

8.3.2 Object Diagrams

8.3.2.1 *Overview*

The following object diagrams describe the states of the system objects at various points of time. The context in which each object exists is described prior to the object. To suit the application, some extensions to the UML object diagram notation have been made, and documented. The object-diagrams in this document only describe the states and not the relationships among the objects.

8.3.2.2 *Notation Extensions*

The following notation is used apart from the standard object diagram notation:

1. An arrow from one field of an object to another object implies that the field holds a pointer to the object.

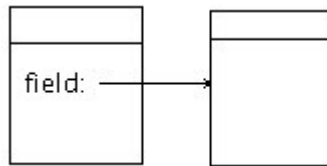


Figure 57. Pointers in Object diagrams

2. A folder-like box represents a single namespace. All objects inside the namespace are depicted skeletally – meaning that the members of the object are not depicted, in the namespace.

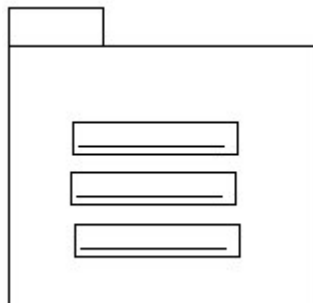


Figure 58. Namespaces

3. A simple dotted box represents a collection of unrelated objects. These objects are only related by the fact that they are communicated by some common object.

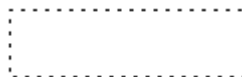


Figure 59. Unrelated object collection

Context: *Objects created at system startup*

These objects are created as soon as the system is started up. These objects are static and are not destroyed between page transitions.



Figure 60. Startup time objects

The Browser object encapsulates the entire functionality of the web-browsing system. It is, in fact, the skeleton for the browser.

The GlobalNamespace is a collection of objects that are modified (not destroyed) with page transitions. The GlobalNamespace objects are created when the system starts up, and are depicted in the next object diagram.

Context: *The global namespace objects*

These objects are skeletal in nature, because their members vary with the input to the browser. They are treated in detail in the next few diagrams.

