Logical

Architecture Document

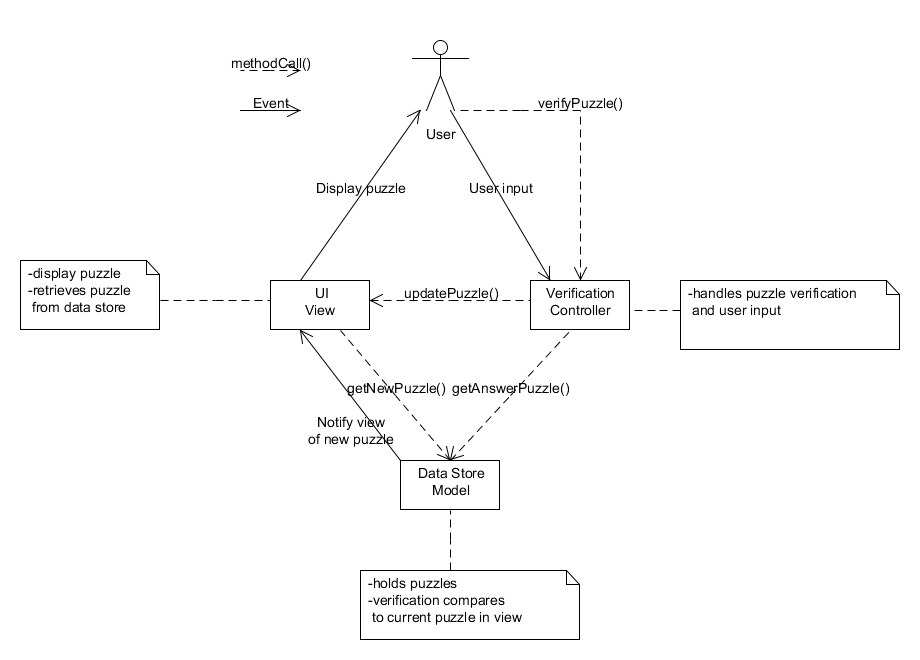
Lori Gildersleeve  
Cody Nelson  
Jennifer Martin

**Logical Architecture for a Simple Sudoku Game**

**Overview**

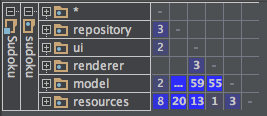
The architecture of the Sudoku game is separated into three different tiers based on the State-Logic-Display pattern. The 3-tier architecture is based on separating the data, logic, and user interface components. This pattern of architecture was chosen for its simple, flexible and maintainable nature. Also, the separation of duties into separate tiers allows for changes to be made independently, thereby allowing for greater ease in adaptability and scalability.

In the following figure (**Figure 1**), one can see the overarching view of how the application operates in the 3-tier environment.



**Figure 1**: Simple MVC diagram for Sudoku application

Here is a visual of the package layers making up the Sudoku architecture:



