientHandler
  - ♣ iBOSPServer
  - ♣ cachingJobs
  - ♣ URLRequest

### **8.3.1.1 Browser Classes**

#### **8.3.1.1.1 Network Class**

The network class is an integral part of Network Interaction Module (NIM) which is deployed at the browser end of iBOS.

The following is the major functionality performed by the Network Class:

- Sends the request in form of streams. The request first goes to the cache manager. If any earlier instance of the same request is found in the cache, the page is fetched from the cache itself otherwise the request is sent to the Proxy server present on the LAN.
- Accepts the response either from the cache manager or the Proxy server.
- Submits user data to the Proxy server using forms which is then forwarded to the remote web server.

Network	
+ Fetch (Void)	String
+ getByName (Void )	Boolean
+ getByURL ( Void)	Boolean

**Figure 26 Network Class**

**CLASS NAME** : **Network**  
*Inherits* : None  
*Inherited by* : None

**MEMBER FUNCTIONS**

**Name** : **Fetch**  
Return Type : String  
Arguments : Void  
Visibility : Public  
Context : This method is used to fetch a file from the network, proxy server. This method calls either the `getByName` method, or the `getByURL` method.

**Name** : **getByName**  
Return Type : Boolean  
Arguments : Void  
Visibility : Public  
Context : This method fetches a file from the proxy server using the old and new URLs. This is required because some hyperlinks may include relative URLs.

**Name** : **getByURL**  
Return Type : Boolean  
Arguments : Void  
Visibility : Public  
Context : This method is used when a user opens up the url-entry window from the browser and inputs a new URL.

ServerIO	
- serverAddr	String
- portNum	Short
- sockfd	Integer
- serverAddress	sockaddr_in
+ success	Boolean
« constructor »	
+ ServerIO( )	
+ println (String)	Boolean
+ readLine (Void)	String

**Figure 27 Server I/O Class**

**CLASS NAME** : ServerIO  
*Inherits* : None  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : serverAddr  
**Type** : String  
**Visibility** : Private  
**Context** : This member stores the IP address of the proxy server.

**Name** : portNum  
**Type** : Short Integer  
**Visibility** : Private  
**Context** : This member stores the port number on which the proxy server listens for connections.

**Name** : sockfd  
**Type** : Integer  
**Visibility** : Private  
**Context** : This member stores a socket-descriptor as is required by Unix systems to write to any type of files. Even network I/O is deemed to be equivalent to file I/O.

**Name** : serverAddress  
**Type** : sockaddr\_in  
**Visibility** : Private  
**Context** : This member stores the IP address of the proxy server.

**Name** : success  
**Type** : Boolean  
**Visibility** : Public  
**Context** : This boolean member represents whether or not the I/O to/from the proxy server failed at any time during the processing.

#### **MEMBER FUNCTIONS**

**Name** : serverIO()  
**Return Type** : Not Applicable  
**Arguments** : None  
**Visibility** : Public  
**Context** : This is the constructor for the server I/O object. This basically attempts to fetch a socket descriptor for the connection to the proxy server.

<b>Name</b>	:	println
Return Type	:	Boolean
Arguments	:	String
Visibility	:	Public
Context	:	This method prints a single line of text along with a new-line onto the network.

<b>Name</b>	:	readLine
Return Type	:	String
Arguments	:	Void
Visibility	:	Public
Context	:	This method reads a single line of text from the network.

#### **8.3.1.1.2 Parser Classes**

A HTML parser is an important part of any web browser. The HTML parser parses the HTML file, recognizes various HTML tags and accordingly lays out the web page. The iBOS too consists of a HTML parser. The iBOS parser parses the HTML file in 3 steps :

- Recognize only active elements.
- Parses the file to layout tables.
- Layouts the file to a double buffer and then finally onto the screen.

In order to work in a proper manner the parser as a whole creates certain data structures some which are retained in the memory to be used later while others are destroyed.

The parser consists of the following classes:

- **Widget Classes** : Various widgets or active elements such as textbox, textarea, combo box, radio box etc have been each modeled as separate classes. All these classes generalize to component class.
- **Utility Classes** : Certain components which assist in parsing and are used again and again are referred to as Utility Classes.
- **Data Structure Classes** : This category of classes consists of a number of data structures such as arrays, stack, linked lists etc
- **Pass specific Classes.** : Certain category of classes are specific to the phase of the parsing. All such classes are clubbed under this category.



#### 8.3.1.1.2.1 Widgets Classes

Widgets or various active elements such as textbox, textarea, combo box, radio box, hyperlinks, etc have been modeled as classes. This type of modeling gives a greater degree of abstraction and encapsulation. All the functions needed to provided the functionality of a given widget is embedded in its class itself. This modeling helps with respect to handling various events associated with each widget.

All the widgets generalizes to component class as shown in the figure:

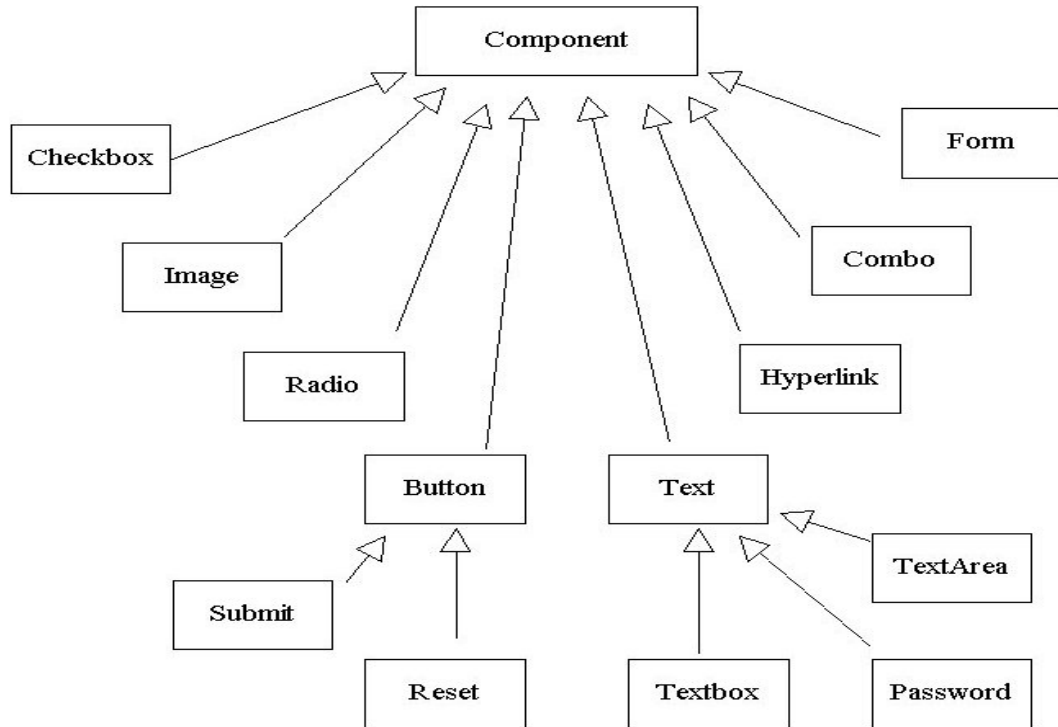


Figure 28 Hierarchy of widgets

Component	
# helpString	String
# getKey( <i>Integer</i> )	<i>Integer</i>
# setXY( <i>Integer,Integer</i> )	<i>Integer</i>
# select	<i>Void</i>
# unselect	<i>Void</i>
# displayed	<i>Void</i>

display() method is a  
method for analysis

**Figure 29 Component Class**

**CLASS NAME** : **Component**  
**Inherits** : None  
**Inherited by** : Checkbox, Radio, Button, Text, Hyperlink, Form, Image, and Combo

#### **DATA MEMBERS**

**Name** : **helpString**  
**Type** : String  
**Visibility** : Protected  
**Context** : Represents the string describing actions possible on that component.

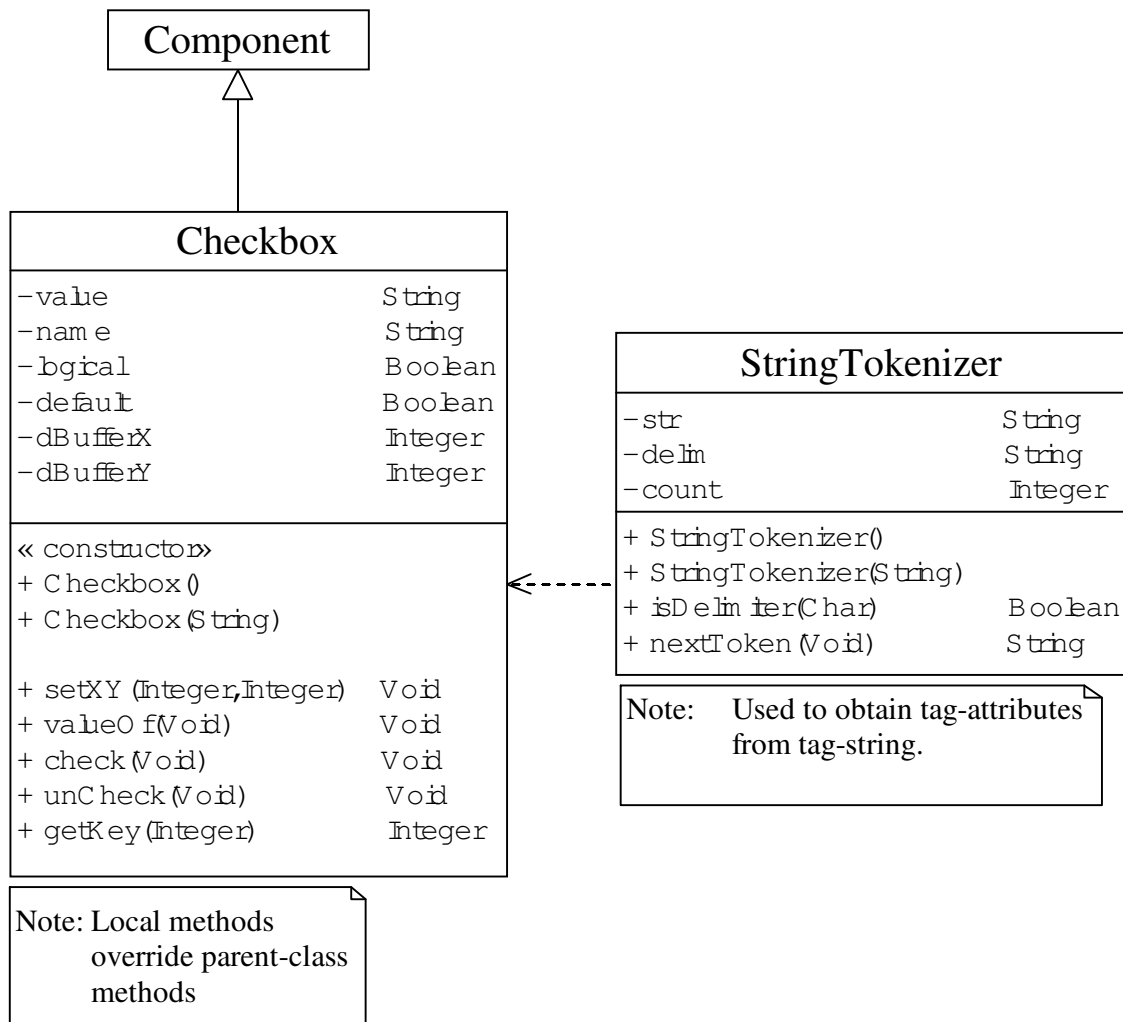
#### **MEMBER FUNCTIONS**

**Name** : **getKey**  
**Return Type** : Integer  
**Arguments** : Integer  
**Visibility** : Public (virtual)  
**Context** : Accepts a key and generates corresponding event. This function is required for the event-handler to use a uniform interface to all widget objects.

**Name** : **select**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public (virtual)  
**Context** : Selects the component for events to be trapped by it.

**Name** : **unselect**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public (virtual)  
**Context** : De-selects the component.

**Name** : **Display**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : Diagnostic method used for debugging the widgets.



**Figure 30 Checkbox Class**

**CLASS NAME** : **Checkbox**  
*Inherits* : Component  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **value**  
Type : String  
Visibility : Private  
Context : Stores the Boolean condition represented by this object.

