Hari Devanathan (hd2zm) and Sai Peddy (ssp5en)

**Final Project Report**

**Algorithm:**

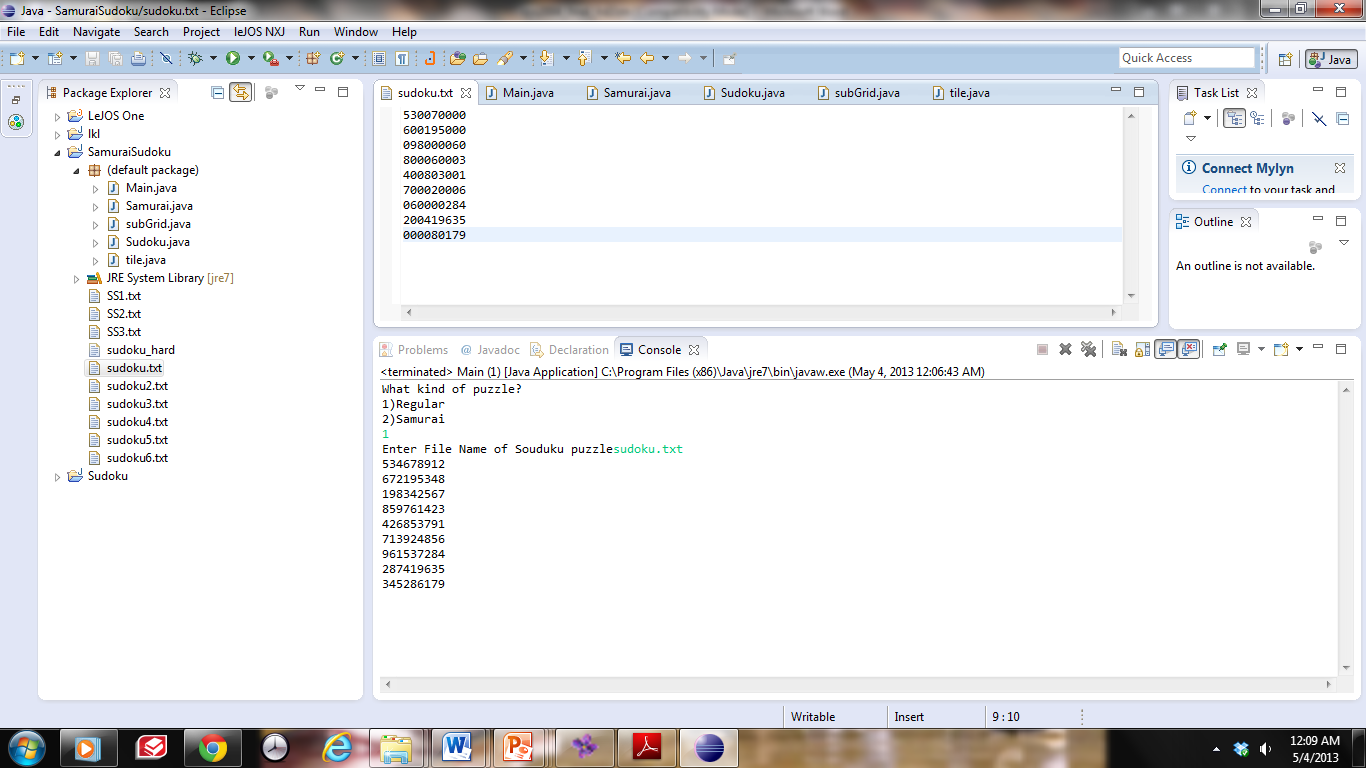
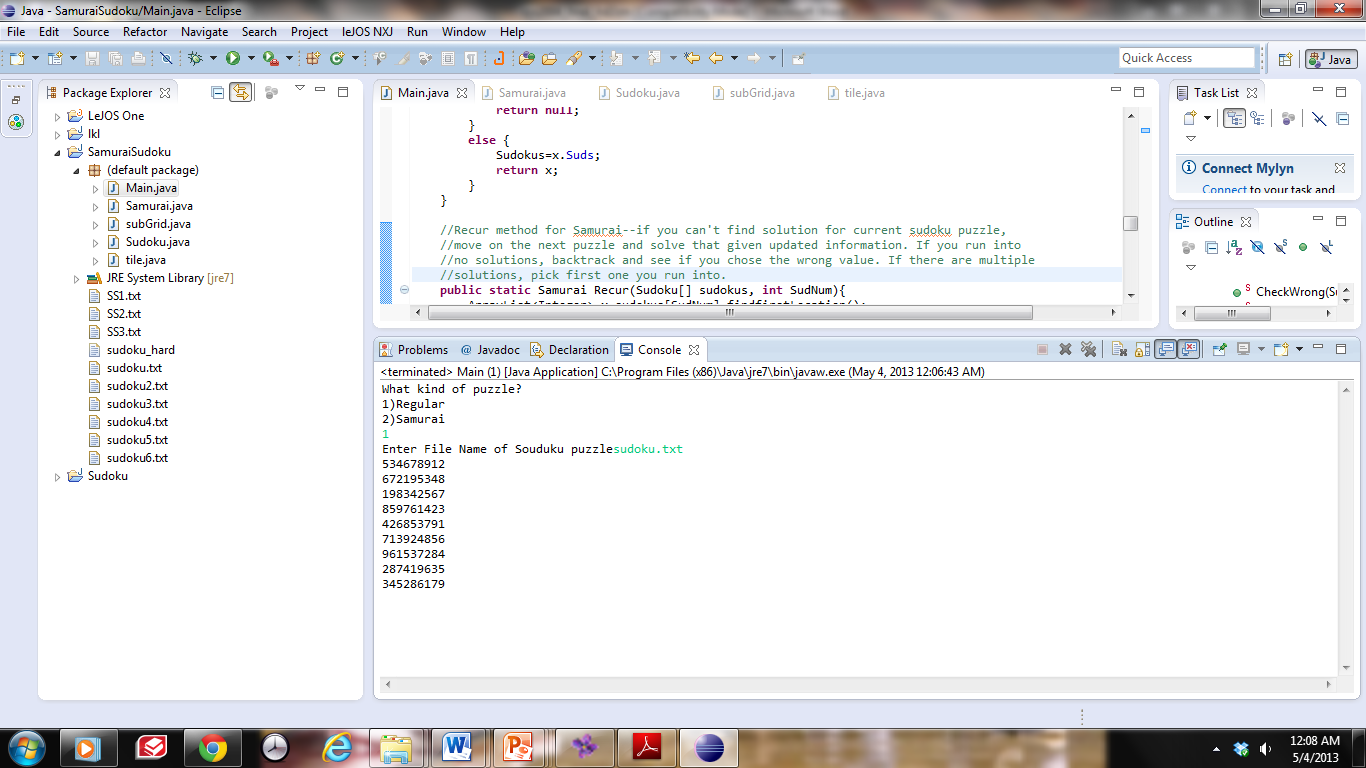
The basic algorithm used for this Sudoku and Samurai Sudoku is that it tries to solve it by logic as much as it can. It has roughly 6 logic statements it follows to make each decision every time. Before the logic the way the data is stored is split into a tile class, which represents each number spot. If the value is 0 then it also contains an Array List of possibilities that the value may be. Then there is a subGrid glass which is the each of the 9 3x3's in each Sudoku. This class is just used as a reference to which grid each tile is in. Then the Sudoku class, which does quite a bit like keeping all the data together for each Sudoku. The Sudoku class does all the basic logic to determine everything. The Samurai Class is also basically just a class to hold all the Sudoku's and it's responsible for the toString that prints out the samurai Sudoku. The logic is that for each tile it will first look in its row and see all the values that already exist and takes them out of the possibilities array list. Then it performs the same logic on the column and the same for the subGrid it belongs to. Then it performs one more set of logic to determine values. If a value exits in the possibilities of a tile but it doesn't exist in the possibilities of any of the other tiles' possibilities of its row then this tile must contain that as the value. The same logic again is performed to each tile for its column and then its grid. Then the last thing it does is, it goes through each of the tiles and determines if there is a tile with only one possible value then set's that as its value. With just this much logic it can solve most intermediate Sudoku's. After that a bit of luck is necessary or in other words a guess is necessary. So there is a recursive method that basically finds the first spot with no value and just tries a value and tries to solve and every time it gets stuck it will go back and see if it’s wrong or it needs to guess again. if wrong it will backtrack and it will continue to repeat. If it's always wrong then it will return null and showing it's not possible to accomplish. The Samurai Sudoku is the same exact logic the only difference is that it looks through all 5 Sudoku's each time and even during the guessing it finds the first and guesses and then repeats solving and if necessary guessing again. This is the basic algorithm.