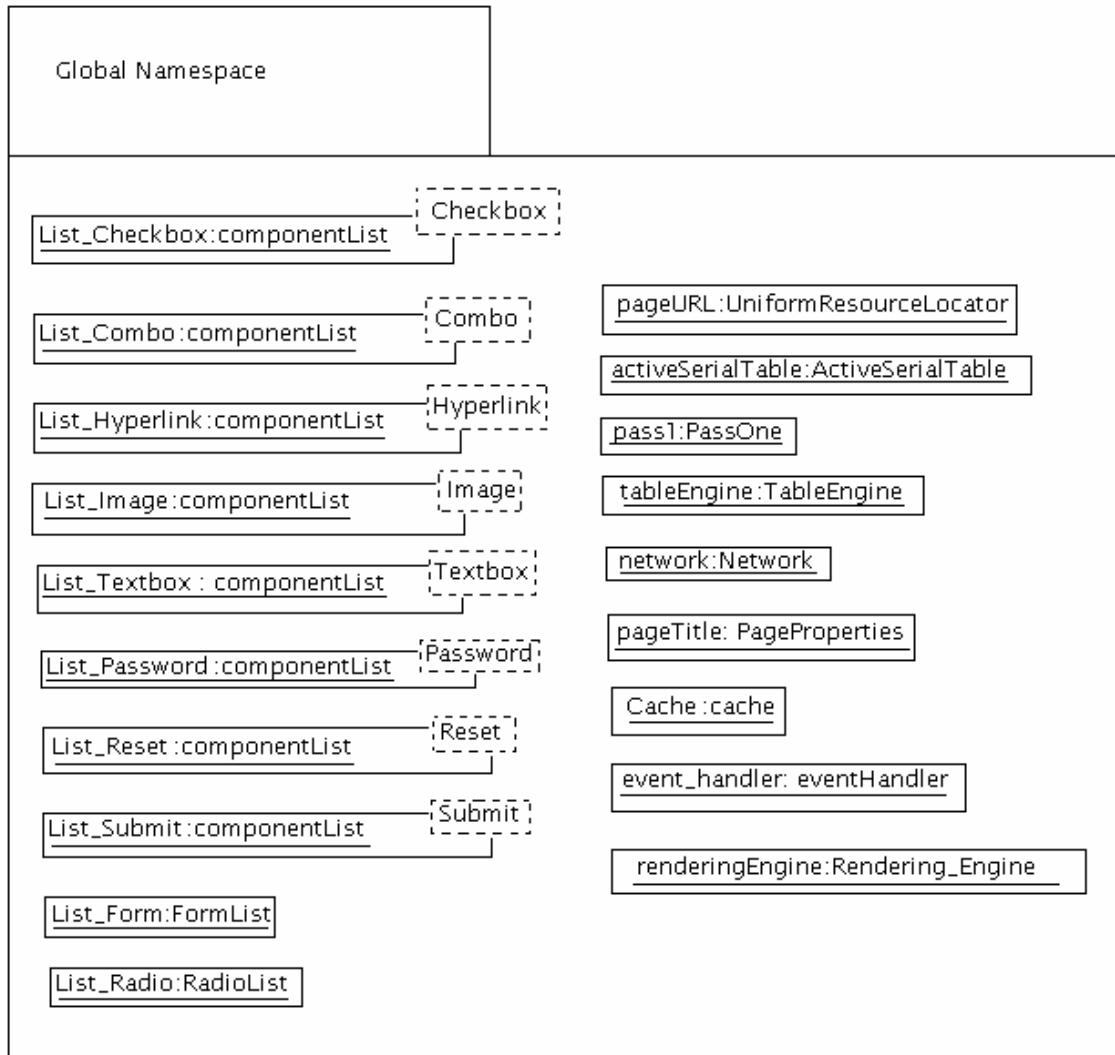


Figure 61 Global namespace objects

Context: *Objects created during the first pass*

The following object diagrams are created with respect to certain input HTML files. The idea is that for each of the input files, the object diagrams are different. So, prior to depicting the object diagram, the corresponding input file is provided. However, some of the objects are created for all pages. Thus, the states for those objects are given without their respective inputs.

- i) Active Serial Table
- ii) Forms
- iii) Checkboxes
- iv) Combo boxes
- v) Hyperlinks
- vi) Images
- vii) Password boxes
- viii) Radio boxes
- ix) Textboxes

i) Active Serial Table

Input HTML File

```
<html>
  <head>
    <title>
      Title
    </title>
  </head>
  <body>
    Page testing simple checkboxes<br>
    <hr> Check your options
    Harry <input type="Checkbox" name="c1"><br>
    Hermione <input type="Checkbox" name="c2"><br>
    Ron <input type="Checkbox" name="c3"><br>
  </body>
</html>
```

Object diagram – Active Serial Table

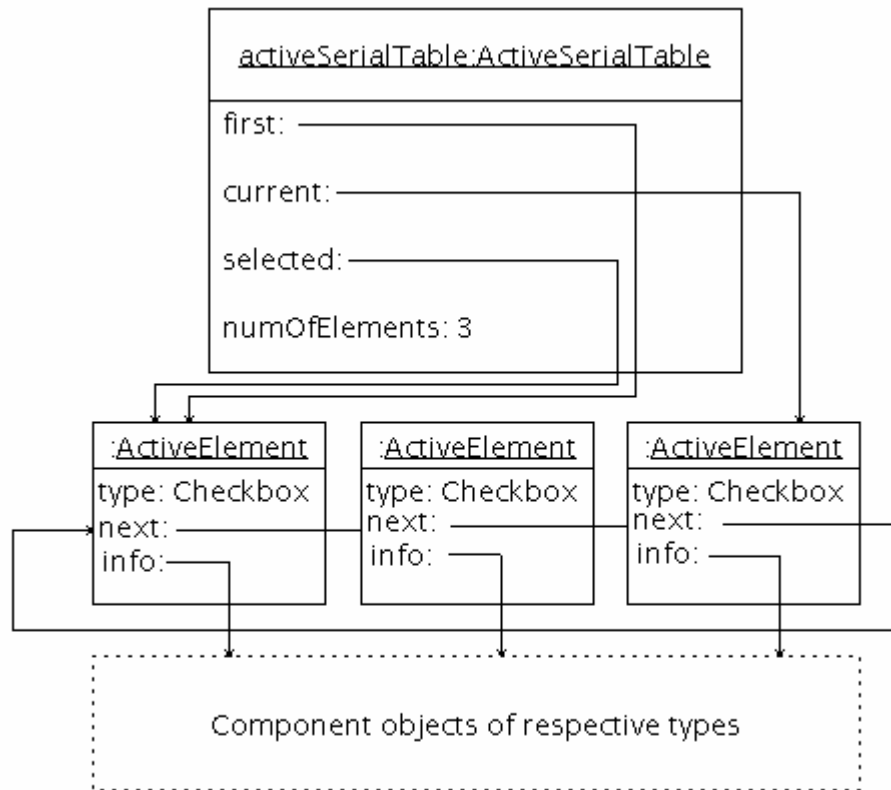


Figure 62 Active Serial Table

ii) Forms

Input HTML file

```
<html>
  <head>
    <title>
      Title
    </title>
  </head>
  <body>
    <form name='Form1' method='POST' action='submit.jsp'>
      Enter your name <input type='textbox' name='saa'><br>
      Want mail? <input type='checkbox' name='che' checked>
      <br>
      Want to subscribe to something?
      <input type='checkbox' name='subscribe'>
    </form>
  </body>
</html>
```