**Figure 2**: Packages within Sudoku Game

**Repository:**  responsible for housing the data layer of the application (aka state tier)

**UI:** responsible for housing the display tier

**Renderer:** responsible for the logic layer

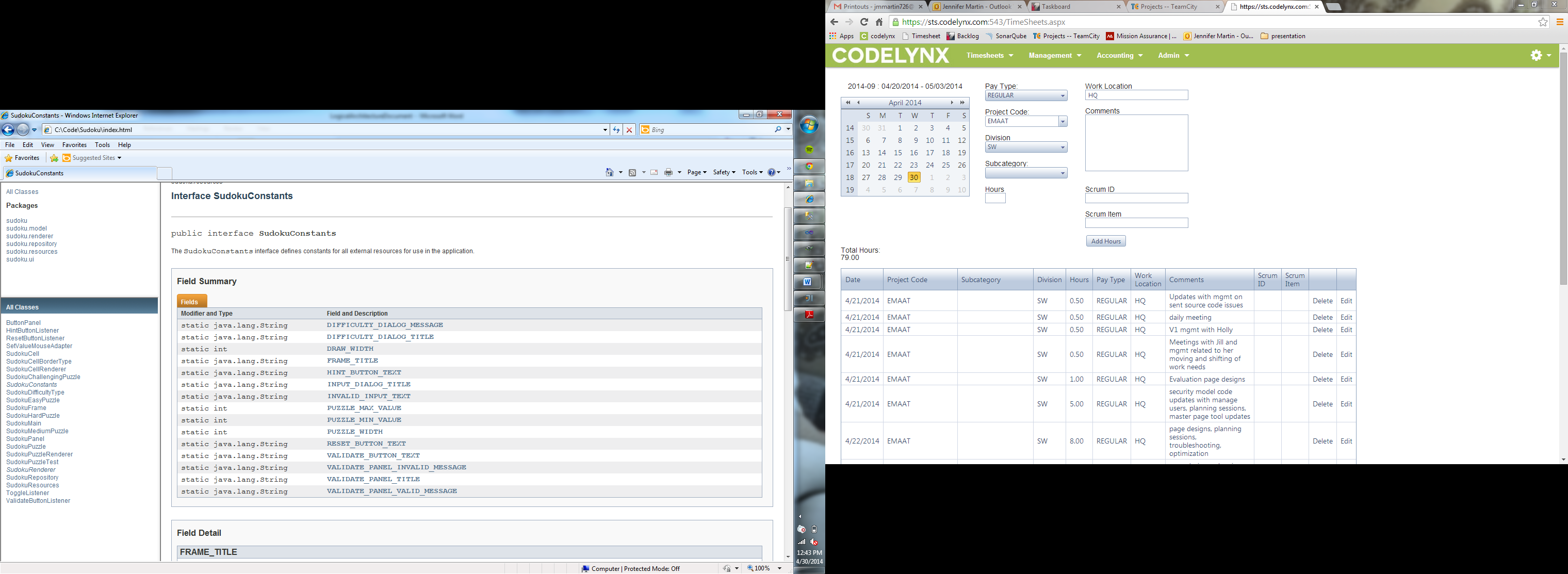
Both the **Model** and **Resources** packages contain methods and variables that are used in the other packages (e.g., constants, general UI settings, etc.).

**Sudoku Interfaces**

The application uses two interfaces:

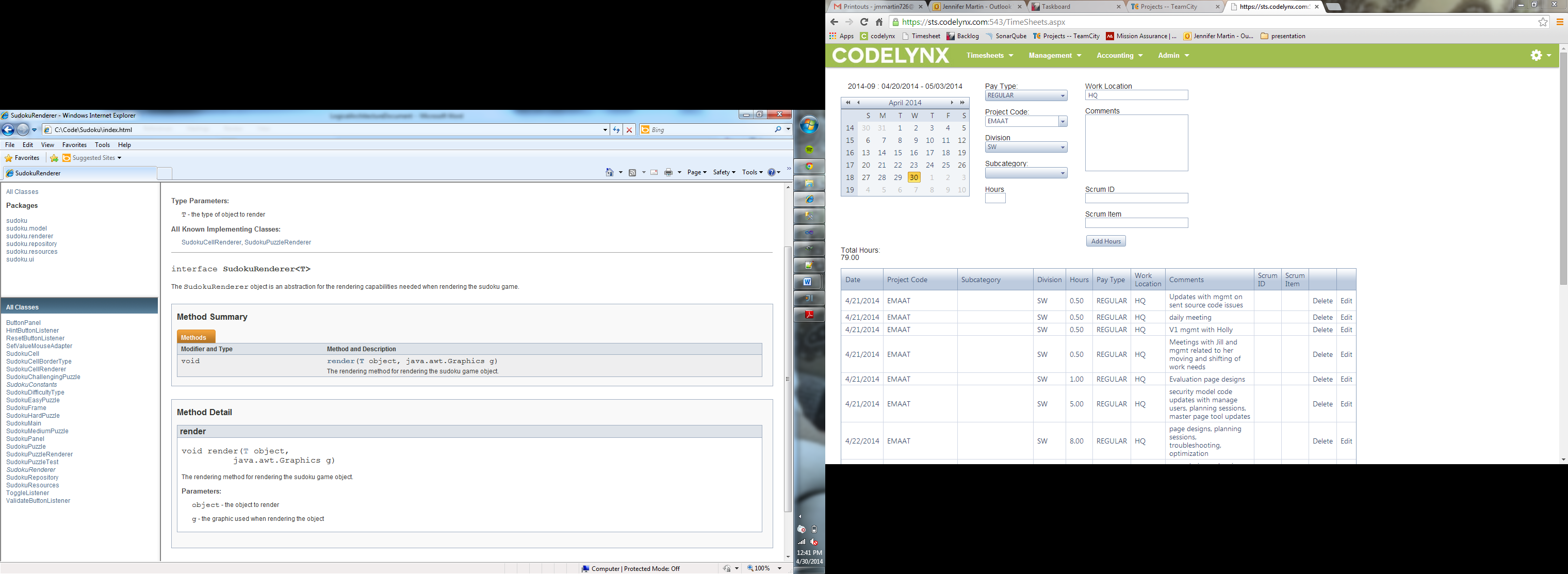
1. SudokuConstants: defines constants for all external resources in the application.