**Name** : **name**  
Type : String  
Visibility : Private  
Context : Represents the name of the object itself.

**Name** : **logical**  
Type : Boolean  
Visibility : Private  
Context : Represents whether the condition is selected to be true or not.

**Name** : **def\_val**  
Type : Boolean  
Visibility : Private  
Context : Represents the default Boolean value for the condition represented by logical.

**Name** : **dBufferX**  
Type : Integer  
Visibility : Private  
Context : Represents the background-buffer column value

**Name** : **dBufferY**  
Type : Integer  
Visibility : Private  
Context : Represents the background-buffer row value

#### **MEMBER FUNCTIONS**

**Name** : **Checkbox**  
Return Type : Not Applicable  
Arguments : None  
Visibility : Public  
Context : Default constructor; sets all values to their trivial case.

**Name** : **Checkbox**  
**Return Type** : Not Applicable  
**Arguments** : String  
**Visibility** : Public  
**Context** : Accepts the input tag representing a checkbox and parses the attributes to be stored into the object.

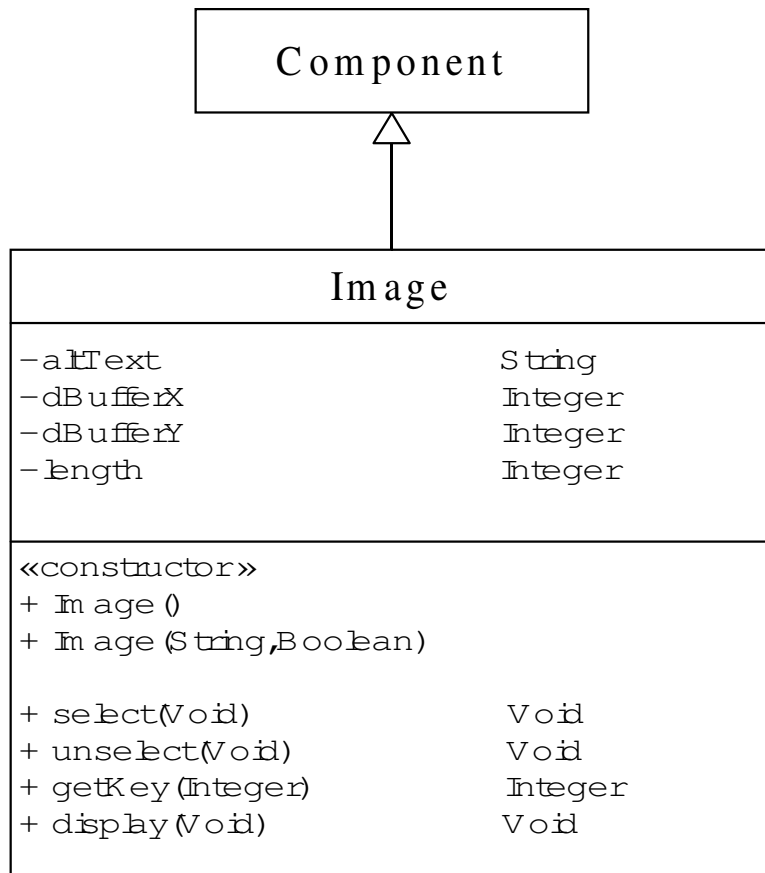
**Name** : **setXY**  
**Return Type** : Void  
**Arguments** : 1.) Integer  
2.) Integer  
**Visibility** : Public  
**Context** : Accepts the background buffer coordinates (column,row) to be stored in dBufferX and dBufferY.

**Name** : **valueOf**  
**Return Type** : String  
**Arguments** : None  
**Visibility** : Public  
**Context** : Returns the actual value represented by the checkbox.

**Name** : **check**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : Sets the logical value of the checkbox to true.

**Name** : **unCheck**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : Sets the logical value of the checkbox to false.

**Name** : **getKey**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : Captures keys and toggles the checkbox value.



**Figure 31 Image Class**

**CLASS NAME** : **Image**  
*Inherits* : Component  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **altText**  
Type : String  
Visibility : Private  
Context : Represents the text to be displayed in place of the image.

**Name** : **dBufferX**  
Type : Integer  
Visibility : Private  
Context : Represents the background-buffer column number where it is located.

**Name** : **dBufferY**  
Type : Integer  
Visibility : Private  
Context : Represents the background-buffer row number where it is located.

**Name** : **length**  
Type : Integer  
Visibility : Public  
Context : Represents the length of the textual representation of the image.

#### **MEMBER FUNCTIONS**

**Name** : **Image**  
Return Type : Not Applicable  
Arguments : None  
Visibility : Public  
Context : Is the default constructor for the class. Sets all attributes to null and zeroes.

**Name** : **Image**  
Return Type : Not Applicable  
Arguments : String  
Visibility : Public  
Context : Constructor that accepts the IMG SRC tag string and extracts the alternative text.

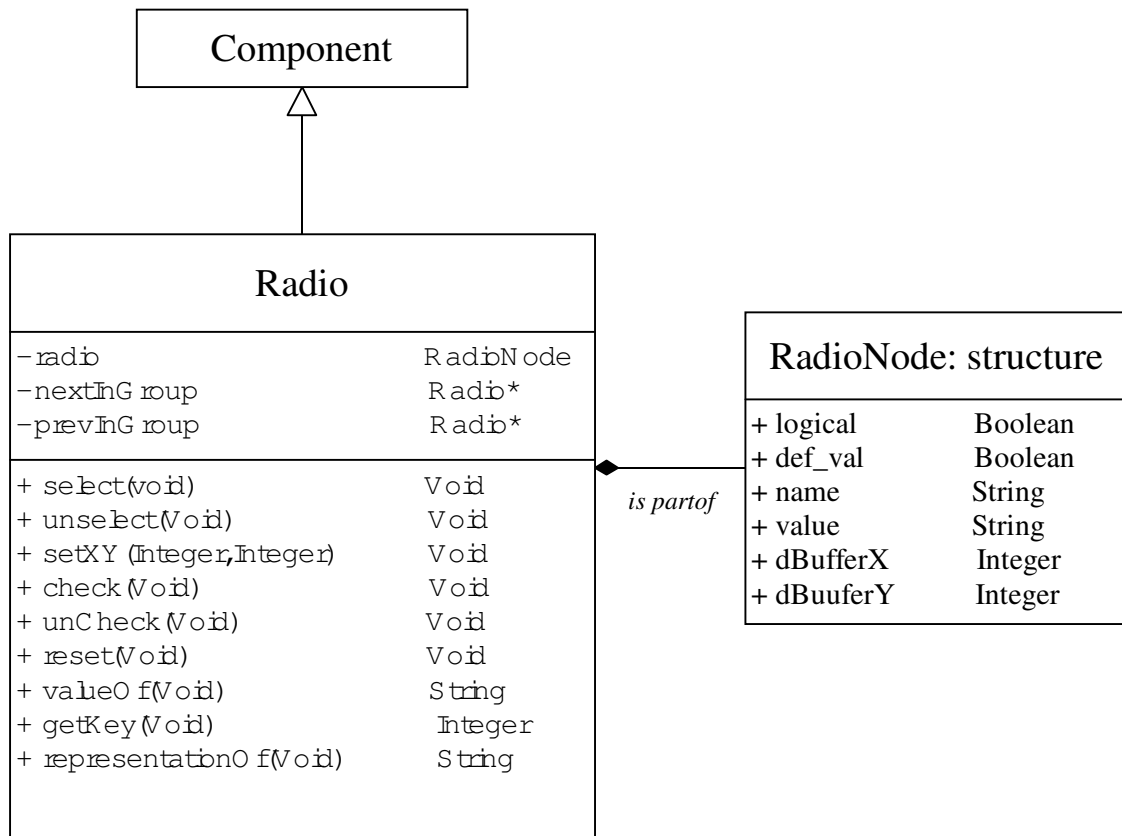


**Name** : **select**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : Selects the visible representation of the image tag in the background buffer to capture events.

**Name** : **unselect**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : De-select the visible representation of the tag.

**Name** : **getKey**  
**Return Type** : Integer  
**Arguments** : Integer  
**Visibility** : Public  
**Context** : Accepts key strokes and performs actions accordingly.

**Name** : **display**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : A debugging method – needs to be used for debugging the image widget.



**Figure 32 Radio Class**

**CLASS NAME** : **Radio**  
*Inherits* : Component  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **radio**  
**Type** : RadioNode  
{ Logical value, default logical value, name, value, dBufferX, dBufferY }

**Visibility** : Private  
**Context** : The node that represents this particular radio box.

**Name** : **nextInGroup**  
**Type** : Pointer of type Radio  
**Visibility** : Public  
**Context** : Refers to the next radio box that is in the same group as it is.

**Name** : **prevInGroup**  
**Type** : Pointer of type Radio  
**Visibility** : Public  
**Context** : Refers to the previous radio box that is in the same group as it is.

#### **MEMBER FUNCTIONS**

**Name** : **Radio**  
**Return Type** : Not Applicable  
**Arguments** : None  
**Visibility** : Public  
**Context** : Sets the attributes to default values

**Name** : **Radio**  
**Return Type** : Not Applicable  
**Arguments** : String  
**Visibility** : Public  
**Context** : Accepts the input tag representing the radio button and extracts all the attributes for storage within this object.

**Name** : **select**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : Selects this radio box for handling events.

**Name** : **unselect**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : De-selects the radio box. An unselected radio box does not handle events.

**Name** : **setXY**  
**Return Type** : Void  
**Arguments** : 1.) Integer  
2.) Integer  
**Visibility** : Public  
**Context** : Sets the background buffer coordinates for this radio box.

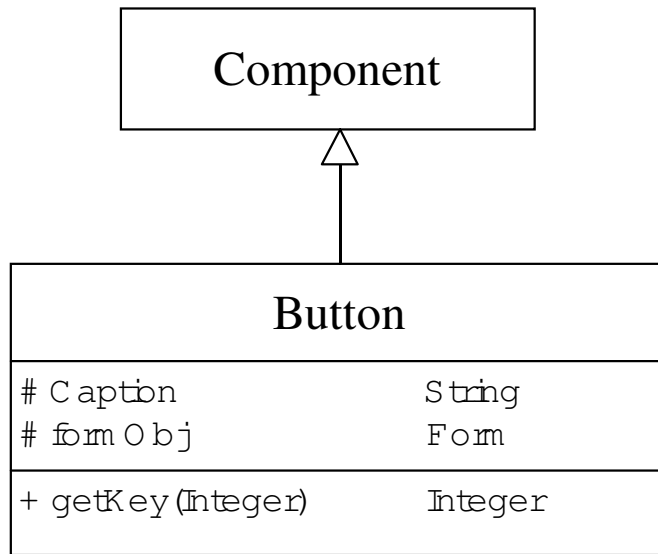
**Name** : **check**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : Sets the value of this radio box to true, and that of all others in the same group to false.

**Name** : **unCheck**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : It will uncheck only this radio box.

**Name** : **reset**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Sets the values to its default value.  
**Context** : Sets the default value of this radio box.

**Name** : **valueOf**  
**Return Type** : String  
**Arguments** : None  
**Visibility** : Public  
**Context** : Returns the value represented by this radio option.

**Name** : **getKey**  
**Return Type** : Integer  
**Arguments** : Integer  
**Visibility** : Public  
**Context** : Traps keys and generates corresponding events.