Object diagram: Forms – FormList and Form objects

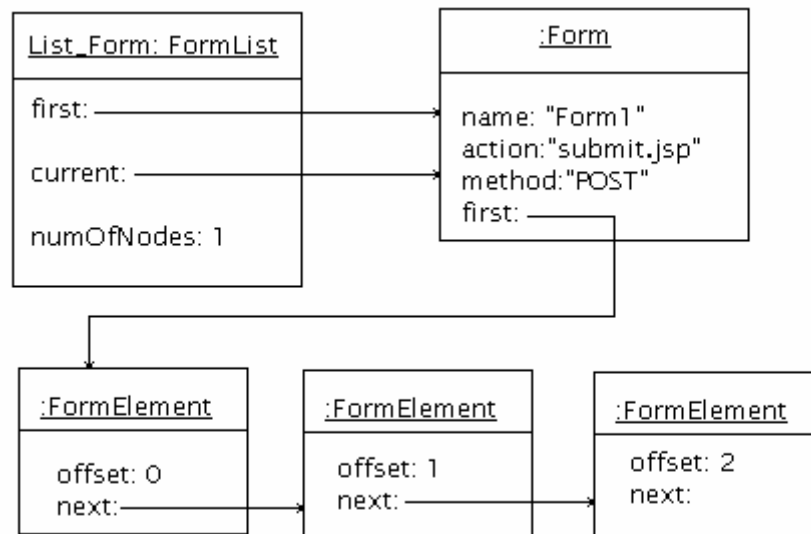


Figure 63 Form objects

iii) Checkboxes

Input HTML file

```
<html>
  <head>
    <title>
      Title
    </title>
  </head>
  <body>
    What all is Harry really? <br>
    Is he a wizard?
    <input type="Checkbox" name="IsWizard" value="Wizard" checked>
    Is he a programmer?
    <input type="Checkbox" name="IsProgrammer" value="Programmer"
  >
  </body>
</html>
```

Object diagram: List_Checkbox and Checkbox objects

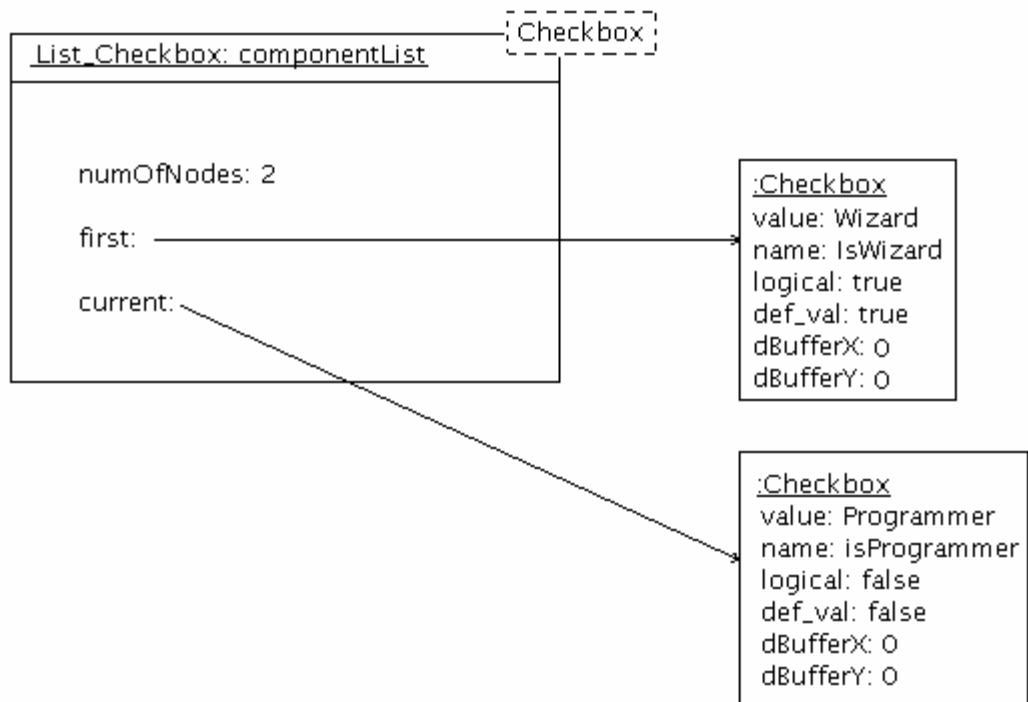


Figure 64 Checkbox Objects

iv) Combo boxes

Input HTML file

```
<html>
  <head>
    <title>
      Title
    </title>
  </head>
  <body>
    Select your absolute favourite character from the book "The
    Philosopher's stone"
    <select>
      <option value='Harry'> Harry </option>
      <option value='Ronald' selected> Ron </option>
      <option value='Hermione'> Hermione </option>
    </body>
  </html>
```

