Interface Control Document

Lori Gildersleeve  
Jennifer Martin  
Cody Nelson**Interface Control for a Simple Sudoku Game**

**Overview of Connectors**

The Sudoku game will have three types of connectors:

1. event connectors to handle user input
2. procedure connectors to handle logic
3. data access connectors to handle state/model requests

**User Interface**

The event connectors will link to the graphical user interface. They will handle input events from the user, as well as connecting the GUI to the verification logic via the Verify/Solve button.

Pre-conditions: The invocation of the GUI event connectors is entirely user controlled and is not invoked outside of user input or interaction.

Post-conditions: Any interactions the user has with the GUI will result in some sort of user interface updates, while always leaving the GUI in a state of usability. All interactions with the user will guarantee that the user interface will be actively maintained, updated, and usable no matter logic or state changes.

**SudokuMain.java**  
Pre-condition: none  
Post-condition: end of program

**SudokuFrame.java**  
Pre-condition: receives valid puzzle from player’s difficulty choice  
Post-condition: returns frame containing valid puzzle

**SudokuPanel.java**  
Pre-condition: receives valid frame and valid puzzle  
Post-condition: returns and displays puzzle for user interaction

**Rendering/Verification Logic**

The procedure connectors will handle calls within the verification logic component. This will be handled primarily via method calls.

Pre-conditions: Invocations of the rendering/verification logic happen from two specific conditions:

1. When the puzzle first loads, initial verification and rendering logic is applied to the puzzle to ensure that the initial state is displayed correctly for the user.
2. When the user interacts with the GUI and commits an action, rendering and/or verification logic will be invoked to handle the interactively invoked component control mechanism selected by the user.

Post-conditions: Any interactions or invocations to the rendering/verification logic will result in the new data being displayed for the user. If an error occurs during the rendering/verification, it will be logged and handled properly to ensure that the GUI is not left unusable, and the user can still interact with the system.

**SudokuPuzzle.java**  
Pre-condition: receives valid user input of a new value in a particular cell  
Post-condition: returns updated display of puzzle

Pre-condition: receives that user selected reset button  
Post-condition: returns valid puzzle restored to initial state

Pre-condition: receives that user selected hint button  
Post-condition: returns possible values for each cell

Pre-condition: receives that user selected validate button  
Post-condition: returns validity of puzzle via a pop-up box and cell color changes

**Data Store Access**

The data access connectors will link the data model and data store to the GUI and verification logic.

Pre-conditions: Any interaction with a puzzle in the system will invoke calls to the data store, since it houses all active and inactive system puzzles that the user can access and play.

Post-conditions: The data modified or returned from the data store will not cause any faults, as it is simply a store, and it does not rely on logic to handle interactions.

**SudokuRepository.java**  
Pre-condition: receives valid player’s difficulty choice  
Post-condition: returns valid puzzle