**Figure 33 Button Class**

***CLASS NAME*** : **Button**  
*Inherits* : Component  
*Inherited by* : Submit, Reset

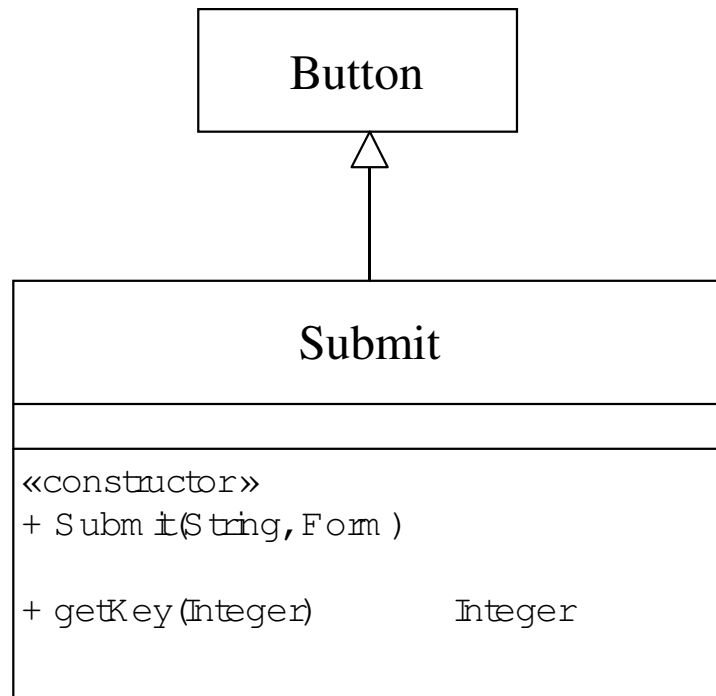
***DATA MEMBERS***

**Name** : **Caption**  
Type : String  
Visibility : Protected  
Context : The visible caption of the button. The button is actually modeled as a hyperlink and so only its caption appears.

**Name** : **formObj**  
Type : Form  
Visibility : Protected  
Context : Represents a reference to the form object that it belongs to.

***MEMBER FUNCTIONS***

**Name** : **getKey**  
Return Type : Integer  
Arguments : Integer  
Visibility : Public  
Context : Accepts key strokes and generates appropriate events.



**Figure 34 Submit Button Class**

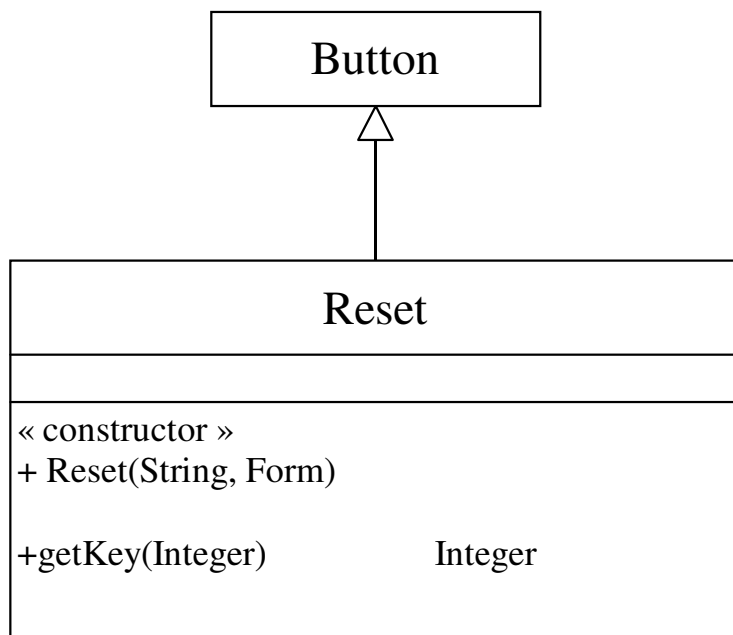
**CLASS NAME** : **Submit**  
*Inherits* : Button  
*Inherited by* : None

**MEMBER FUNCTIONS**

**Name** : **Submit**  
Return Type : Not Applicable  
Arguments : 1) String  
              2) Form  
Visibility : Public  
Context : Accepts the string representing the submit tag and the form  
          object to which it belongs.

**Name** : **getKey**  
Return Type : Integer  
Arguments : Integer  
Visibility : Public  
Context : Accepts keystrokes and generates corresponding events.





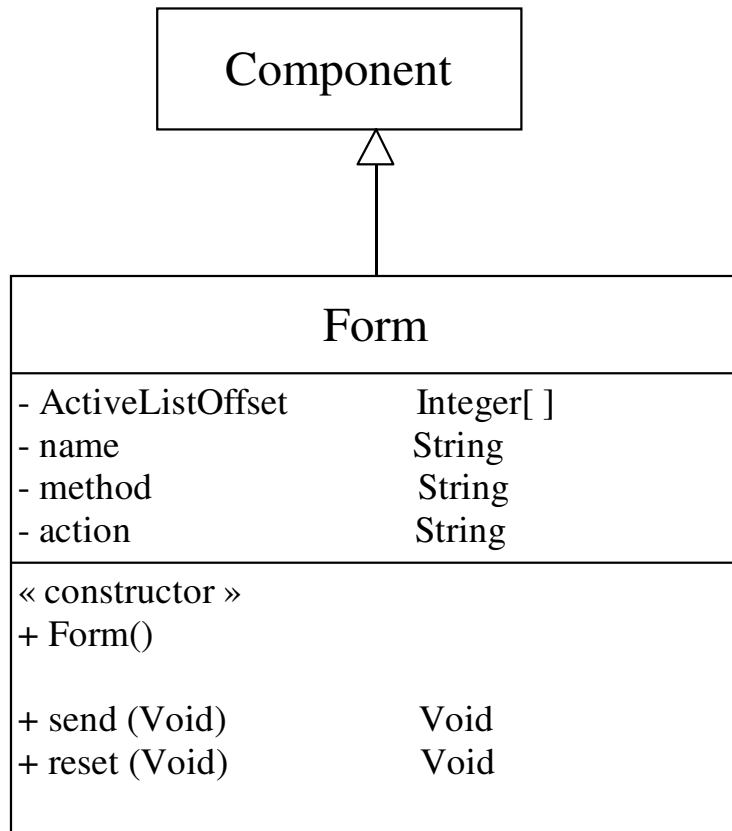
**Figure 35 Reset Button Class**

**CLASS NAME** : **Reset**  
*Inherits* : Button  
*Inherited by* : None

**MEMBER FUNCTIONS**

**Name** : Reset  
Return Type : Not Applicable  
Arguments : 1) String  
              2) Form  
Visibility : Public  
Context : Constructor accepts the input tag corresponding to the reset button and the form object's reference to which the reset button belongs.

**Name** : **getKey**  
Return Type : Integer  
Arguments : Integer  
Visibility : Public  
Context : Accepts keystrokes and generates corresponding events.



**Figure 36 Form Class**

**CLASS NAME** : **Form**  
*Inherits* : Component  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **ActiveListOffset**  
**Type** : Integer array  
**Visibility** : Private  
**Context** : Holds the offsets in the AST of those elements which form a part of this form.

**Name** : **name**  
**Type** : String  
**Visibility** : Private  
**Context** : Refers to the name of this form as a component.

**Name** : **method**  
**Type** : String  
**Visibility** : Private  
**Context** : Refers to the method (GET/POST) to use for sending data.

**Name** : **action**  
**Type** : String  
**Visibility** : Private  
**Context** : Refers to the server page accepting the data in the form.

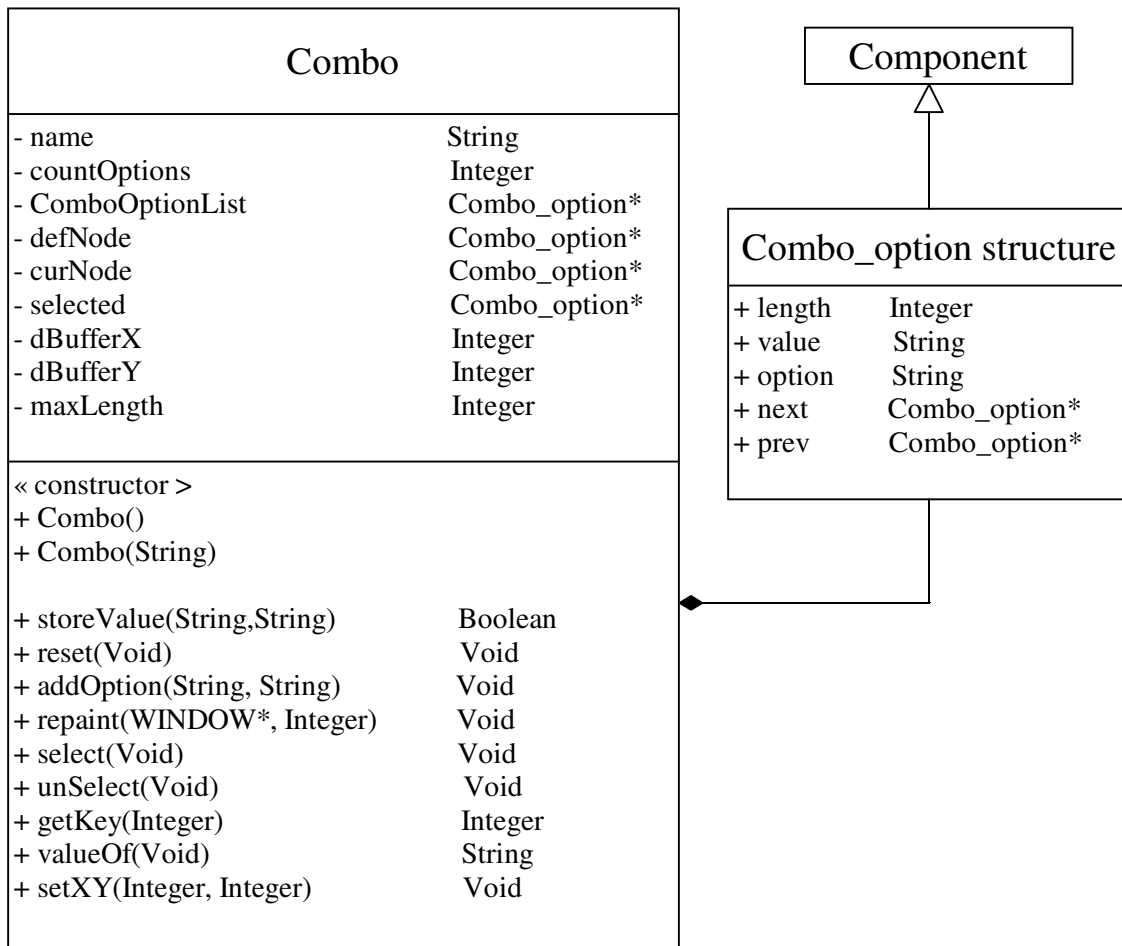
#### **MEMBER FUNCTIONS**

**Name** : **Form**  
**Return Type** : Not Applicable  
**Arguments** : None  
**Visibility** : Public  
**Context** : Constructs the form object.

**Name** : **send**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : Sends the data to the network object to be transmitted to the proxy server.

**Name** : **reset**  
**Return Type** : Void  
**Arguments** : None

Visibility : Public  
 Context : Resets all the form elements.



**Figure 37 Combo Class**

**CLASS NAME** : **Combo**  
*Inherits* : Component  
*Inherited by* : None

**DATA MEMBERS**

**Name** : **name**  
**Type** : String  
**Visibility** : Private  
**Context** : Name of the combo box component.

**Name** : **countOptions**  
**Type** : Integer  
**Visibility** : Private  
**Context** : The number of options included in this combo object

**Name** : **ComboOptionList**  
**Type** : Pointer of type <Combo\_Option>  
**Visibility** : Private  
**Context** : The actual list of options.

**Name** : **defNode**  
**Type** : Pointer of type <Combo\_Option>  
**Visibility** : Private  
**Context** : The default node (selected value) in the ComboOptionList.

**Name** : **curNode**  
**Type** : Pointer of type <Combo\_Option>  
**Visibility** : Private  
**Context** : The currently pointed node in the ComboOptionList. This is the pointer used for adding nodes to the ComboOptionList.

**Name** : **selected**  
**Type** : Pointer of type <Combo\_Option>  
**Visibility** : Private  
**Context** : Points to that node which the user has selected.

**Name** : **dBufferX**  
**Type** : Integer data type  
**Visibility** : Private  
**Context** : The column number in the background buffer of the rendering engine.

<b>Name</b>	:	<b>dBufferY</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	The row number in the background-buffer, where the object is located.

<b>Name</b>	:	<b>maxlength</b>
Type	:	Integer data type
Visibility	:	Public
Context	:	The maximum length of any option. This is used to normalize all other options to the same length.

### ***MEMBER FUNCTIONS***

<b>Name</b>	:	<b>Combo</b>
Return Type	:	Not Applicable
Arguments	:	None
Visibility	:	Public
Context	:	Default constructor; does not do anything.