Object diagram: List_Combo, Combo and Combo_option objects

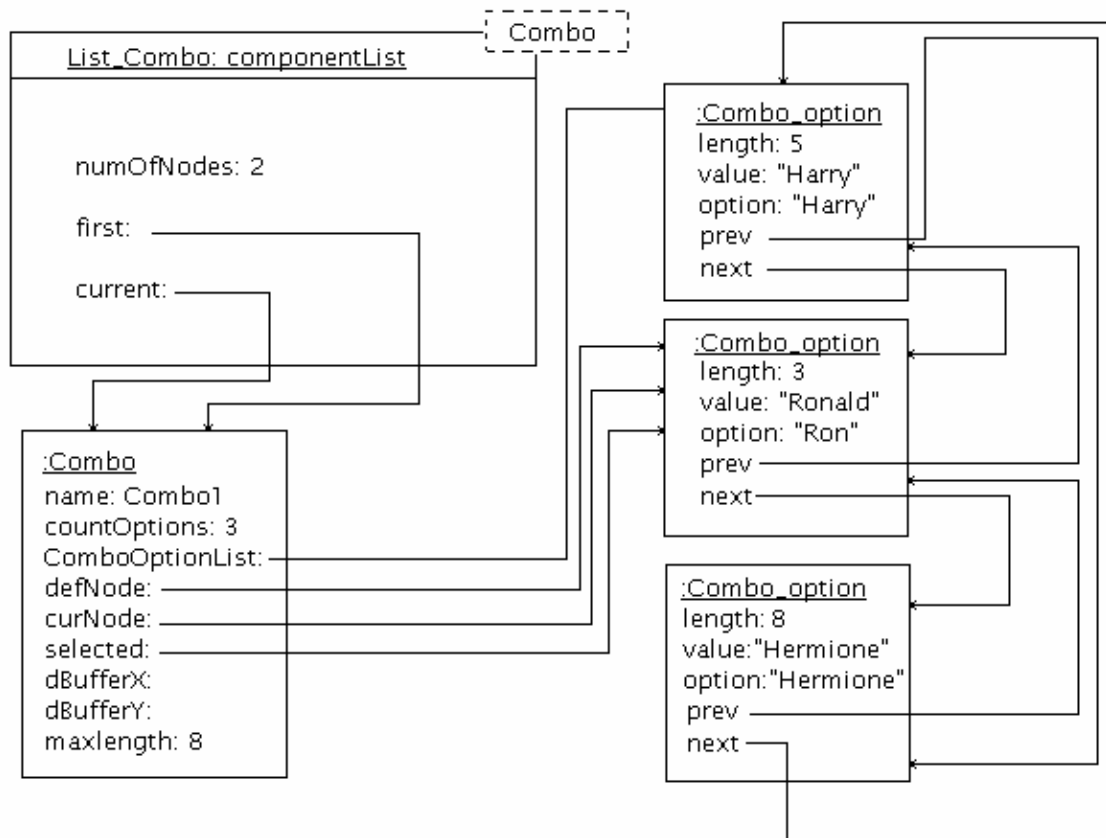


Figure 65 Combo Objects

v) Hyperlinks

Input HTML file

```
<html>
  <head>
    <title>
    </title>
  </head>
  <body>
    Search using the google search engine <br>
    <a href='www.google.com'> Google search </a>
    <br>
    Search the students web-site <br>
    <a href='students.jsp'> Students </a>
  </body>
</html>
```

Object diagram: List_Hyperlink and Hyperlink objects

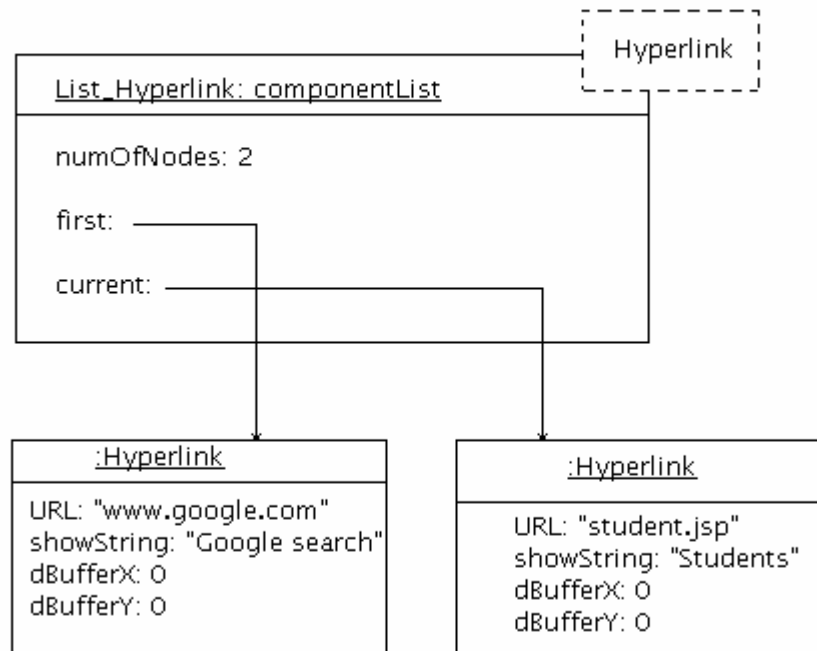


Figure 66 Hyperlink objects

vi) Images

Input HTML file

```
<html>
  <head>
    <title>
      Title
    </title>
  </head>
  <body>
    The first image supplies a normal alternative text field <br>.
    <br>

    The second image does not supply any alternative text<br>
    <br>

    And the third image tag is also a hyperlink tag
    <a href="www.google.com">
      
    </a>

  </body>
</html>
```