To execute the program, download all the files. Run the main program and follow the instructions the program gives you.

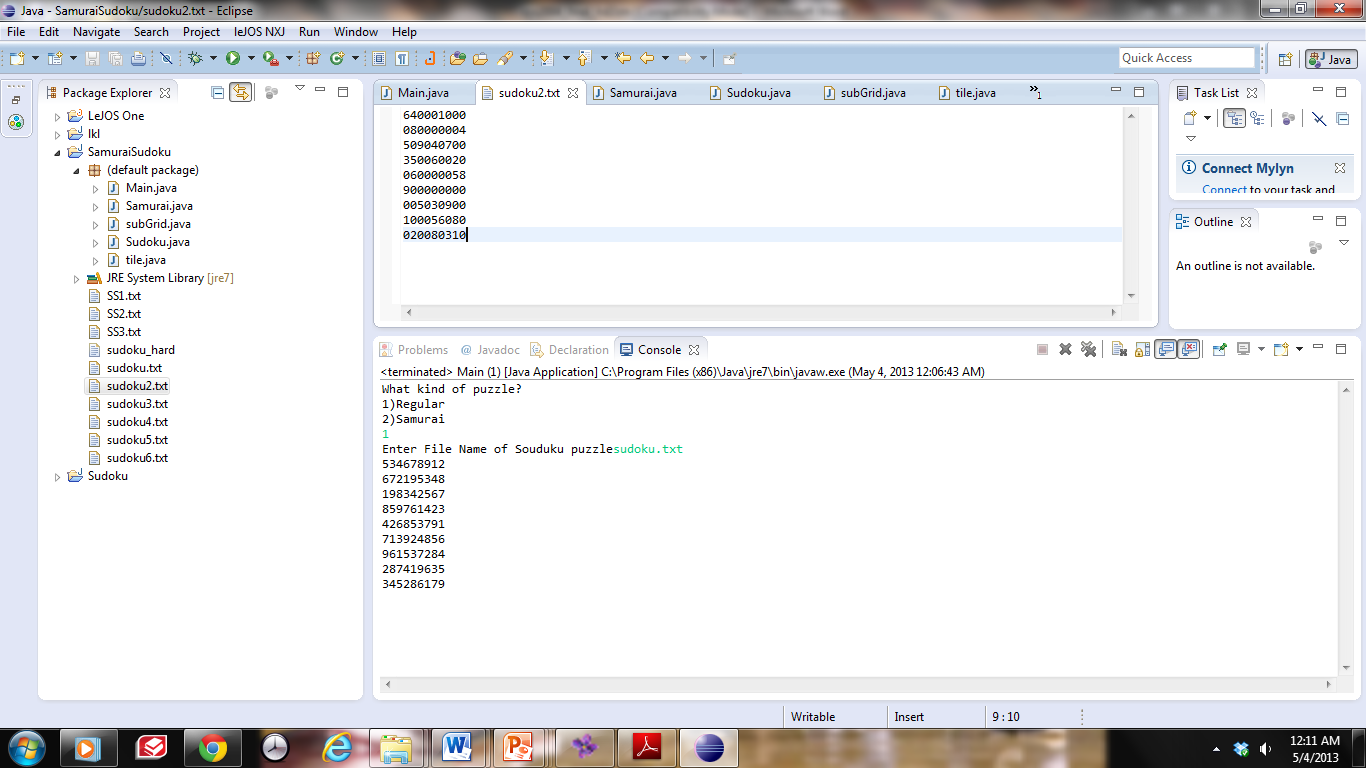