<b>Name</b>	:	<b>Combo</b>
Return Type	:	Not Applicable
Arguments	:	String
Visibility	:	Public
Context	:	Accepts the <select> tag string as parameter and generates the corresponding Combo object.

<b>Name</b>	:	<b>storeValue</b>
Return Type	:	Boolean
Arguments	:	1.) String 2.) String
Visibility	:	Public
Context	:	Accepts the combo option tag and the text to be displayed for it. Returns true if the option is the selected one. Is used by the addOption method.

<b>Name</b>	:	<b>reset</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	Sets the value of the combo option to its default value.

<b>Name</b>	:	<b>addOption</b>
Return Type	:	Void
Arguments	:	1.) String

2.) String  
 Visibility : Public  
 Context : Adds a new option to the combo box. Accepts the option tag and the text to be displayed for it.

**Name** : **repaint**  
 Return Type : Void  
 Arguments : 1.) Pointer of type WINDOW  
 2.) Integer  
 Visibility : Public  
 Context : Displays the newly selected option onto the background-buffer.

**Name** : **select**  
 Return Type : Void  
 Arguments : None  
 Visibility : Public  
 Context : Selects the object and opens up a temporary window for handling events (letting the user select an option).

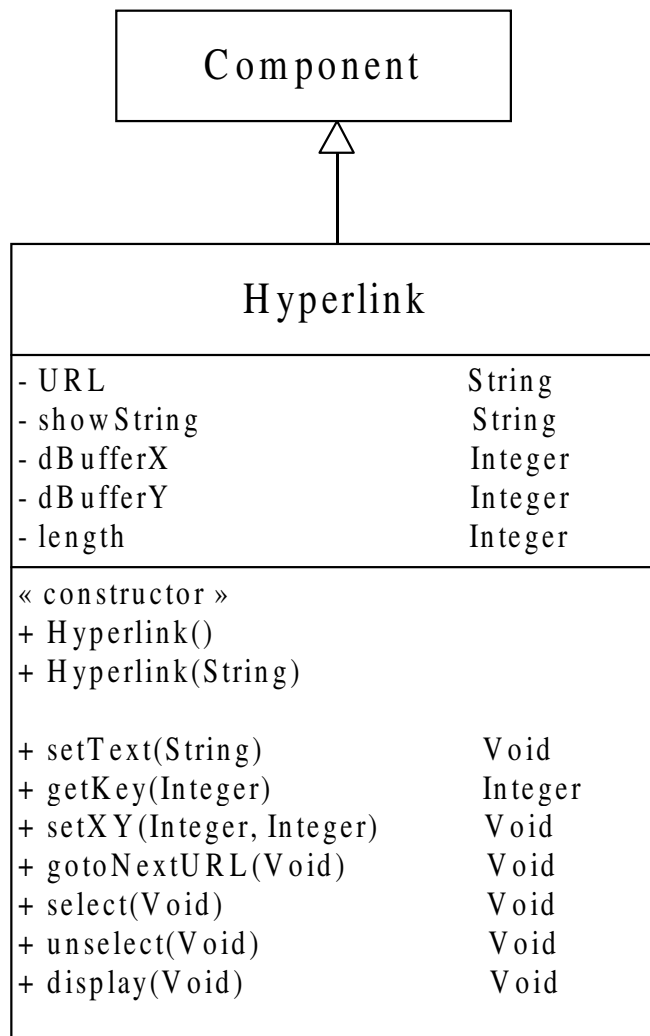
**Name** : **unselect**  
 Return Type : Void  
 Arguments : None  
 Visibility : Public  
 Context : Unselects the object.

**Name** : **getKey**  
 Return Type : Integer  
 Arguments : Integer  
 Visibility : Public  
 Context : Accepts a key and generates the corresponding event.

**Name** : **valueOf**  
 Return Type : String  
 Arguments : None  
 Visibility : Public  
 Context : Returns the string value represented by the currently selected option.

**Name** : **setXY**  
 Return Type : Void  
 Arguments : 1.) Integer  
 2.) Integer  
 Visibility : Public  
 Context : Sets the background buffer coordinates for the object.





**Figure 38 Hyperlink Class**

**CLASS NAME** : **Hyperlink**  
*Inherits* : Component  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **URL**  
Type : String  
Visibility : Private  
Context : Represents the URL of the resource pointed to by the  
hypertext.

**Name** : **showString**  
Type : String  
Visibility : Private  
Context : Represents the hypertext string to be displayed for the  
hyperlink.

**Name** : **dBufferX**  
Type : Integer  
Visibility : Private  
Context : The background-buffer column number of the hyperlink  
object.

**Name** : **dBufferY**  
Type : Integer  
Visibility : Private  
Context : The background-buffer row number of the hyperlink object.

**Name** : **length**  
Type : Integer  
Visibility : Private  
Context : The length of the hypertext's visual representation.

#### **MEMBER FUNCTIONS**

**Name** : **Hyperlink**  
Return Type : Not Applicable  
Arguments : None  
Visibility : Public  
Context : This function is the default constructor of the class, which  
initializes all the data members to some pre-defined values.

**Name** : **Hyperlink**  
Return Type : Not Applicable  
Arguments : String

Visibility	:	Public
Context	:	This function (overloaded constructor) sets the data member <URL> equivalent to value contained in the passed parameter <String>.
<b>Name</b>	:	<b>setText</b>
Return Type	:	Void
Arguments	:	String
Visibility	:	Public
Context	:	Sets the text to be displayed for the hyperlink.
<b>Name</b>	:	<b>getKey</b>
Return Type	:	Integer
Arguments	:	Integer
Visibility	:	Public
Context	:	Accepts keystrokes and generates events accordingly.
<b>Name</b>	:	<b>setXY</b>
Return Type	:	Void
Arguments	:	1.) Integer 2.) Integer
Visibility	:	Public
Context	:	Sets the background-buffer coordinates for this object.
<b>Name</b>	:	<b>gotoNextURL</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	Sets the Global URL to a new one and then returns without further delay.
<b>Name</b>	:	<b>select</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	Selects the object's visible representation for handling events.
<b>Name</b>	:	<b>unselect</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	Unselects the object's visible representation.
<b>Name</b>	:	<b>display</b>
Return Type	:	Void

Arguments	:	None
Visibility	:	Public
Context	:	Diagnostic function used for displaying the hypertext object.

textBox	
- logicalString	WINDOW*
- name	String
- String	String
- charCount	Integer
- stringX	Integer
- stringY	Integer
- dBufferX	Integer
- dBufferY	Integer
- tokens	StringTokenizer
« constructors »	
+ textBox(Void)	
+ textBox(String)	
+ storeString(Void)	Void
+ retrieveString(Void)	Void
+ getKey(Integer)	Integer
+ setXY(Integer, Integer)	Void
+ select(Void)	Void
+ unselect(Void)	Void
+ getLength(Void)	Integer
+ getRow(Void)	Integer
+ getRepresentation(Void)	String
+ valueOf(Void)	String
+ reset(Void)	Void
+ nameOf(Void)	String
+ writeToDb(Void)	String

**Figure 39 Textbox Class**

**CLASS NAME** : **textBox**  
*Inherits* : None  
*Inherited by* : None

**DATA MEMBERS**

**Name** : **logicalString**  
**Type** : Pointer of type WINDOW  
**Visibility** : Private  
**Context** : Stores the complete string in the textbox field, and a part of this string is displayed on the screen.

**Name** : **name**  
**Type** : String  
**Visibility** : Private  
**Context** : This is the name of the textbox specified in the HTML document.

**Name** : **String**  
**Type** : String  
**Visibility** : Private  
**Context** : This stores the character string stored in the textbox and at time of submission this value is sent again.

**Name** : **charCount**  
**Type** : Integer  
**Visibility** : Private  
**Context** : Keeps a count on number of characters entered into the textbox.

**Name** : **stringX**  
**Type** : Integer  
**Visibility** : Private  
**Context** : Serves as the reference point for cursor position along X-axis.

**Name** : **stringY**  
**Type** : Integer  
**Visibility** : Private  
**Context** : Serves as the reference point for cursor position along Y-axis.

<b>Name</b>	:	<b>dBufferX</b>
Type	:	Integer
Visibility	:	Private
Context	:	This variable stores the column number of the text-box (i.e. where is the component located in the double-buffer ).

<b>Name</b>	:	<b>dBufferY</b>
Type	:	Integer
Visibility	:	Private
Context	:	This variable stores the column number of the text-box (i.e. where is the component located in the double-buffer ).

<b>Name</b>	:	<b>tokens</b>
Type	:	Instance of class StringTokenizer
Visibility	:	Public
Context	:	This object has access to the tokenizer stream that extracts out all the required tokens from the <textbox tag> (say, <i>NAME</i> , <i>VALUE</i> ).

### ***MEMBER FUNCTIONS***

<b>Name</b>	:	<b>textBox</b>
Return Type	:	Not applicable
Arguments	:	None
Visibility	:	Public
Context	:	This is the default constructor for the textbox.

<b>Name</b>	:	<b>textBox</b>
Return Type	:	Not applicable
Arguments	:	String
Visibility	:	Public
Context	:	This parameterized constructor accepts the input tag for the textbox and stores the extracted tags back in data-members.

<b>Name</b>	:	<b>storeString</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	This function transfers the contents from the data member <logicalString> into <String>.

<b>Name</b>	:	<b>retrieveString</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public

Context : This function extracts the text from <String> into the textbox field, and is activated whenever textbox is opened for editing.

**Name** : **getKey**

Return Type : Integer

Arguments : Integer

Visibility : Public

Context : This module is an independent event handler for the textbox which is passed control and exits whenever ESCAPE-key or TAB-key is pressed.

**Name** : **setXY**

Return Type : Void

Arguments : 1.) Short integer

2.) Short integer

Visibility : Public

Context : This function is passed two integer parameters that serve as the location of the active element in the double buffer.

**Name** : **select**

Return Type : Void

Arguments : None

Visibility : Public

Context : This function is responsible for making the textbox look as a selected element on the display screen.

**Name** : **unselect**

Return Type : Void

Arguments : None

Visibility : Public

Context : This function is responsible for unselecting the textbox and completing the background operations for doing the same.

**Name** : **getLength**

Return Type : Short integer

Arguments : None

Visibility : Public

Context : This function returns back the length of the textbox (i.e. length of on-screen textbox representation length).

**Name** : **getRow**

Return Type : Short integer

Arguments : None

Visibility : Public

Context : This function returns back the location of the textbox



		(row number in the double buffer).
<b>Name</b>	:	<b>getRepresentation</b>
Return Type	:	String
Arguments	:	None
Visibility	:	Public
Context	:	This function returns back the onscreen character representation of the textbox (which in this case is 18-spaces enclosed within two square brackets).
<b>Name</b>	:	<b>reset</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	This function resets the textbox component back to initial State (i.e. clearing all the text and previous text stored).
<b>Name</b>	:	<b>valueOf</b>
Return Type	:	String
Arguments	:	None
Visibility	:	Public
Context	:	This function returns the String representing the value of the textbox (this is called at time of submitting the page to the server).
<b>Name</b>	:	<b>nameOf</b>
Return Type	:	String
Arguments	:	None
Visibility	:	Public
Context	:	This function returns the name of textbox component as specified in the HTML document.
<b>Name</b>	:	<b>writeToDb</b>
Return Type	:	String
Arguments	:	None
Visibility	:	Public
Context	:	This function calculates the String of length 18-characters that is always displayed on the screen while textbox is active.

password	
- logicalString	WINDOW*
- passwordString	WINDOW*
- name	String
- String	String
- charCount	Integer
- stringX	Integer
- stringY	Integer
- dBufferX	Integer
- dBufferY	Integer
- tokens	StringTokenizer
« constructors »	
+ password(Void)	
+ password(String)	
+ storeString(Void)	Void
+ getKey(Integer)	Integer
+ setXY(Integer, Integer)	Void
+ select(Void)	Void
+ unselect(Void)	Void
+ getLength(Void)	Integer
+ getRow(Void)	Integer
+ getRepresentation(Void)	String
+ valueOf(Void)	String
+ reset(Void)	Void
+ nameOf(Void)	String
+ dBuffString(Void)	String

**Figure 40 Password Class**

**CLASS NAME** : **password**  
*Inherits* : None  
*Inherited by* : None

**DATA MEMBERS**

**Name** : **logicalString**  
**Type** : Pointer of type WINDOW  
**Visibility** : Private  
**Context** : Stores the string containing the ascii values of character (\*) that is displayed in the password box.