Object diagram: List_Image and Image objects

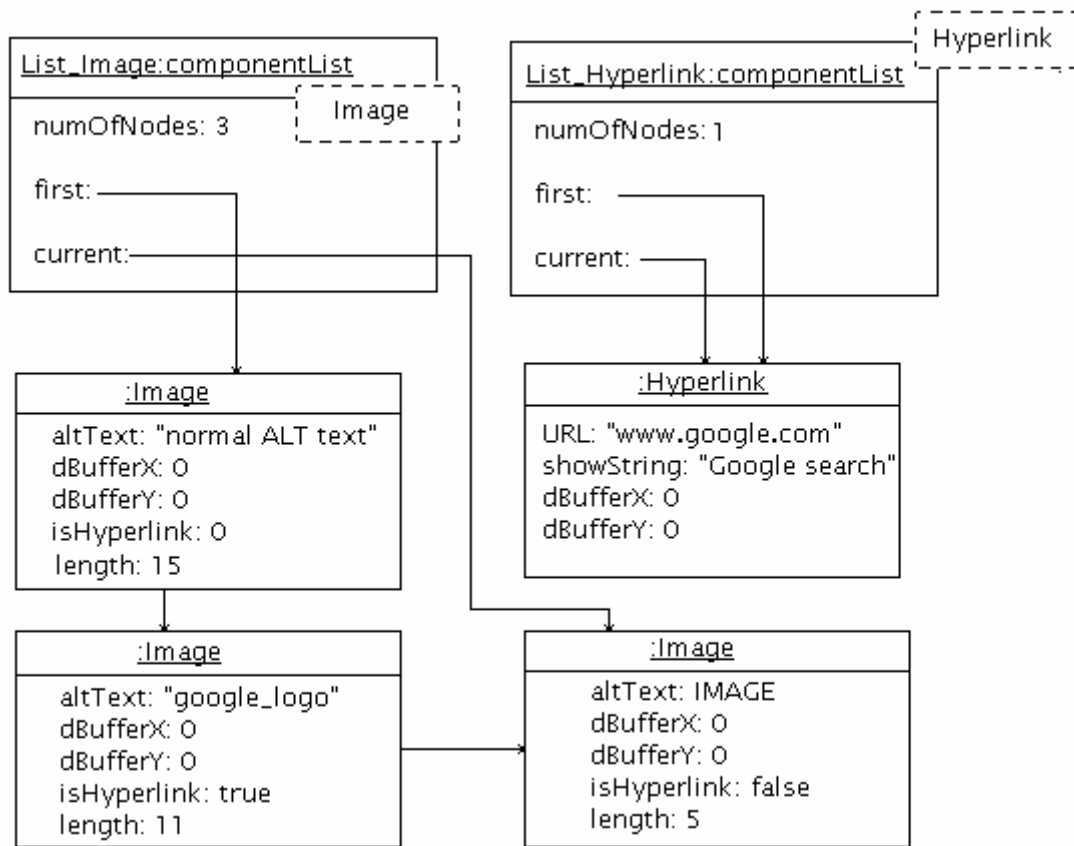


Figure 67 Image objects

vii) Password objects

Input HTML file

```
<html>
  <head>
    <title>
      Title
    </title>
  </head>
  <body>
    Enter your password here <input type="password" name="textbox1">
    <br>
    Re-enter password here <input type="password" name="username">
    <br>
  </body>
</html>
```

Object diagrams: List_Password and Password objects

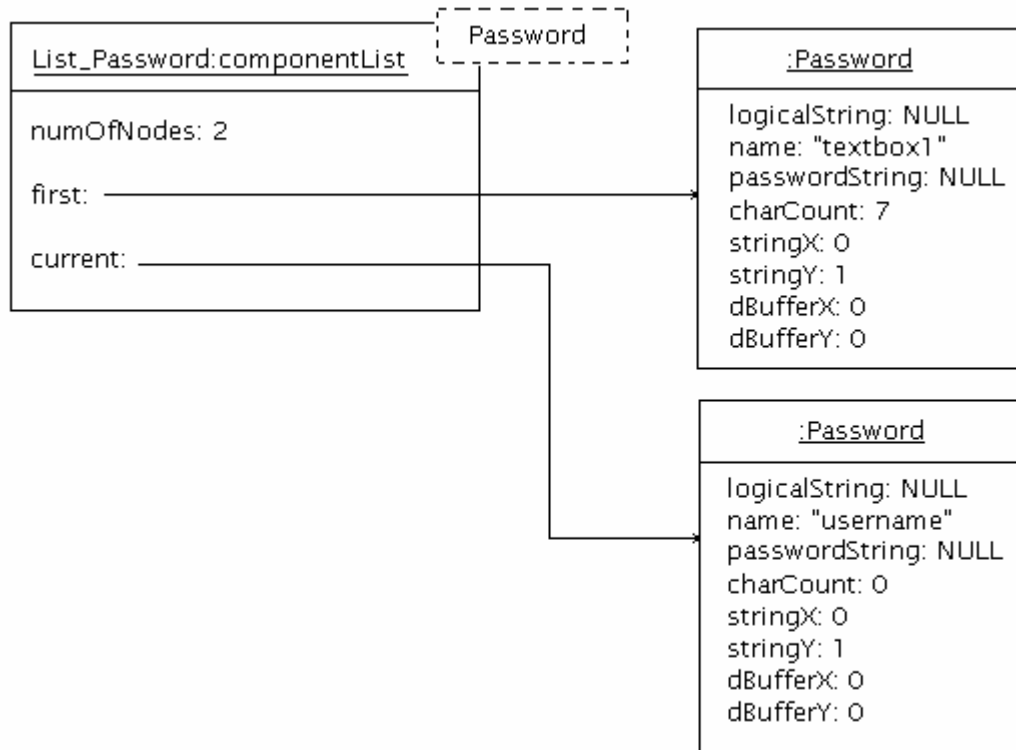


Figure 68 Password objects

viii) Radio boxes

Input HTML file

```
<html>
  <head>
    <title>
      Title
    </title>
  </head>
  <body>
    Test your Harry Potter knowledge: - <br>
    In the "Order of the Phoenix" book, who kills Sirius Black:
    <br>a. Lord Voldemort
      <input type="radio" name="radio1" value="Voldemort">

    <br>b. Bellatrix Lestrange
      <input type="radio" name="radio1" value="BELLA">

    <br>c. Harry Potter
      <input type="radio" name="radio1" value="HARRY">

  </body>
</html>
```