**Figure 3**: The list of constants

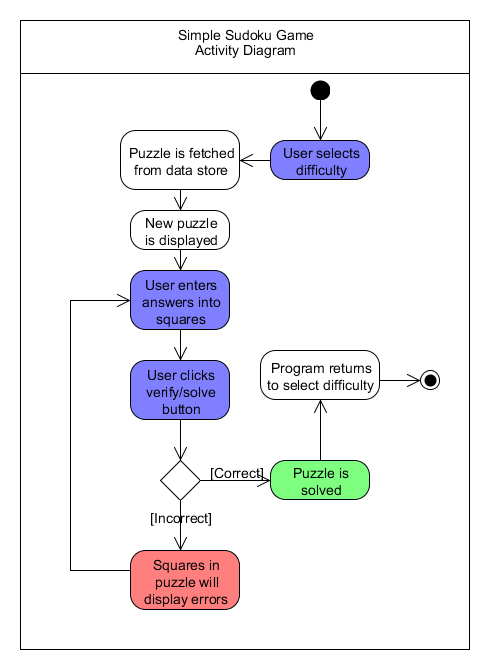


1. SudokuRenderer<T>: an abstraction for the rendering capabilities needed when rendering the Sudoku game.
   * 2 classes implement this interface: SudokuCellRenderer and SudokuPuzzleRenderer
   * 1 method is contained within this interface

**Figure 4**: The method for rendering the Sudoku game



**Activity Diagram**

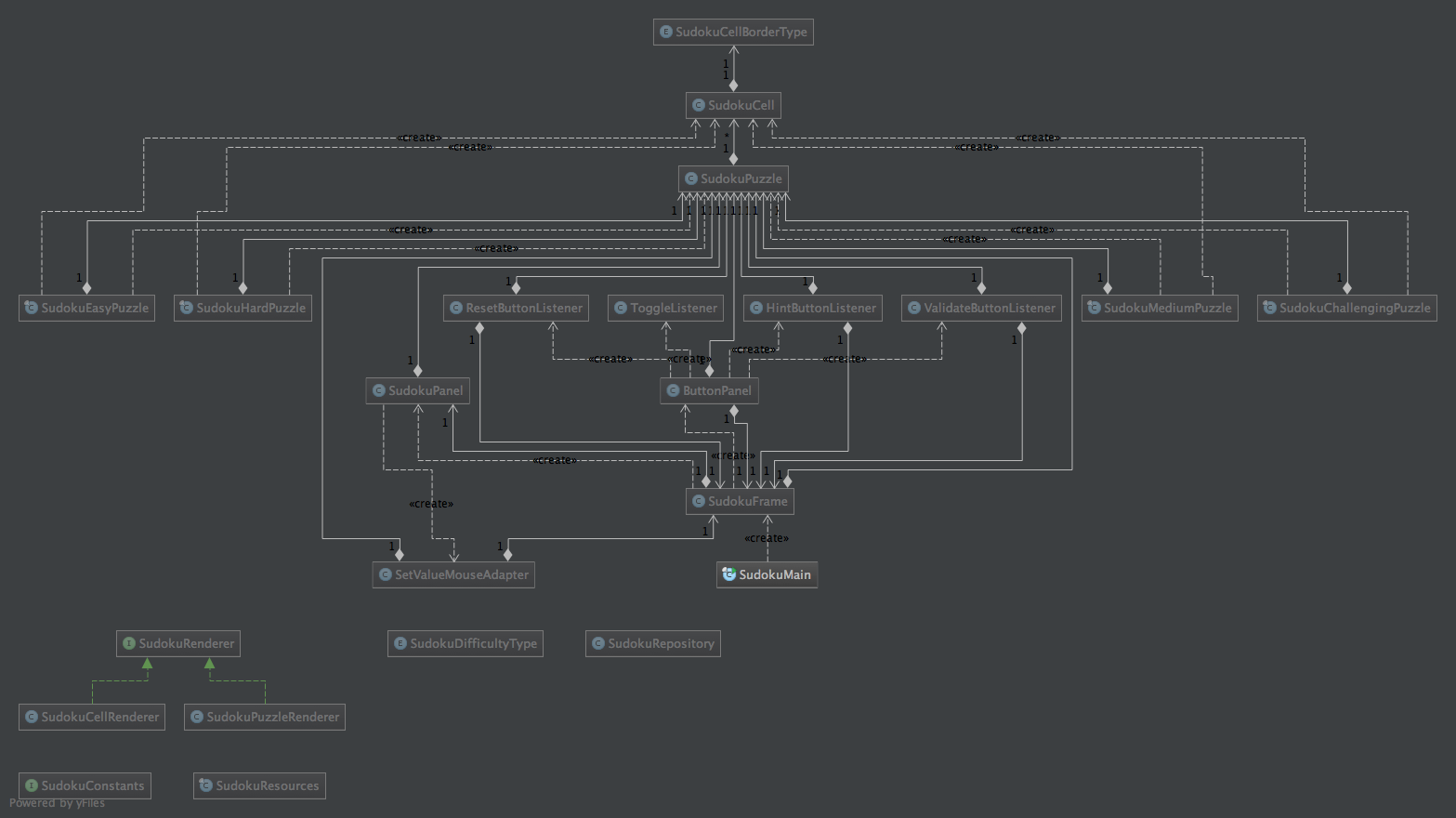
The basic flow of the application is as follows in the activity diagram:

**Figure 5**: Activity diagram for Sudoku application

The connectors used in the application are listeners and, in some cases, data access connectors.

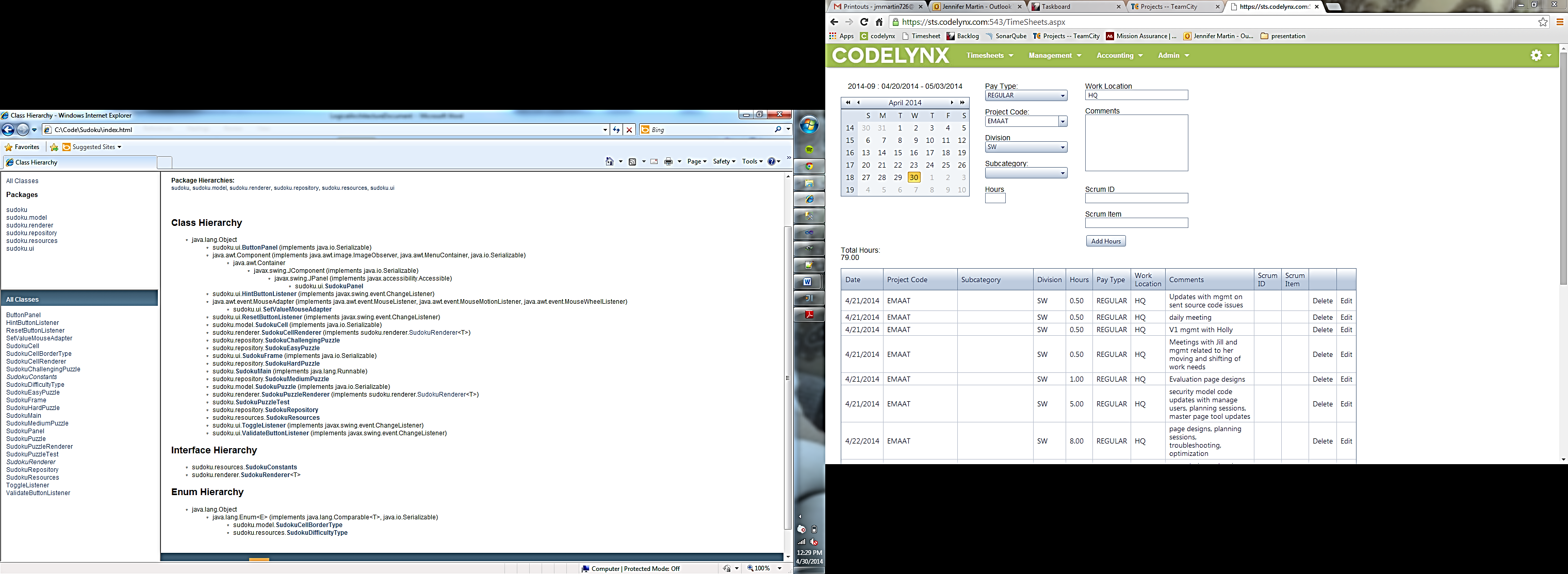
**Class Diagram**

Below is a UML class diagram (**Figure 7**) that shows the usages, creations, and relations of the object model for the Sudoku application. The UML diagram contains the class hierarchy of the program, defining all the major pieces of the architecture, including external components.



**Figure 6**: UML class diagram for Sudoku application

**Class Hierarchy Tree**



**Figure 7**: Class hierarchy tree for Sudoku application