**Name** : **passwordString**  
**Type** : Pointer of type WINDOW  
**Visibility** : Private  
**Context** : Stores the complete string that has been entered in the password field.

**Name** : **name**  
**Type** : String  
**Visibility** : Private  
**Context** : This is the name of the password specified in the HTML document.

**Name** : **String**  
**Type** : String  
**Visibility** : Private  
**Context** : This stores the character string stored in the password and at time of submission this value is sent to the console.

**Name** : **charCount**  
**Type** : Integer  
**Visibility** : Private  
**Context** : Keeps a count on number of characters entered into the password.

**Name** : **stringX**  
**Type** : Integer  
**Visibility** : Private  
**Context** : Serves as the reference point for cursor position along X-axis.

**Name** : **stringY**  
**Type** : Integer  
**Visibility** : Private  
**Context** : Serves as the reference point for cursor position along

Y-axis.

<b>Name</b>	:	<b>dBufferX</b>
Type	:	Integer
Visibility	:	Private
Context	:	This variable stores the column number of the password (i.e. where is the component located in the double-buffer ).

<b>Name</b>	:	<b>dBuferY</b>
Type	:	Integer
Visibility	:	Private
Context	:	This variable stores the column number of the password (i.e. where is the component located in the double-buffer ).

<b>Name</b>	:	<b>tokens</b>
Type	:	Instance of class StringTokenizer
Visibility	:	Public
Context	:	This object has access to the tokenizer stream that extracts out all the required tokens from the <textbox tag> ( <i>NAME</i> ).

#### ***MEMBER FUNCTIONS***

<b>Name</b>	:	<b>password</b>
Return Type	:	Not applicable
Arguments	:	String
Visibility	:	Public
Context	:	This parameterized constructor accepts the input tag for the password and stores the extracted tags back in data members.

<b>Name</b>	:	<b>storeString</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	This function transfers the contents from the data member <logicalString> into <String>.

<b>Name</b>	:	<b>getKey</b>
Return Type	:	Integer
Arguments	:	Integer
Visibility	:	Public
Context	:	This module is an independent event handler for the password which is passed control and exits whenever ESCAPE-key or TAB-key is pressed.

**Name** : **setXY**  
**Return Type** : Void  
**Arguments** : 1.) Short integer  
                   2.) Short integer  
**Visibility** : Public  
**Context** : This function is passed two integer parameters that serve as the location of the active element in the double buffer.

**Name** : **select**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : This function is responsible for making the textbox look as a selected element on the display screen.

**Name** : **unselect**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : This function is responsible for unselecting the password and completing the background operations for doing the same.

**Name** : **getLength**  
**Return Type** : Short integer  
**Arguments** : None  
**Visibility** : Public  
**Context** : This function returns back the length of the password (i.e. length of on-screen textbox representation length).

**Name** : **getRow**  
**Return Type** : Short integer  
**Arguments** : None  
**Visibility** : Public  
**Context** : This function returns back the location of the password (row number in the double buffer).

**Name** : **getRepresentation**  
**Return Type** : String  
**Arguments** : None  
**Visibility** : Public  
**Context** : This function returns back the onscreen character representation of the password (which in this case is 18-spaces enclosed within two square brackets).

<b>Name</b>	:	<b>reset</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	This function resets the password component back to initial State (i.e. clearing all the text and previous text stored).

<b>Name</b>	:	<b>valueOf</b>
Return Type	:	String
Arguments	:	None
Visibility	:	Public
Context	:	This function returns the String representing the value of the password (this is called at time of submitting the page to the server).

<b>Name</b>	:	<b>nameOf</b>
Return Type	:	String
Arguments	:	None
Visibility	:	Public
Context	:	This function returns the name of password component as specified in the HTML document.

<b>Name</b>	:	<b>dBuffString</b>
Return Type	:	String
Arguments	:	None
Visibility	:	Public
Context	:	This function calculates the String of length 18-characters that is always displayed on the screen while textbox is active.

#### **8.3.1.1.2.2** *Utility Classes*

Utility classes are the set of classes which are used by certain other classes or objects of these classes are shared between various classes or objects of these classes. Because of their nature of reusability or sharing they have been specified separately. Classes such as tokenizer, string tokenizer etc have been clubbed under this category.

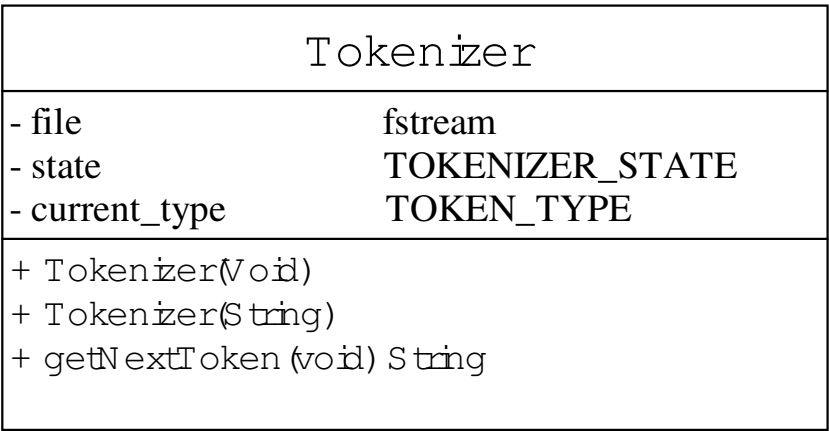


Figure 41 Tokenizer Class



<b>CLASS NAME</b>	:	<b>Tokenizer</b>
<i>Inherits</i>	:	None
<i>Inherited by</i>	:	None

#### **DATA MEMBERS**

<b>Name</b>	:	<b>file</b>
Type	:	object of type <fstream>
Visibility	:	Private
Context	:	This is the object of the file, which is read and parsed in pass1 stage.

<b>Name</b>	:	<b>state</b>
Type	:	Data type <TOKENIZER_STATE>
Visibility	:	Private
Context	:	This represents the state of the tokenizer object,( whether inside some token or not).

<b>Name</b>	:	<b>current_type</b>
Type	:	Data type <TOKEN_TYPE>
Visibility	:	Public
Context	:	Represents whether the current token is a TEXT_TOKEN or a TAG_TOKEN.

#### **MEMBER FUNCTIONS**

<b>Name</b>	:	<b>Tokenizer</b>
Return Type	:	Not Applicable
Arguments	:	None
Visibility	:	Public
Context	:	Default contructor, does nothing

<b>Name</b>	:	<b>Tokenizer</b>
Return Type	:	Not Applicable
Arguments	:	String
Visibility	:	Public
Context	:	Constructor accepts the name of the file as input and attaches a stream to it.

<b>Name</b>	:	<b>getNextToken</b>
Return Type	:	String
Arguments	:	Void
Visibility	:	Public
Context	:	Returns the next token in the file stream and sets the current_type value to the type of token being returned.

StringTokenizer		
- str		String
- delim		Integer
- count		Integer
« constructor »		
+ StringTokenizer(Void)		
+ StringTokenizer(String,String)		
+ isDelimiter(Char)		Boolean
+ nextToken(Void)		String

**Figure 42 StringTokenizer Class**

**CLASS NAME** : **StringTokenizer**  
*Inherits* : None  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **str**  
Type : String  
Visibility : Private  
Context : Represents the string to be tokenized.

**Name** : **delim**  
Type : Character array  
Visibility : Private  
Context : This character array is a collection of delimiting characters.

**Name** : **count**  
Type : Integer  
Visibility : Private  
Context : Counts the number of tokens already generated

#### **MEMBER FUNCTIONS**

**Name** : **StringTokenizer**  
Return Type : Not Applicable  
Arguments : None  
Visibility : Public  
Context : Default constructor; sets all attributes to null.

**Name** : **StringTokenizer**  
Return Type : Not Applicable  
Arguments : 1.) String  
2.) String  
Visibility : Public  
Context : Copies the strings *str* and *delim* as passed into the constructor.

**Name** : **isDelimiter**  
Return Type : Boolean  
Arguments : Character  
Visibility : Public  
Context : Returns true if the character passed to it is a delimiter as specified by the *delim* array.

<b>Name</b>	:	<b>nextToken</b>
Return Type	:	String
Arguments	:	Void
Visibility	:	Public
Context	:	Generates and returns the next token in the string.

pass3_Tokenizer	
- special_delimiters	String
« constructor »	
+ pass3_Tokenizer(Void)	Void
+ findToken(String,Rendering_Engine&)	Void
+ ifDelimiter(Character)	Boolean

**Figure 43 Pass3\_Tokenizer Class**

**CLASS NAME** : **pass3\_Tokenizer**  
*Inherits* : None  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : special\_delimiters  
**Type** : Character array  
**Visibility** : Private  
**Context** : This character array holds all the “delimiter codes” that differentiates normal text from special text (i.e. delimiters). These delimiters are none other than tags like <Bold>, <U> or <Img>.

#### **MEMBER FUNCTIONS**

**Name** : **pass3\_Tokenizer**  
**Return Type** : Not applicable  
**Arguments** : None  
**Visibility** : Public  
**Context** : This is the default constructor for the class, which initializes the data members to some default value.

**Name** : **findToken**  
**Return Type** : Void  
**Arguments** : 1.) String  
                   2.) Rendering\_Engine& (object reference)  
**Visibility** : Public  
**Context** : This function parses the output file returned by Pass2 (i.e. pass2.tmp) for occurrence of special delimiters along with normal html text. This is then passed to other functions for processing and finally rendering it to the screen.

**Name** : **ifDelimiter**  
**Return Type** : Boolean  
**Arguments** : Character  
**Visibility** : Public  
**Context** : This function returns TRUE if the character read from file <pass2.tmp> matches some delimiter code.

UniformResourceLocator :structure		
+ URL		String
+ oldURL		String
+ callType		ReqType
+ setURL(String , ReqType)		Void

**Figure 44 URL Class**

**CLASS NAME** : **Uniform Resource Locator**  
*Inherits* : None  
*Inherited by* : None

**DATA MEMBERS**

**Name** : **URL**  
Type : String  
Visibility : Public  
Context : Stores the current URL.

**Name** : **oldURL**  
Type : String  
Visibility : Public  
Context : Stores the previous URL

**Name** : **callType**  
Type : Enumeration ReqType  
{ NEW\_URL, HYPER\_EVENT, REFRESH\_EVENT }  
Visibility : Public  
Context : Represents the event that caused the URL to be set

**MEMBER FUNCTIONS**

**Name** : **setURL**  
Return Type : Void  
Arguments : URL String  
Visibility : 1) String  
                  2) ReqType  
Context : Sets the URL and copies the current URL to the old URL.