Object diagram: List_Radio, Radio and RadioListNode objects

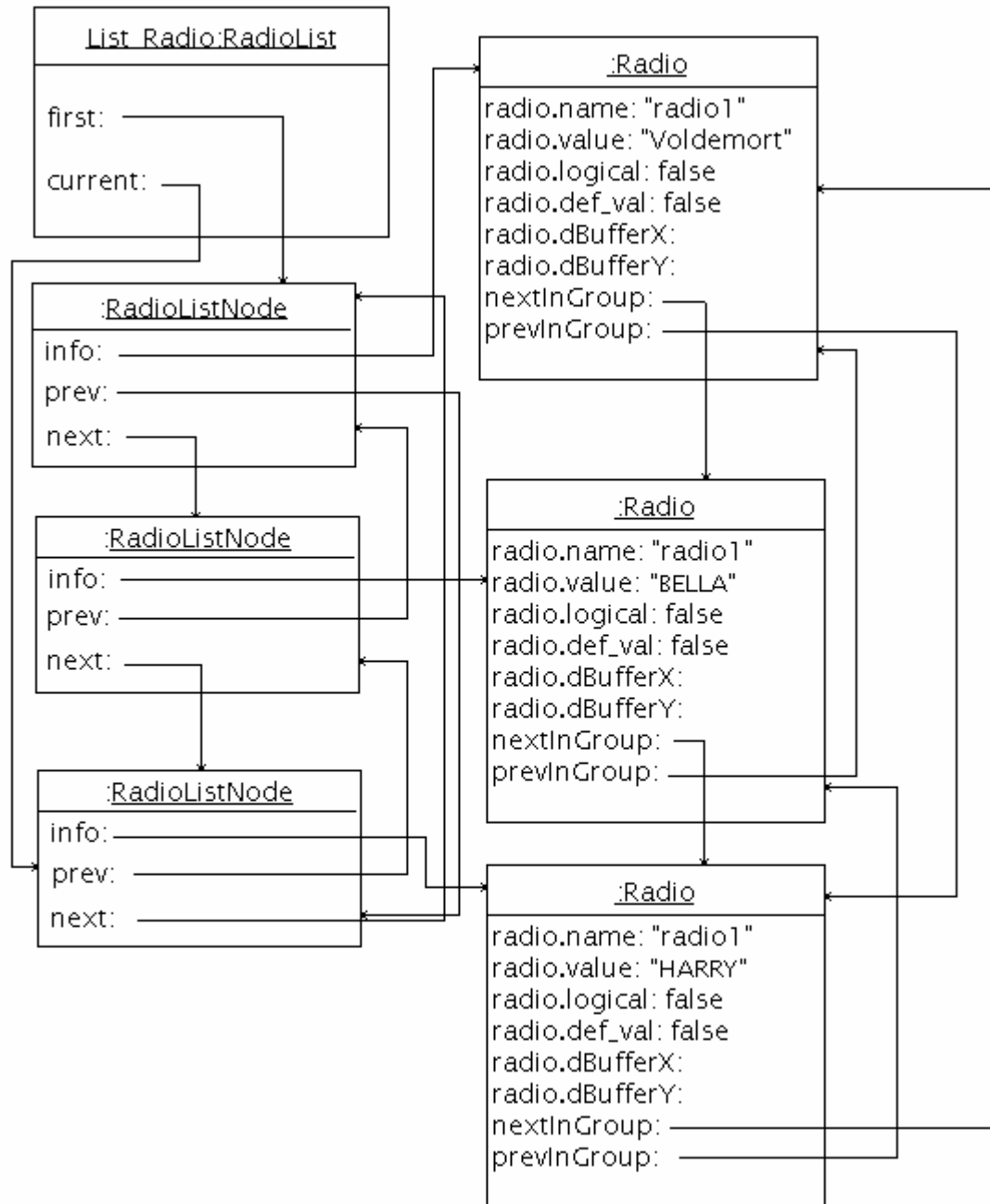


Figure 69 Radio objects

ix) Textboxes

Input HTML file

```
<html>
  <head>
    <title>
      Title
    </title>
  </head>
  <body>
    Any messages from your side to Harry, Ron, 'ermione, and Dumbledore?
    <input type="textbox" name="textbox1" value="default"> <br>
    Anyways, do not forget ur username<br>
    <input type="textbox" name="username">
  </body>
</html>
```

