Data Structure

Dailton Rabelo

Pat Ryan

**Requirements for Puzzle Game**

**Functionality:**

What will the system do?

* Create a frame with three four by four panels.
* Each panel will have 16 buttons, each a possible answer to the puzzle!
* There will be a fourth panel that includes clues for the puzzle.
* There will be a fifth panel holding the action buttons such as hint, clue, and reset.
* There will be a “Hint” button that when clicked will give the user one hint at a time!
* The user can click the hint button as many times as he or she wishes, but that may take the fun out of it!
* The “Hint” button will randomly select a correct button on each panel, each time it is clicked, until the puzzle is solved.
* The “Reset” button will reset the game.
* The “Check Solution” button will check if you have won the game. If yes then it will display a message announcing it and resetting the game. If not, it will say “Not yet” and let you continue.
* Include versatility. I.E. If needed, we could create implement a new logic puzzle with very little work. Just modifying the text file.

When will the system do it?

* The system will run when intended to by the user.
* Hints will be given purely by discretion of the user, and as many times as he or she wishes throughout the course of the game.
* The game frame and panels will be created upon compilation.
* If a new logic puzzle is wanted, we would implement it when asked.

What kind of computation or data transformation must be performed?

* In the file loader, it creates a map using 4 Array Lists. Three of them are the categories string and the fourth is an array List holding all the clues.
* In the puzzle maker class, we convert the string array lists in the map, into puzzle piece array lists to insert into the board.
* There will be three puzzles for each panel, each will be held in a PuzzlePiece array list.

**Data**

For both input and output what should be the format of the data?

* The file with the puzzle clues, and the puzzle itself will be in .txt format!
* The first three lines of the text file will hold the 3 boards.
* The format should be x1-x2-x3-x4-y1-y2-y3-y4-a1-a2-a3-a4
* X1 will be the first category on the first button then x2 will be the second. While y1 is the corresponding y value for x1. A1 to A4 is the solution index of the board. It will be the answer.
* After the three categories, the clues are then written into the text file. You can have as many as you want.

Must any data be retained for any period of time?

* User data must be retained for the exact length of time the game is played.
* At this point we have not found the need for a save game, score tracking, or record keeping.

**Usability:**

How easy should it be for user to understand and use the system?

* + - The system should be intuitive for all users!
    - Simplicity in structure, and familiar format make this easy to accomplish!
    - The game will come with hints and clues, to be sure that the user is able to have an easy and rewarding experience while using the software!
    - The game itself is designed in the fashion of a logic puzzle, and therefore the user should have at least a general understanding that puzzles can be difficult to solve, and to solve them is rewarding for many users. With that being said, the logic puzzle itself should be designed to challenge the user, and make them come back for more.

How difficult should it be for a user to misuse the system?

* + - The system is designed to be self reliant, and connectionless.
    - There should be no permissions needed to run the software, and the software should need nothing more than java.
    - The software will not be capable of compromising any machine, in any fashion.

**Reliability and availability:**

Must system detect and isolate faults?

* The system will recognize and show errors when the wrong file type is used.

How often should the system be backed up?

* The system will only require backing up after each patch, or change to the software itself.
* The system has no capability of compromising any other part of the machine as a whole, and does not have the functionality to hold data beyond a current game.

**Performance:**

Are there constraints on execution speed, response time, or throughput?

* + - There are no constraints on speed, as the program is lightweight and there is not much data to be processed throughout the course of the game.
    - Response time will be as expected in software such as this, as long as user is playing on a machine dated within the past decade.
    - There are no bottlenecks in the code, and one file to be read from, therefore I/O should not cause any response issues.

How much data will flow through the system?

* + - This puzzle in particular will have 3 panels of game buttons with the possibility of 16 possible answers.
    - The software itself is capable of handling much more, but that is case by case, and as noted, this software is lightweight.

How often will data be received or sent?

* + - Data will be sent and received every time a button is clicked.
    - The hint button, and each tile in each panel will send data for the software to evaluate, and how often data will be transmitted is based purely on when the user presses one of the buttons.

**Supportability:**

When and in what ways might the system be changed in the future?

* The software is designed to evolve as needed, quickly and with ease.
* Could implement save feature.
* Could implement a timer to give a score on how fast the game was completed.

How easy should it be to add features to the system?

* + - As noted, the software is designed to be compatible with many other puzzles of the same type.
    - The software should be able to easily add features within reason to the system as a whole.
    - Data is kept separate from the interface, and is therefore capable of being easily manipulated with reason to the current software.