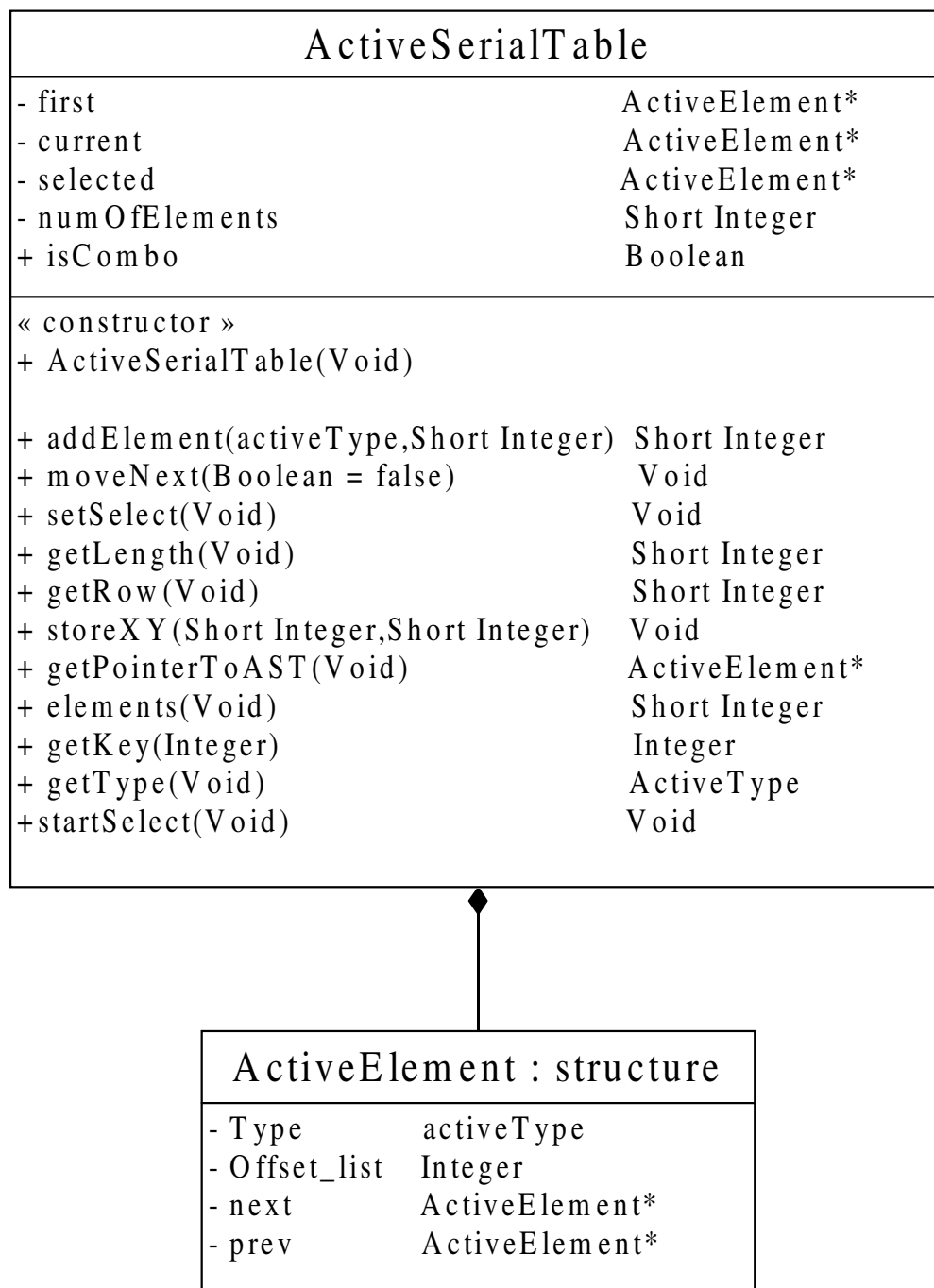


#### **8.3.1.1.2.3** *Data Structure Classes*

Data structures are used to store vital information which is often needed again and again. The web browser of *iBOS* too generates a lot of data structures which are used at various stages starting from requesting a web page to event handler. The following section explains data structure classes which are used to generate objects to store vital information.

Some of the data structures are:

- AST
- linked list
- table class
- list Stack



**Figure 45 ActiveSerialTable Class**

**CLASS NAME** : **ActiveSerialTable**

*Inherits* : None

*Inherited by* : None

### **DATA MEMBERS**

**Name** : **selected**

Type : Pointer of type <ActiveElement>

Visibility : Private

Context : Refers to the node that is presently active. This node is actually one of the real active elements, for example, a hyperlink.

**Name** : **first**

Type : Pointer of type <ActiveElement>

Visibility : Private

Context : Refers to the first element of the linked list.

**Name** : **numOfElements**

Type : Integer

Visibility : Private

Context : Keeps a count of the number of active elements added to the active serial table.

### **MEMBER FUNCTIONS**

**Name** : **addElement**

Return Type : Void

Arguments : 1.) String  
2.) Data type <activetype>

Visibility : Public

Context : This method adds an active element of a given tag string and type.

**Name** : **getLength**

Return Type : Integer

Arguments : None

Visibility : Public

Context : This method returns the length of an active element whose offset is passed as the parameter.

**Name** : **storeXY**

Return Type : Void

Arguments : 1.) Integer  
2.) Integer

Visibility : Public

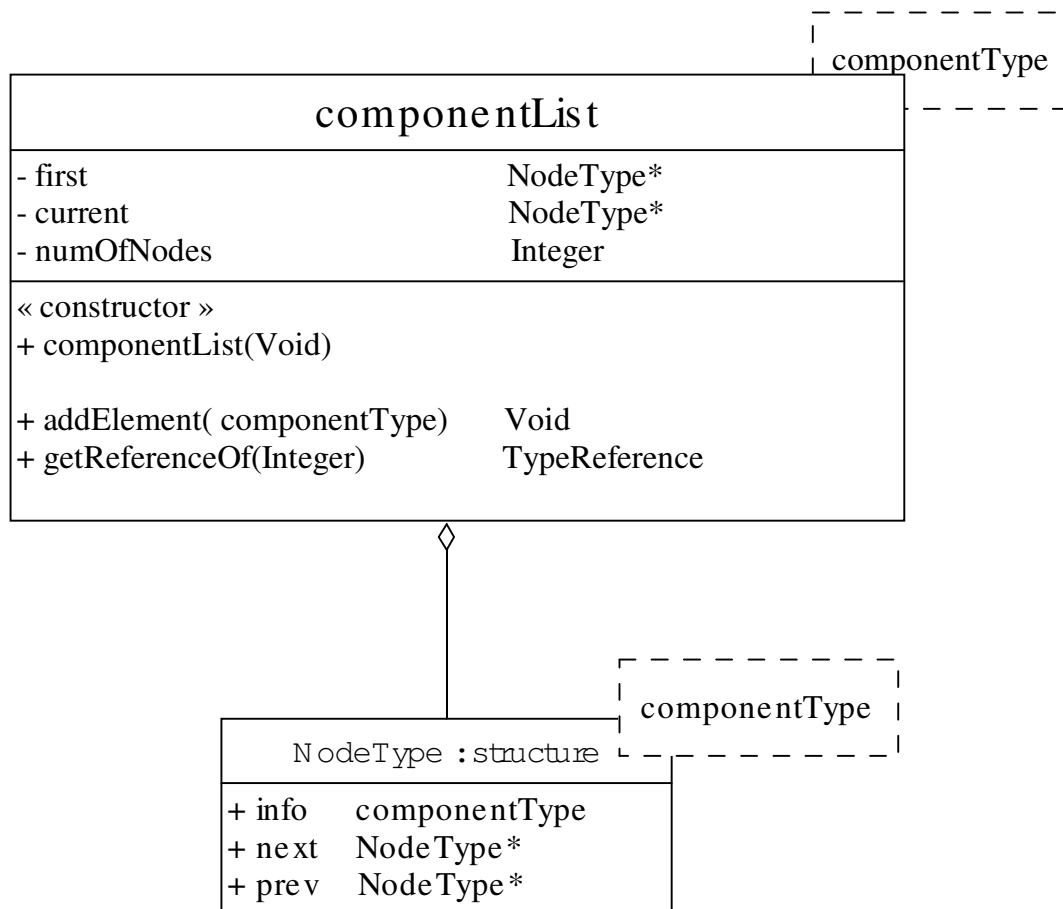
Context : Stores the background buffer of the currently selected

object into the object's data structure.

<b>Name</b>	:	<b>storeScreenXY</b>
Return Type	:	Void
Arguments	:	1.) Integer 2.) Integer
Visibility	:	Public
Context	:	Stores the screen coordinates of the currently selected element in the object's data structure.

<b>Name</b>	:	<b>moveNext</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	Moves the selected pointer to the next element in the list.



**Figure 46 componentList Class**

**CLASS NAME** : **componentList**  
*Inherits* : None  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **first**  
**Type** : Pointer of type <NodeType>  
**Visibility** : Private  
**Context** : Refers to the first element of the queue data structure.

**Name** : **current**  
**Type** : Pointer of type <NodeType>  
**Visibility** : Private  
**Context** : Refers to the tail element of the queue.

**Name** : **numOfNodes**  
**Type** : Integer  
**Visibility** : Private  
**Context** : Refers to the number of nodes already added to the queue.

#### **MEMBER FUNCTIONS**

**Name** : **componentList**  
**Return Type** : Not Applicable  
**Arguments** : None  
**Visibility** : Public  
**Context** : The default constructor initializes all pointers to null, and the node-count to zero.

**Name** : **addElement**  
**Return Type** : Void  
**Arguments** : Data element of type <componentType>  
**Visibility** : Public  
**Context** : Adds a new object to the component list.

**Name** : **getReferenceOf**  
**Return Type** : Data element of type <componentType>  
**Arguments** : Integer  
**Visibility** : Public  
**Context** : Returns a reference to the object located at *offset* position in the queue.

table	
- col_count	Integer
- max_cols	Integer
- col_length	Integer*
- max_col_length	Integer*
« constructor »	
+ table(Void)	
+ table(Integer)	
+ getCount(Void)	Integer
+ getMaxCount(Void)	Integer
+ getLength(Integer)	Integer
+ getMaxLength(Integer)	Integer
+ addColumn(Integer,Integer)	Void
+ updateMaxLength(Integer,Integer)	Void
+ endRow(Void)	Void

**Figure 47 table Class**

**CLASS NAME** : **table**  
*Inherits* : None  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **col\_count**  
Type : Integer data type  
Visibility : Private  
Context : Keeps a track of total number of columns in a table.

**Name** : **max\_cols**  
Type : Integer data type  
Visibility : Private  
Context : A counter for maximum number of columns in any row of the table.

**Name** : **col\_length**  
Type : Pointer of type integer  
Visibility : Private  
Context : Pointer to array that stores lengths of various columns in a table.

**Name** : **max\_col\_lengths**  
Type : Pointer of type integer  
Visibility : Private  
Context : Pointer to array that stores maximum length of any column in a table.

#### **MEMBER FUNCTIONS**

**Name** : **table**  
Return Type : Not Applicable  
Arguments : None  
Visibility : Public  
Context : Default constructor of the class that simply initializes various data members to their default value.

**Name** : **table**  
Return Type : Not Applicable  
Arguments : Integer  
Visibility : Public  
Context : One argument constructor that initializes various data members to default values and offset to value of the parameter passed.

.



**Name** : **getCount**  
 Return Type : Integer data type.  
 Arguments : Null  
 Visibility : Public  
 Context : Returns the value of <col\_count>

**Name** : **getMaxCount**  
 Return Type : Integer data type.  
 Arguments : Null  
 Visibility : Public  
 Context : Returns the value of <max\_cols>

**Name** : **getLength**  
 Return Type : Integer data type.  
 Arguments : Integer  
 Visibility : Public  
 Context : Returns the length of a column from <col\_length> array stored at the position equal to the index passed as parameter.

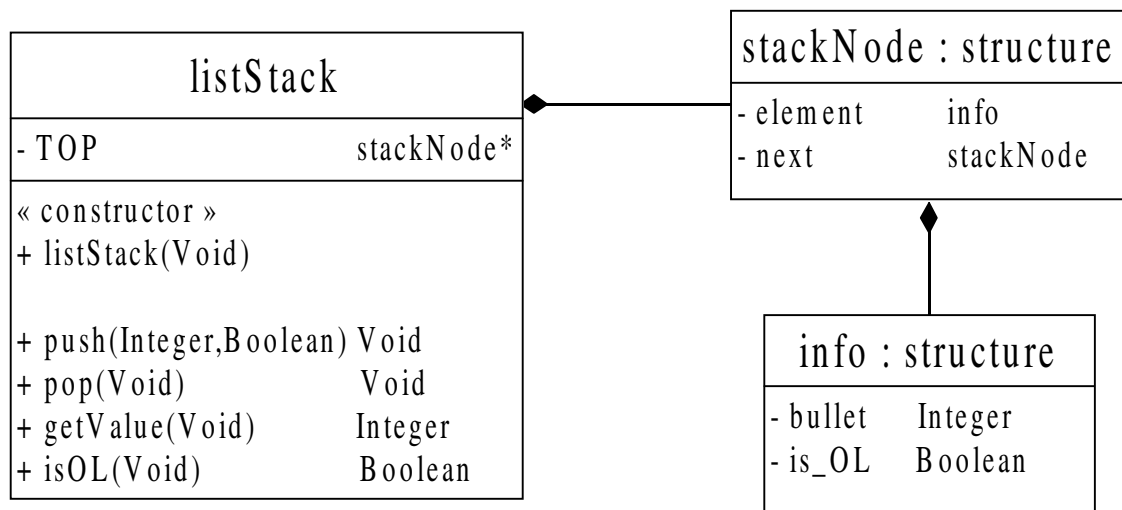
**Name** : **getMaxLength**  
 Return Type : Integer data type.  
 Arguments : Integer  
 Visibility : Public  
 Context : Returns the maximum length of a column from <max\_col\_length> array stored at the position equal to the index passed as parameter.