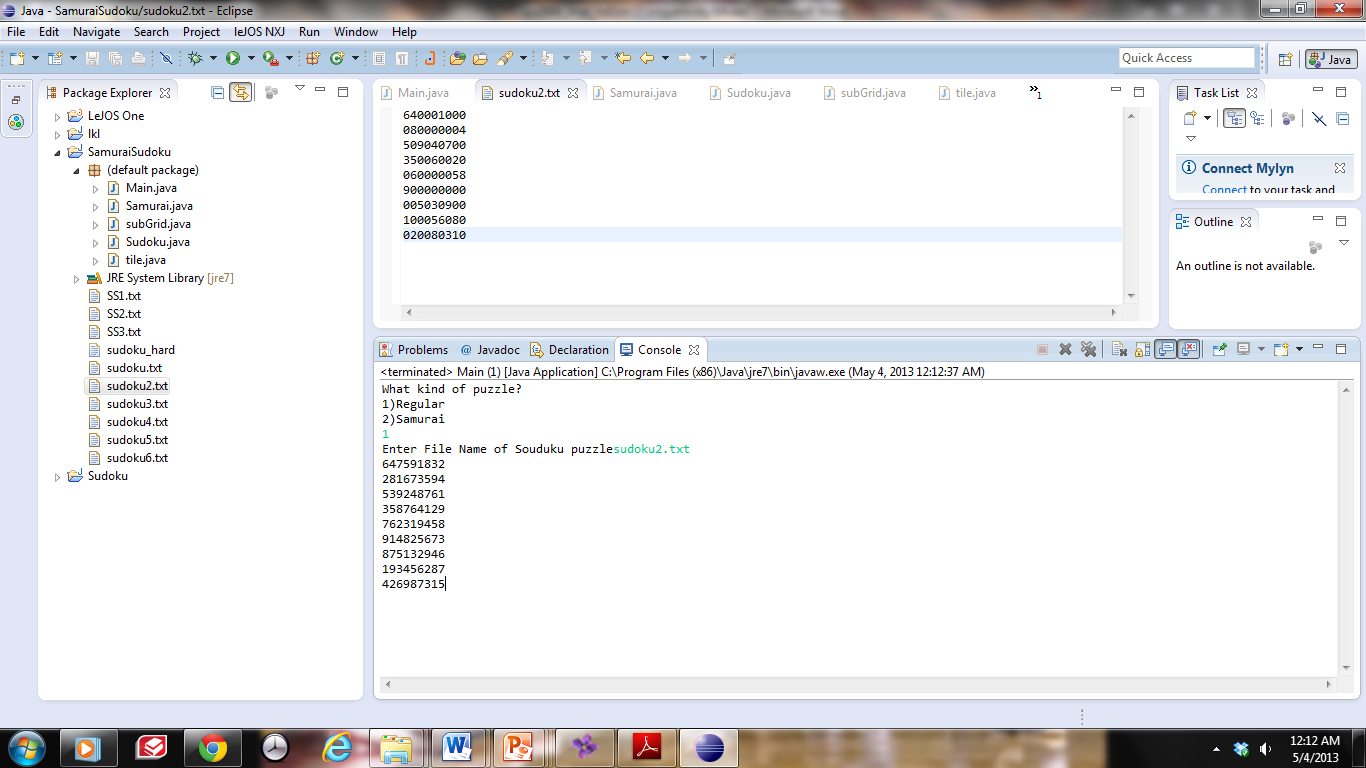
**Screenshots**

**Regular Puzzles**

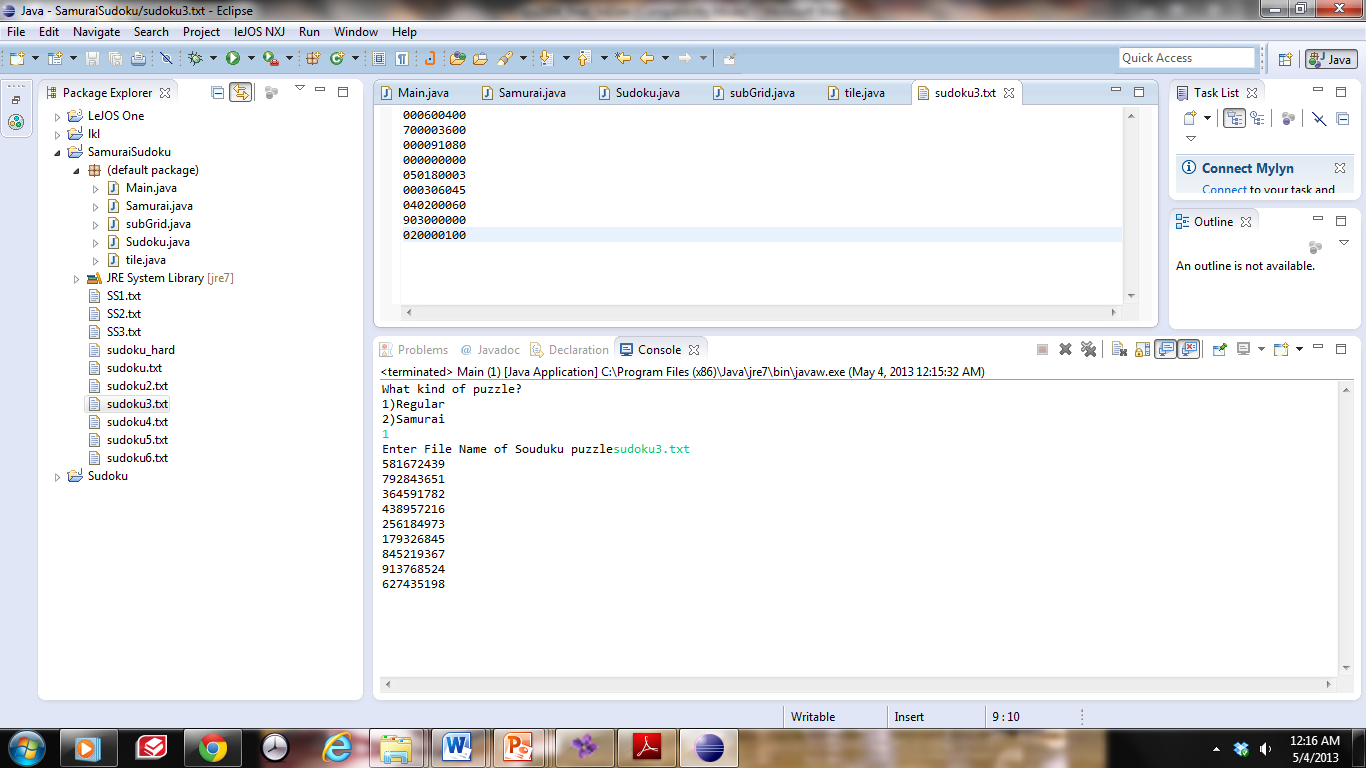