Object diagram: List_Textbox and Textbox objects

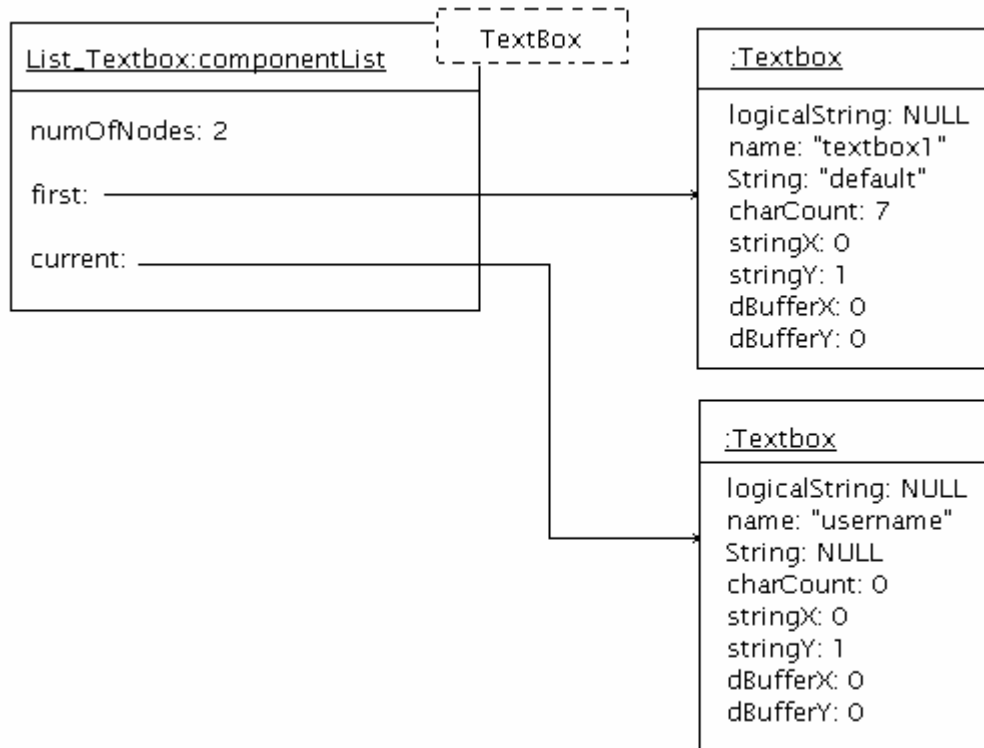


Figure 70 Textbox objects

Context: *Objects create in Pass 2*

The following object diagrams depict objects created in pass-2. This pass resolves the table-layout issue; there are two main types of objects in this pass – the table-engine and the table-object.

Input HTML file

```
<html>
  <head>
    <title>
      Title
    </title>
  </head>
  <body>
    <table>
      <tr>
        <td>
          Table cell 1, 1
        </td>
        <td>
          Table cell 1, 2
        </td>
      </tr>
      <tr>
        <td>
          Table cell 2, 1
        </td>
        <td>
          Table cell 2, 2
        </td>
      </tr>
    </table>
  </body>
</html>
```

Object diagram: table_engine and table objects

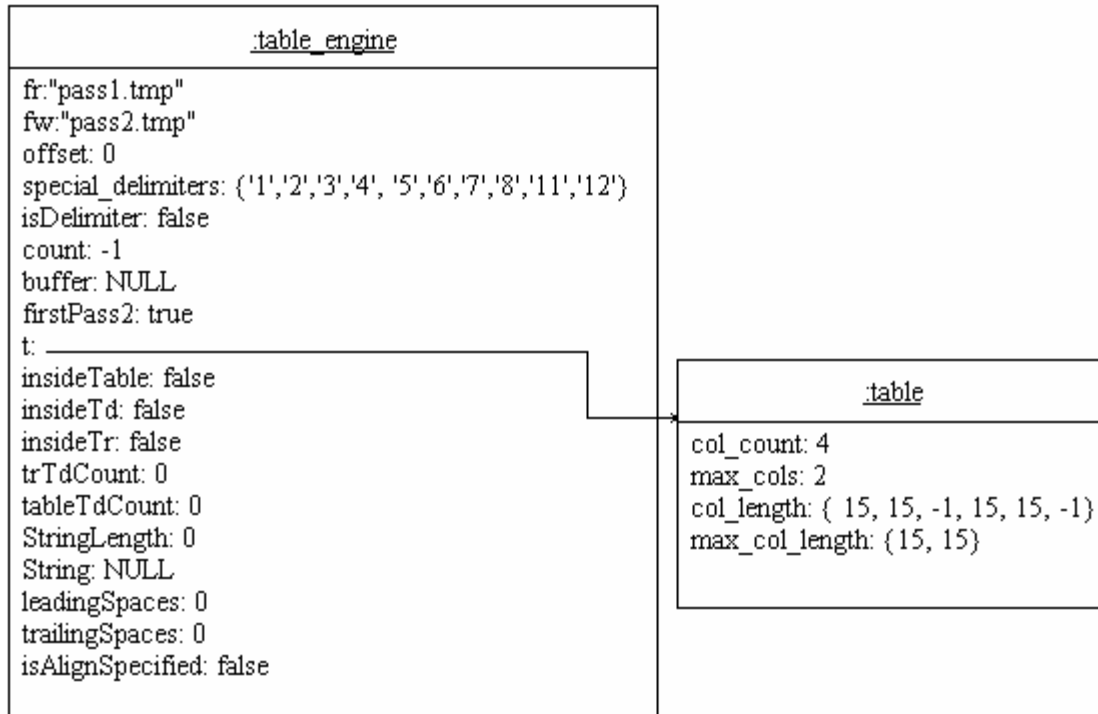


Figure 71 Table objects

Context: *Objects created/modified during and after the third pass*

In this phase, the different lists (ordered and unordered), and other formatting tags are resolved. Also, after the pass is done with, the text directly appears in the browser window. The main objects that are created/modified at this stage is of the type `Rendering_Engine` and a `list_Stack`.