**Name** : **addColumn**  
 Return Type : Null  
 Arguments : 1) Integer  
               2) Integer  
 Visibility : Public  
 Context : This function adds an entry in the <col\_length> array. The first integer argument acts as the index specifying the position of entry and the second argument specifies the value to be added. This function then calls updateMaxLength(int,int) function.

**Name** : **updateMaxLength**  
 Return Type : Null  
 Arguments : 1) Integer  
               2) Integer  
 Visibility : Public  
 Context : This function updates/adds an entry in the

<max\_col\_length> array. The first integer argument acts as the index specifying the position where the entry is to be added/updated and the second argument specifies the value to be added/updated.

<b>Name</b>	:	<b>endRow</b>
Return Type	:	Null
Arguments	:	Null
Visibility	:	Public
Context	:	Inserts a special value '-1' in the <col_length> array to mark the end of a row in a table.



**Figure 48 listStack Class**

<b>CLASS NAME</b>	:	<b>listStack</b>
<i>Inherits</i>	:	None
<i>Inherited by</i>	:	None

#### **DATA MEMBERS**

<b>Name</b>	:	<b>TOP</b>
Type	:	Pointer of type stackNode (a structure)
Visibility	:	Private
Context	:	This pointer serves as a flag that always points to the top of the listStack. Thus at any point of time a reference via TOP gives the currently open <list tag> in the parsing process.

#### **MEMBER FUNCTIONS**

<b>Name</b>	:	<b>listStack</b>
Return Type	:	Not Applicable
Arguments	:	None
Visibility	:	Public
Context	:	This function serves as the default constructor of the class that simply initializes all the data members.

<b>Name</b>	:	<b>push</b>
Return Type	:	Void
Arguments	:	1.) Integer 2.) Boolean
Visibility	:	Public
Context	:	This function saves the current bullet symbol <Integer> and a Boolean data type <Boolean> that signifies whether it is an <OL list> or not.

<b>Name</b>	:	<b>pop</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	This function simply pops the topmost <bullet> and <boolean> flag.

<b>Name</b>	:	<b>getValue</b>
Return Type	:	Integer
Arguments	:	None
Visibility	:	Public
Context	:	This function returns incremented value if the <boolean> flag at the top

<b>Name</b>	:	<b>isOL</b>
Return Type	:	Boolean
Arguments	:	None
Visibility	:	Public
Context	:	This function returns TRUE if the list type on top of stackList is <Ordered List>

#### **8.3.1.1.2.4** *Pass specific Classes*

Certain classes are specific for a phase of a parser. Due to their nature these classes have been categorized separately. These classes are used during layout of tables and final rendering of web page on double buffer.

table_engine	
- fr	File*
- fw	File*
- offset	Integer
- special_delimiters	String
- isDelimiter	Boolean
- count	Integer
- buffer	String
- firstPass2	Boolean
- t	Table
- insideTable	Boolean
- insideTr	Boolean
- insideTd	Boolean
- trTdCount	Integer
- tableTdCount	Integer
- StringLength	Integer
- String	String
- leadingSpaces	Integer
- trailingSpaces	Integer
- isAlignSpecified	Boolean
« constructor»	
+ table_engine(Void)	
+ table_engine(String)	
+ if_delimiter(Character)	Boolean
+ write_to_buffer(Void)	Void
+ write_from_buffer(Void)	Void

**Figure 49 table\_engine Class**

**CLASS NAME** : **table\_engine**

*Inherits* : None

*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **fr**

Type : Pointer of type FILE

Visibility : Private

Context : A file pointer used to read a temporary file character by character.

**Name** : **fw**

Type : Pointer of type FILE

Visibility : Private

Context : A file pointer used to write a temporary file character by character.

**Name** : **offset**

Type : Integer data type

Visibility : Private

Context : Offset keeps a track of number of bytes read from the file.

**Name** : **special\_delimiters**

Type : Character array

Visibility : Private

Context : An array that contains table specific tokens which are intermediate form of html table tags such as <table>, <tr>, <td> etc.

**Name** : **isDelimiter**

Type : Boolean data type

Visibility : Private

Context : A boolean flag that keeps track whether a given token is a delimiter or a normal text.

**Name** : **Count**

Type : Integer data type

Visibility : Private

Context : A counter that keeps counts of number of characters in the buffer.

**Name** : **Buffer**

Type : Character data type

Visibility : Private

Context : A buffer to store characters being read from the



intermediate file. The same buffer is then used to write to another intermediate file.

<b>Name</b>	:	<b>firstPass2</b>
Type	:	Boolean data type.
Visibility	:	Private
Context	:	A boolean flag that keeps track whether a given table specific tags are read for the first time or second time.
<b>Name</b>	:	<b>t</b>
Type	:	Object of type table class
Visibility	:	Private
Context	:	An object of class table to store lengths and maximum lengths of various columns when table specific tags are parsed for the first time.
<b>Name</b>	:	<b>trTdCount</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	Counts the number of td tags encountered in within a tr tag.
<b>Name</b>	:	<b>tableTdCount</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	Counts the number of td tags encountered in within a table.
<b>Name</b>	:	<b>stringLength</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	The length of the string finally being displayed in a cell of a table.
<b>Name</b>	:	<b>leadingSpace</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	Number of leading spaces to be inserted before the final string is displayed on the screen.
<b>Name</b>	:	<b>trailingSpace</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	Number of trailing spaces to be inserted after the final string is displayed on the screen.
<b>Name</b>	:	<b>isAlignSpecified</b>
Type	:	Boolean data type

Visibility : Private  
 Context : A Boolean variable to mark that alignment within the cell has been specified or not.

**Name** : **insideTable**  
 Type : Boolean data type.  
 Visibility : Private  
 Context : A boolean flag to mark the state of automation. It signifies weather the file pointer <fr> at given time has encountered a <table> or a </table> tag.

**Name** : **insideTr**  
 Type : Boolean data type.  
 Visibility : Private  
 Context : A boolean flag to mark the state of automation. It signifies weather the file pointer <fr> at given time has encountered a <tr> or a </tr> tag.

**Name** : **insideTd**  
 Type : Boolean data type.  
 Visibility : Private  
 Context : A boolean flag to mark the state of automation. It signifies weather the file pointer <fr> at given time has encountered a <td> or a </td> tag.

### ***MEMBER FUNCTIONS***

**Name** : **table\_engine**  
 Return Type : Not Applicable  
 Arguments : None  
 Visibility : Public  
 Context : Default constructor of the class that simply initializes various data members to their default value.

**Name** : **table\_engine**  
 Return Type : Not Applicable  
 Arguments : String  
 Visibility : Public  
 Context : Default constructor of the class that simply initializes various data members to their default value and the file to be read to the value of argument of type string.

**Name** : **if\_Delimiter**  
 Return Type : Boolean data type  
 Arguments : Character  
 Visibility : Public.  
 Context : Returns true/ false according to weather the character

passed is a delimiter or not.

<b>Name</b>	:	<b>write_to_buffer</b>
Return Type	:	Not Applicable
Arguments	:	Null
Visibility	:	Public
Context	:	This function simply reads from the intermediary file and fills the buffer.

<b>Name</b>	:	<b>write_from_buffer</b>
Return Type	:	Not Applicable
Arguments	:	Null
Visibility	:	Public
Context	:	This function simply reads from the buffer, interprets the characters read and accordingly writes to another intermediary file.

Rendering_Engine	
- dBuffer	WINDOW *
- S	listStack
- centerAlign	Boolean
- isActiveE	Boolean
- margin	Integer
- aE_X	Integer
- aE_Y	Integer
- tempX	Integer
- tempY	Integer
- OL_count	Integer
- UL_count	Integer
- UL_bullets	String
- last_aE_Y	Integer
- is_last_aE	Boolean
- aE_Found	Integer
- aE_Position	Integer Array
« constructor »	
+ Rendering_Engine()	
+ Rendering_Engine(String , Integer)	
+ browserWindow(Void)	Void
+ acceptToken(String , Integer)	Void
+ toTempBuffer(Boolean, String, Integer)	Void
+ toTempBuffer(Character)	Void
+ toDoubleBuffer(Void)	Void
+ writeToDB(Integer, Integer,String,Integer)	Void
+ scrollText(Integer)	Void
+ scrollNext_aE(Void)	Void
+ setActiveE_Position(Integer , Integer)	Void
+ updateActiveE_Position(Integer)	Void

**Figure 50 Rendering\_Engine Class**

**CLASS NAME** : **Rendering\_Engine**

*Inherits* : None

*Inherited by* : None

**DATA MEMBERS**

**Name** : **dBuffer**

Type : Pointer of type WINDOW

Visibility : Private

Context : Serves as Double Buffer in which all the formatted text is stored, before rendering it to the screen.

**Name** : **S**

Type : Object of class <listStack>

Visibility : Private

Context : This data structure serves as a temporary stack onto which the last bullet symbol is pushed. It has been devised to serve while parsing nested list structures in Html.

**Name** : **centerAlign**

Type : Boolean data type

Visibility : Private

Context : Tells the status of alignment mode (i.e. whether to align the text normally or in center of the display screen).

**Name** : **is\_last\_aE**

Type : Boolean data type

Visibility : Private

Context : If this status flag is set to TRUE, then it denotes that last element has been visited by the user while switching between various active elements

**Name** : **isActiveE**

Type : Boolean data type

Visibility : Private

Context : Serves as a reference flag, that tells whether the current text being rendered into the double buffer is text representation of some active element or not.

**Name** : **margin**

Type : Integer data type

Visibility : Private

Context : Sets the margin from leftmost corner of the display screen.

<b>Name</b>	:	<b>last_aE_Y</b>
Type	:	Integer
Visibility	:	Private
Context	:	This integer variable stores the row of the last active element encountered while rendering them.
<b>Name</b>	:	<b>aE_X</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	Position of active element in the double buffer, relative to the X-axis.
<b>Name</b>	:	<b>aE_Y</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	Position of active element in the double buffer, relative to the double buffer.
<b>Name</b>	:	<b>aE_Found</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	This denotes the number of active elements encountered in a single row (only if Active align tag is active).
<b>Name</b>	:	<b>aE_Position</b>
Type	:	Integer number array
Visibility	:	Private
Context	:	This array stores the positions of active elements found in a single row. The size of array has been limited to 78 since, in a row a maximum of 78 active elements can be center aligned.
<b>Name</b>	:	<b>tempX</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	Stores the X-coordinate (i.e. the temporary buffer of characters) to which text will be copied.
<b>Name</b>	:	<b>tempY</b>
Type	:	Integer data type
Visibility	:	Private
Context	:	Stores the Y-coordinate (i.e. the temporary buffer of characters) to which text will be copied.

**Name** : **OL\_count**  
**Type** : Integer data type  
**Visibility** : Private  
**Context** : Keeps a count on how many <OL tags> are open. (I.e. for which a closing tag has not been encountered)

**Name** : **UL\_count**  
**Type** : Integer data type  
**Visibility** : Private  
**Context** : Keeps a count on how many <UL tags> are open. (I.e. for which a closing tag has not been encountered)

**Name** : **UL\_bullets**  
**Type** : Array of Integer elements.  
**Visibility** : Private  
**Context** : Different symbols (i.e. Bullets) which are unique for each level of nesting in case of <UL tags> (i.e. unordered list)

### ***MEMBER FUNCTIONS***

**Name** : **Rendering\_Engine**  
**Return Type** : Not Applicable  
**Arguments** : None  
**Visibility** : Public  
**Context** : This function is the default constructor of the class. It initializes the data members to some pre-defined values.

**Name** : **acceptToken**  
**Return Type** : Void  
**Arguments** : 1.) String  
               2.) Integer data type  
**Visibility** : Public  
**Context** : Accepts the tokens generated by the function <findToken> and takes appropriate action on basis of current token found.