sudoku1.txt Output

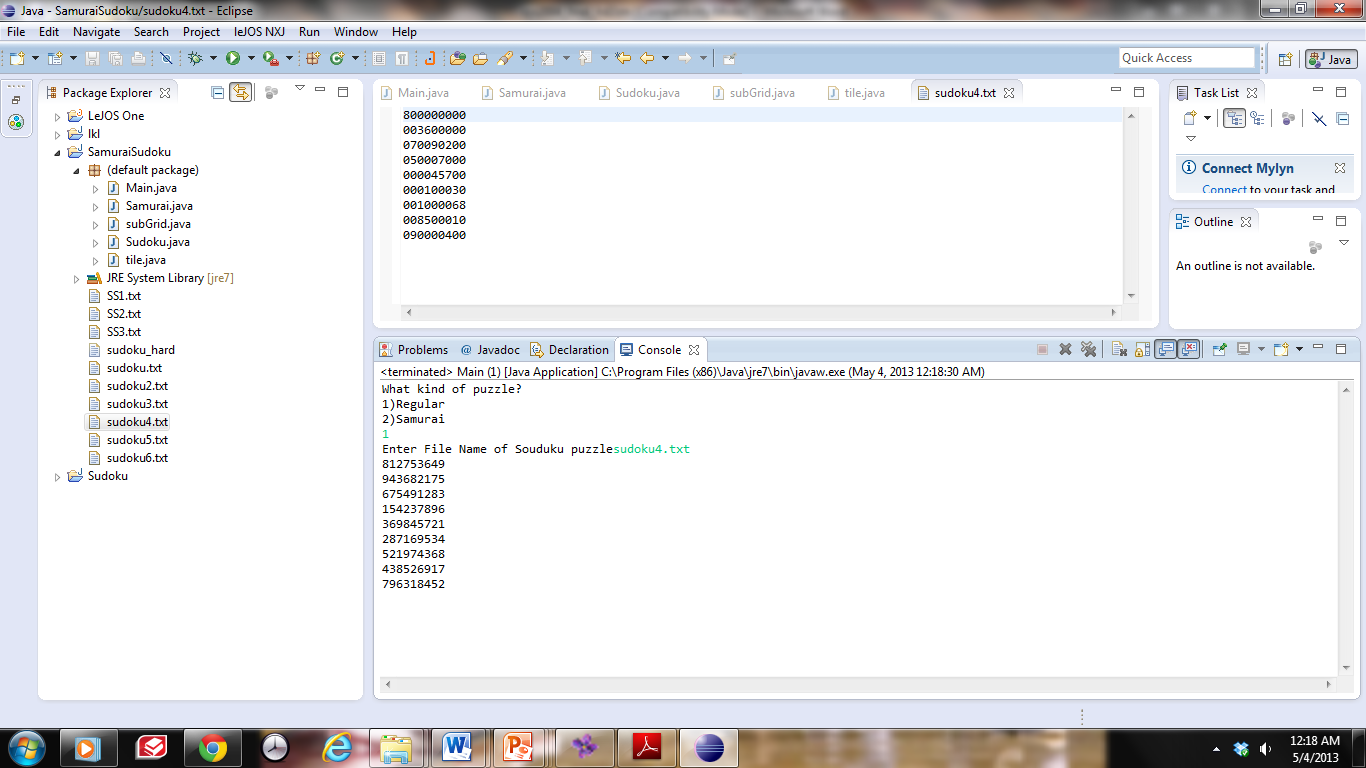
