CS509 Course Assignment

Objective:

Design a suite of components, office509, to perform basic functionality required by Information Management Systems.

Executive Summary:

There are five components planned for Office509:

- **Renderer** is able to display a string with embedded formatting commands
- **Spreadsheet** enables user to construct complex relationships among data items
- **Database** provides interface for adding, deleting, and querying information
- **Presentation** enables user to create presentations from complex data sources
- **Repository** provides persistent storage of information

There will be a matrix containing the assignment of students to these components. Each student will analyse, design, and implement one of these components and produce a sample application shell to verify that the component works as required. Each row of the matrix defines an informal “Group” of students. As each component is completed, the students in a “Group” will be responsible for integrating their component with the other components. In this way, each component will be used at least once by some other student. There is a set of standard collaborative scenarios (Listed at the end of this document) from which a student selects one to pursue.

Details

The rest of this document contains details on each of the components. Note that the first task (which should last one week) is to decide upon the requirements for each component. It is expected that students will discuss the requirements with other students working on the same component. Don't be overly ambitious. You should construct a full list of the desired requirements together with a documented set of sample scenarios that will be used to verify that the expected requirements are complete.
**Word Renderer:**

**Primary Purpose: To render a string of information to a scrollable GUI entity.**

The information string is formatted using textual elements similar to HTML. The core part of the component must understand the following commands:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;a href=&quot;tag.html&quot;&gt;…&lt;/a&gt;</code></td>
<td>characters in between become “active” and when the user clicks on these characters, an event is announced with “tag.html” as information.</td>
</tr>
<tr>
<td><code>&lt;b&gt;…&lt;/b&gt;</code></td>
<td>characters in between are shown in bold face</td>
</tr>
<tr>
<td><code>&lt;i&gt;…&lt;/i&gt;</code></td>
<td>characters in between are shown in italics face.</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;file.gif&quot;&gt;</code></td>
<td>show image stored in file.gif</td>
</tr>
<tr>
<td><code>&lt;p&gt;</code></td>
<td>Move to new line</td>
</tr>
</tbody>
</table>

**Figure 1: Basic Formatting Commands**

This set is a minimal starting point that you must have working. If you choose to add any more formatting commands, you must ensure that these are fully functioning first. For example, the graphical rendering of the string “This is a `<b>`test of `<i>`how</i>` `<b>`well you understand` `<p>`the format” is shown in Figure.

```
This is a
**test of how** well you understand
the format
```

**Figure 2: Sample Rendering**

The output of the component will be a region of the screen (most likely a `java.awt.Canvas`) that contains the rendered information and allows the user to scroll vertically to show the entire information in the provided area.

**Advanced:**

More advanced concepts can be introduced into the renderer, such as different fonts or tables, for example, but you must complete the core functionality first.
Spreadsheet:

*Primary Purpose:* To enable user to construct and compute complex relationships among data.

This component will be part of the assignment for HW2.
**Database:**

*Primary Purpose: Provides interface for adding, deleting, and querying information from a single database table*

This component will implement a small subset of functionality found in database systems. A database table is the basic element of a database. A table has a set of fields, one of which is the primary key for the table. The database ensures that the values for the primary key are unique. For example, a database could represent the table "ID, First Name, Last Name" where ID is a unique identifier. It is possible to add an entry to the database, such as "add id=123 First Name=George Last Name=Heineman". Also it is possible to delete an entry "delete id=123" which should delete a single entry (if it exists); one can also delete based on other fields, such as "delete First Name="George", which will delete all entries in the database whose First name is George.

The database must be fully parameterizable so it can store any size table, with any number of fields.

To query a database you should construct a query, such as "query First Name=George". The database executes this query and returns an Enumeration. This enumeration contains entries in the database and has two basic methods, hasMoreElements and nextElement. These methods are used to extract the information from the query.

You will likely use a hashtable (java.util.Hashtable) to represent the fields for a database.

**Advanced**

More advanced concepts can be introduced into the database, such as allowing updates, but make sure that the core concepts are complete before initiating any advanced project.
Presentation:

*Primary Purpose: Enables user to create visual presentation from complex data sources*

The presentation component reads a description of a presentation and executes it. The presentation description will be formatted with presentation commands, describing the placement of objects on a virtual page. There should be a simple GUI interface allowing the user to step forward/backward between pages.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;text x=10 y=20&gt;…&lt;/text&gt;</code></td>
<td>Characters in between are placed on the screen at the given x,y position</td>
</tr>
<tr>
<td><code>&lt;rectangle x=10 y=20 w=5 h=7&gt;</code></td>
<td>Rectangle with given (w,h) dimensions drawn at (x,y)</td>
</tr>
<tr>
<td><code>&lt;line x1=10 y1=20 x2=80 y2=20&gt;</code></td>
<td>Line drawn between the given two points.</td>
</tr>
<tr>
<td><code>&lt;page&gt;</code></td>
<td>New page</td>
</tr>
</tbody>
</table>

Advanced:
A more advanced presentation component would provide a design mode where the user can create a presentation for later viewing. Before doing any advanced work on the component, however, you must ensure that the core functionality is complete.
**Repository:**

*Primary Purpose: To enable persistent reading and writing of arbitrary data*

You will note that the other components do not have the capacity to store information persistently. The repository component will provide this service. To be as generic as possible, the repository will store information using a unique string. Each resource is typed by a String. The strings contain the characters "a-zA-Z_" but the special character "/" is a hierarchical separator. For example, the resource "common/constants/pi" is a child (or belongs) to the resource "common/constants".

A client will first request to *open* a resource by name, in read/write/append mode; this request will return a `java.io.ObjectOutputStream` or `java.io.ObjectInputStream` object as appropriate. Clients can then read/write information, ultimately issuing a *close* request to the resource when complete.

Clients should be able to: retrieve a list of all top-level resources, retrieve a particular resource by full name, retrieve information about a resource (size, last access time, last modified time, resource type).

It must be possible for a client to determine what the type of information is for a particular repository resource.

*Advanced:*

A more advanced repository component would provide a hierarchical structure for the repository that would enable browsing of the information stored in the repository.
Sample Collaborative Scenarios:
S1: Spreadsheet has information in cells that are extracted “live” from a database.
S2: Renderer includes information from one of a spreadsheet’s cells
S3: Renderer includes information from a database
S4: Presentation displays a Renderer component in a presentation
S5: Presentation displays a spreadsheet table
S6: Spreadsheet has information in cells extracted from a presentation
**Advanced Component:**

DATA: Database adaptor (for accessing information within a spreadsheet like a database)

This is responsible for bridging the gap between spreadsheets and databases.