**Name** : **toTempBuffer**  
**Return Type** : Void  
**Arguments** : 1.) Boolean  
               2.) String  
               3.) Integer  
**Visibility** : Public  
**Context** : This function is responsible for pasting the token into the temporary buffer and then into the double buffer.

<b>Name</b>	:	<b>toTempBuffer</b>
Return Type	:	Void
Arguments	:	1.) Character data type
Visibility	:	Public
Context	:	This function passes the character to above function <toTempBuffer>, which again takes the same semantic actions.

<b>Name</b>	:	<b>toDoubleBuffer</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	This module copies the text from temporary buffer to the double buffer. Finally this text is rendered to the display screen.

<b>Name</b>	:	<b>writeToDB</b>
Return Type	:	Void
Arguments	:	1.) Integer 2.) Integer 3.) String
Visibility	:	Public
Context	:	This function has been provided for active elements. It is required when the operations regarding an active element have been completed (say, selecting an option from the combo box). This function provides the capability of replacing the text with <String> at position specified by passed <Integer> coordinates in the double buffer

<b>Name</b>	:	<b>scrollText</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	This function displays the text (i.e. copies the text from double buffer to the display screen). This function also handles the vertical scrolling of the browser window, in case the rendered text cannot be shown on a single screen.

<b>Name</b>	:	<b>scrollNext_aE</b>
Return Type	:	Void
Arguments	:	None
Visibility	:	Public
Context	:	This is responsible for automatically scrolling the page up



or down depending on the fact that whether next active element lies above or below the current display section.

#### **8.3.1.1.3 Event Handler Classes**

Once the web page is parsed and laid out on the screen, the event handler takes over. The event handler traps the user input and accordingly takes action. If the events are widget specific, the keys trapped are passed to the particular widgets which take the necessary action on its own other wise if the keystrokes are system specific the event handler takes action.

The following section describes event handler class.

eventHandler	
- key	Integer
+ globalMode	Boolean
- transition	Boolean
« constructor »	
+ eventHandler(Void)	
+ handleEvents (Void)	Boolean

**Figure 51 eventHandler Class**

**CLASS NAME** : **eventHandler**

*Inherits* : None

*Inherited by* : None

**DATA MEMBERS**

**Name** : **currentMode**

Type : Boolean

Visibility : Private

Context : This data value is set to TRUE only if the current mode is <Global> else it is set to FALSE.

**Name** : **key**

Type : Integer

Visibility : Private

Context : This holds the value of the keystroke returned by the user.

**MEMBER FUNCTIONS**

**Name** : **eventHandler**

Return Type : Not Applicable

Arguments : None

Visibility : Public

Context : This is the default constructor of the class, and simply puts the event handle in <Global> mode.

**Name** : **handleEvents**

Return Type : Boolean

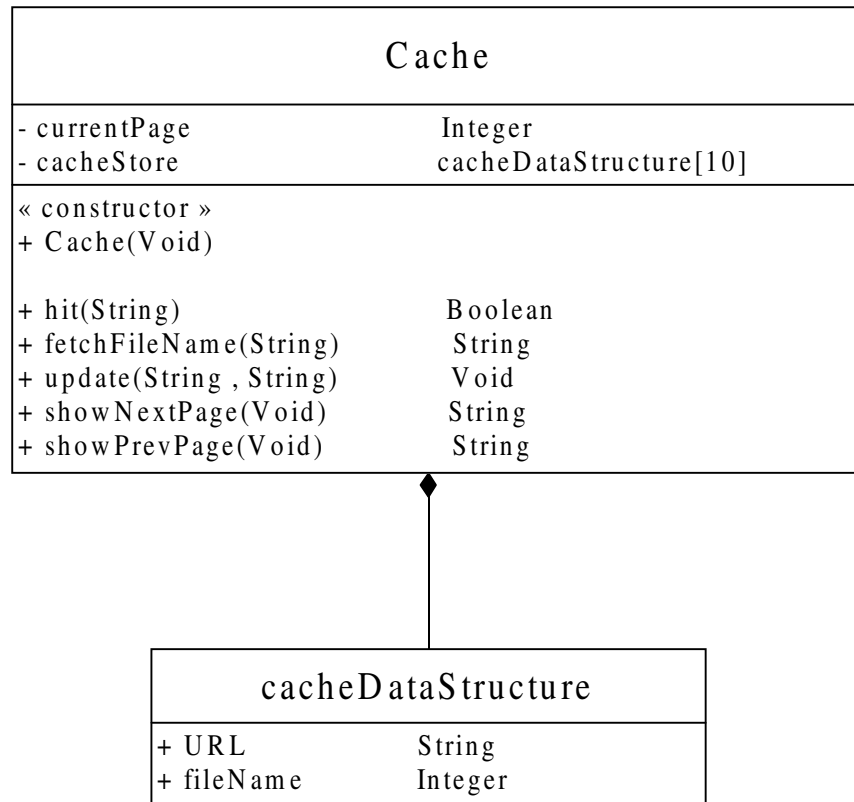
Arguments : None

Visibility : Public

Context : This function is basically an infinite loop that catches all the keystrokes generated by the user, and returns TRUE only if the Exit key-sequence is pressed.

#### **8.3.1.1.4 Caching Classes**

Cache manager is an important component of any web browser. *iBOS* too implements caching. When a page is requested the cache is sought, if the page is available in the cache it's a hit and the page is loaded from the cache. However if the page is not found in the cache, a new request is sent. The web page is fetched, parsed, laid on the screen and its entry is made with the cache manager. The following section describes the cache class of *iBOS*.



**Figure 52 Cache Class**

**CLASS NAME** : **Cache**  
*Inherits* : None  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **currentPage**  
**Type** : Integer  
**Visibility** : Private  
**Context** : This serves as an index into the cache store where the current page is stored.

**Name** : **cacheStore**  
**Type** : Array of type <cacheDataStructure>  
**Visibility** : Private  
**Context** : This is the cache where the last 10 viewed pages are stored.

#### **MEMBER FUNCTIONS**

**Name** : **Cache**  
**Return Type** : Not Applicable  
**Arguments** : None  
**Visibility** : Public  
**Context** : This is the default constructor of the class, which initializes the data members to some default values.

**Name** : **hit**  
**Return Type** : Boolean  
**Arguments** : String  
**Visibility** : Public  
**Context** : This function returns TRUE if the requested URL <String> is found in the cache itself.

**Name** : **fetchFileName**  
**Return Type** : String  
**Arguments** : String  
**Visibility** : Public  
**Context** : This function returns the name of the file <String> corresponding to the URL <String>.

**Name** : **update**  
**Return Type** : Void  
**Arguments** : 1.) String

2.) String  
Visibility : Public  
Context : This function updates the cache store (i.e. adding, deleting, or replacing a page present in the cache).

**Name** : **showNextPage**  
Return Type : String  
Arguments : None  
Visibility : Public  
Context : This function returns the URL that follows next in order of occurrence into the cache.

**Name** : **showPrevPage**  
Return Type : String  
Arguments : None  
Visibility : Public  
Context : This function returns the URL previous to the current URL.

#### **8.3.1.2 Proxy-Server classes:**

The Proxy-Server class is an integral part of Network Interaction Module (NIM) which is deployed at the server end of *iBOS*.

The following is the major functionality performed by the Proxy-Server Class:

- Sends request to the remote web server for the page requested by the *iBOS client*.
- Accepts the response from the remote web server and sends the response to *iBOS client*.
- The requested pages are cached at the Proxy server also (along with a local cache store at the *iBOS* client end) to increase performance.



iBOSClientHandler	
- conn	Socket
- netWriter	PrintWriter
- netRead	BufferedReader
« constructor »	
+ iBOSClientHandler(Void)	
+ run(Void)	Void
+ handleSimpleGet(Void)	Void
+ handleRefreshGet(Void)	Void
+ handleformGet(Void)	Void
+ handleformPost(Void)	Void

**Figure 53 iBOS Client Handler**

**CLASS NAME** : **iBOSClientHandler**

*Inherits* : java.lang.Thread

*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **conn**

Type : Socket

Visibility : Private

Context : This is a socket for communication with an inbound client.

**Name** : **netWriter**

Type : PrintWriter

Visibility : Private

Context : This is an object that produces network output when a print or println method is called on it.

**Name** : **netRead**

Type : BufferedReader

Visibility : Private

Context : This is an object that produces network input when a read or readLine method is called on it.

#### **MEMBER FUNCTIONS**

**Name** : **iBOSClientHandler**

Return Type : Not Applicable

Arguments : None

Visibility : Public

Context : This is the default constructor for the class that initializes data members to default values.

**Name** : **run**

Return Type : Not Applicable

Arguments : None

Visibility : Public

Context : An inherited method (from java.lang.Thread), it is a method that characterizes the execution of single thread.

**Name** : **handleSimpleGet**

Return Type : Void

Arguments : None

Visibility : Public

Context : This method handles HTTP requests based on simple GET messages.

**Name** : **handleRefreshGet**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : This method handles a page-refresh request from an *iBOS* client.

**Name** : **handleFormGet**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : This method handles the form posting request based on the GET method.

**Name** : **handleFormPost**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public  
**Context** : This method handles the form posting request based on the HTTP Post method.

iBOSPServer	
+ main (Void)	Void

**Figure 54 iBOS Proxy Server**

***CLASS NAME*** : **iBOSPServer**

*Inherits* : None

*Inherited by* : None

***MEMBER FUNCTIONS***

**Name** : **main**

Return Type : Integer

Arguments : None

Visibility : Public

Context : This function is analogous to an outermost shell that supervises all other sub-processes.

Cachingjobs	
- nextfilename	String
+ AddEntry(String)	Void
+ getNextFilename(Void)	String
+ IsExactMatch(String)	String
+ IsPartialMatch(String)	Boolean

**Figure 55 Server Side Caching**

**CLASS NAME** : **Cachingjobs**

*Inherits* : None

*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **nextfilename**

Type : String

Visibility : Private

Context : This variable stores the next name to be returned to a thread wanting to store a new cache entry.

#### **MEMBER FUNCTIONS**

**Name** : **AddEntry**

Return Type : Void

Arguments : String

Visibility : Public

Context : Adds a new database entry for the URL and filename.

**Name** : **getNextFilename**

Return Type : Void

Arguments : String

Visibility : Public

Context : Returns the next filename available for storage in the cache.

**Name** : **IsExactMatch**

Return Type : String

Arguments : String

Visibility : Public

Context : Attempts to match an input URL exactly against a single cache entry.

**Name** : **IsPartialMatch**

Return Type : Boolean

Arguments : String

Visibility : Public

Context : Attempts to find approximate matches for an input URL against the cache entry. It also constructs a partial-match option presenting file.

URLRequest	
- toSEND	String
- ptrToClient	iBOSClientHandler
- filename	String
- URL	String
+ URLRequest(String, String, Integer)	
+ run (Void)	Void

**Figure 56 URL Request Class**



**CLASS NAME** : **URLRequest**  
*Inherits* : java.lang.Thread  
*Inherited by* : None

#### **DATA MEMBERS**

**Name** : **toSEND**  
**Type** : String  
**Visibility** : Private  
**Context** : This is an array of strings that holds the parameters to be sent over to the remote web-server.

**Name** : **ptrToClient**  
**Type** : iBOSClientHandler  
**Visibility** : Private  
**Context** : This is a pointer to a parent object of type iBOSClientHandler. It is used to make a callback at the calling object.

**Name** : **filename**  
**Type** : String  
**Visibility** : Private  
**Context** : Stores the filename fetched from the cache object. This is the file to which the HTTP client must write the contents fetched from the web-server.

**Name** : **URL**  
**Type** : String  
**Visibility** : Private  
**Context** : The URL string holds the URL for which the request is being made.

#### **MEMBER FUNCTIONS**

**Name** : **URLRequest**  
**Return Type** : Not Applicable  
**Arguments** : 1.) String  
2.) String  
3.) Integer  
**Visibility** : Public  
**Context** : This is the parameterized constructor of the class.

**Name** : **run**  
**Return Type** : Void  
**Arguments** : None  
**Visibility** : Public

Context : This is the thread's main run method that characterizes an execution thread.