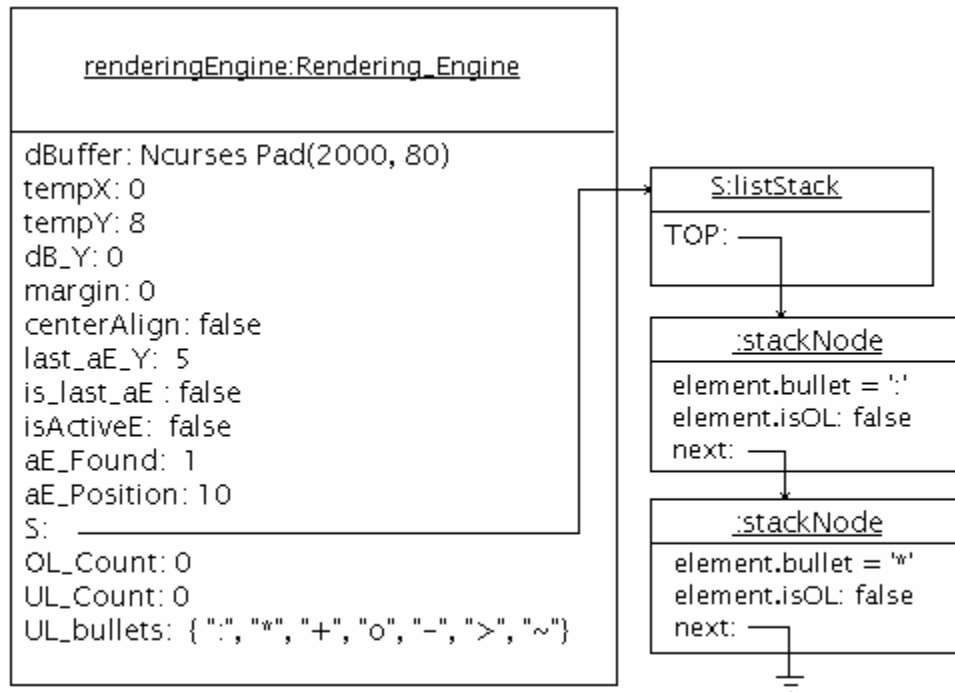


Figure 72 Rendering Engine objects

Context: *Proxy server objects*

The following objects will be created when a simple GET request is sent to the proxy server for a file *abc.html*. There are two sides of the proxy server – the server side that listens to iBOS client requests and the client side that fetches files from the Internet.

Object diagram: The server-part objects:

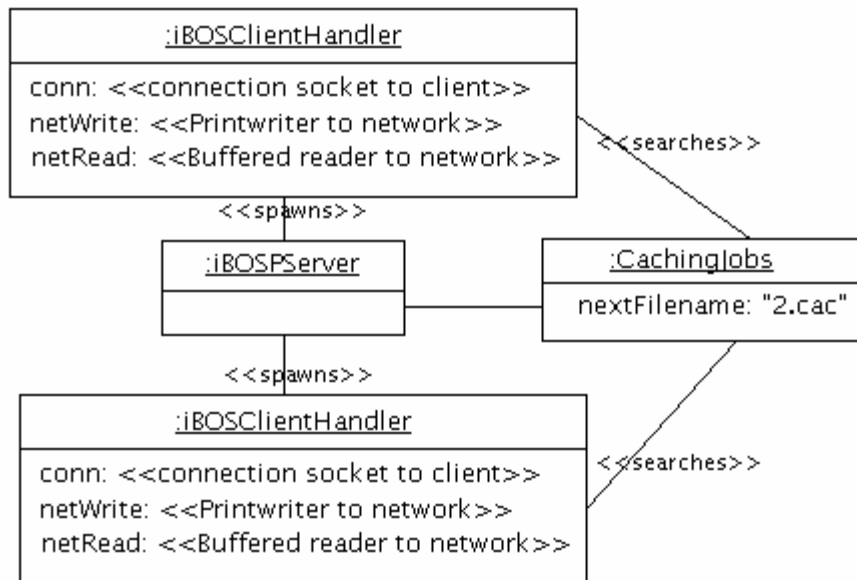


Figure 73 Proxy - Server side Objects

Object diagram: Proxy client objects

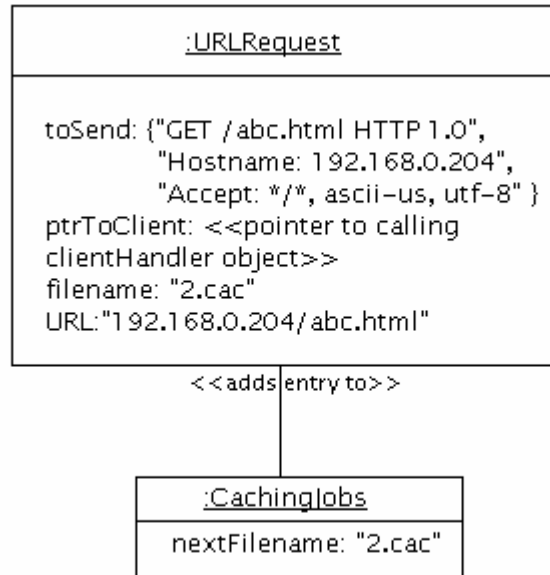


Figure 74 URL Request