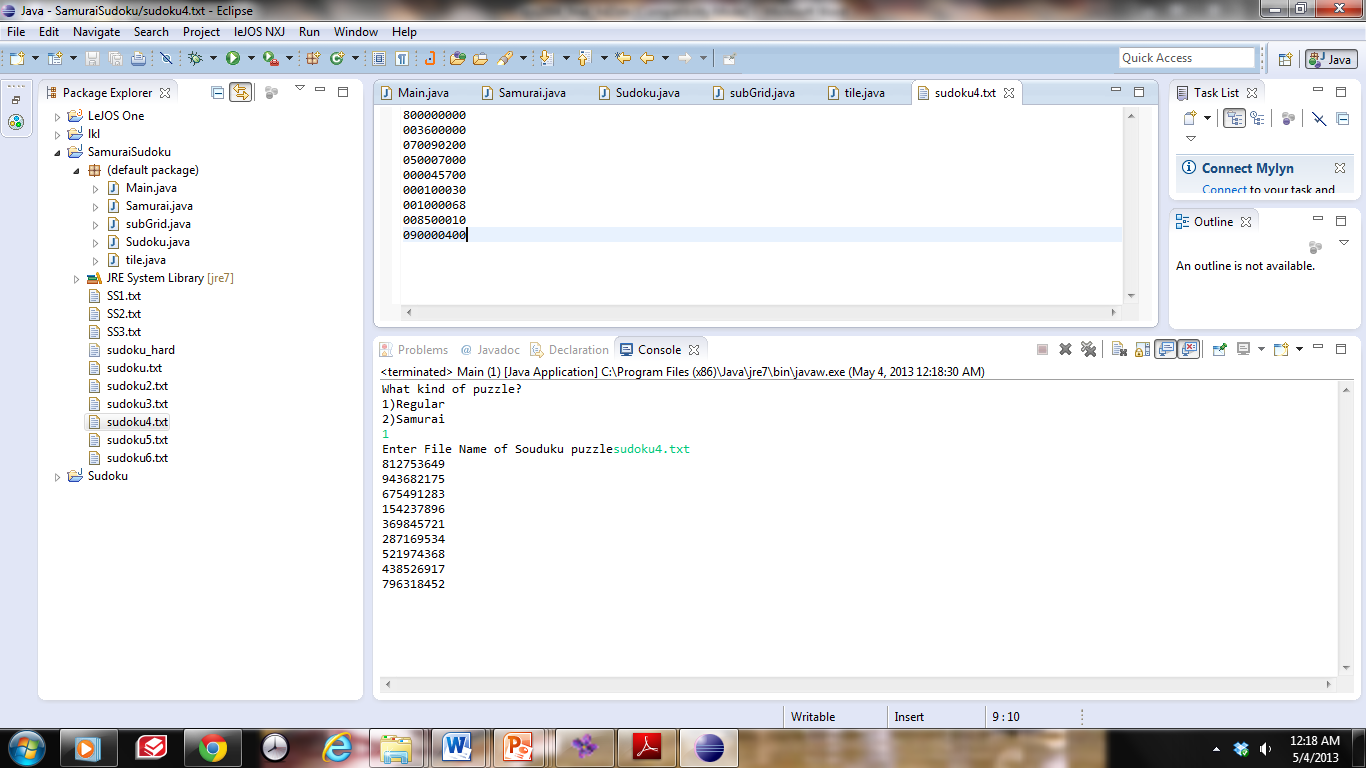
sudoku2.txt Output

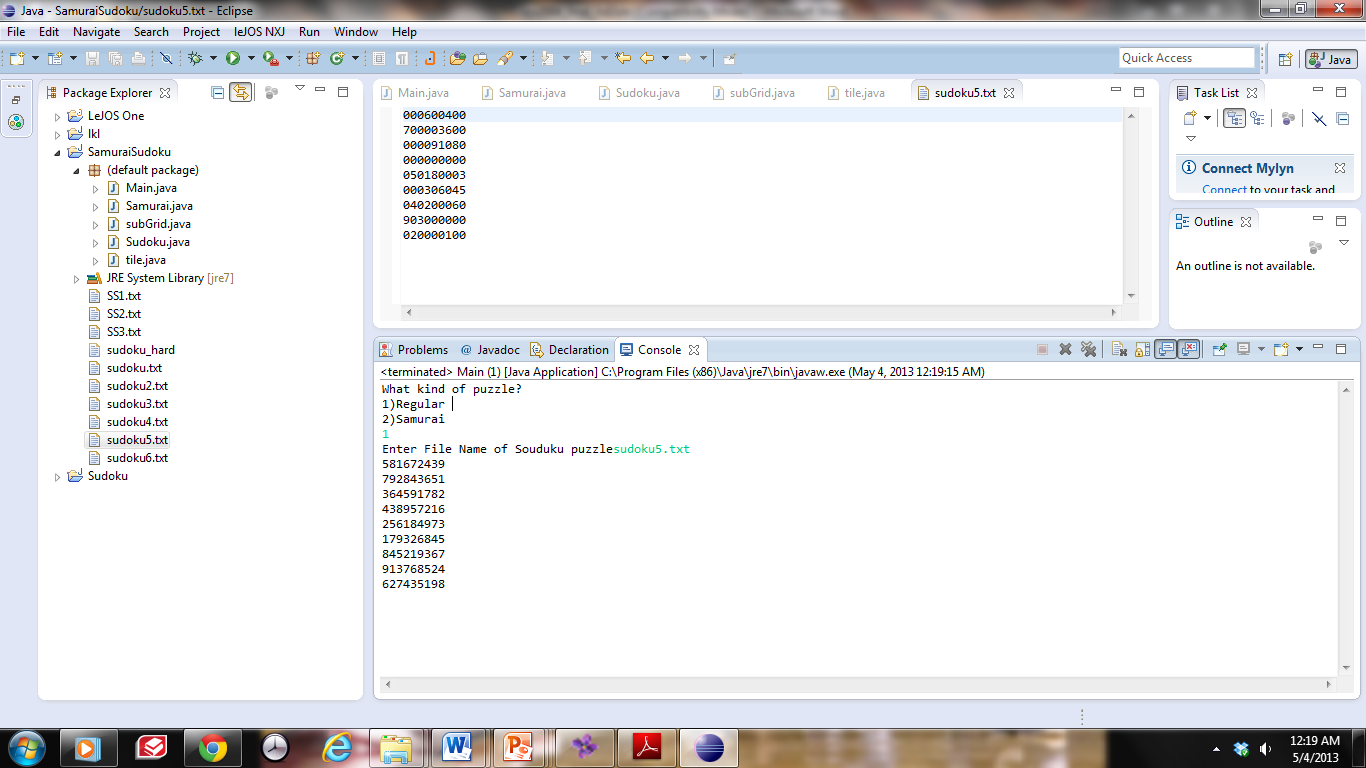
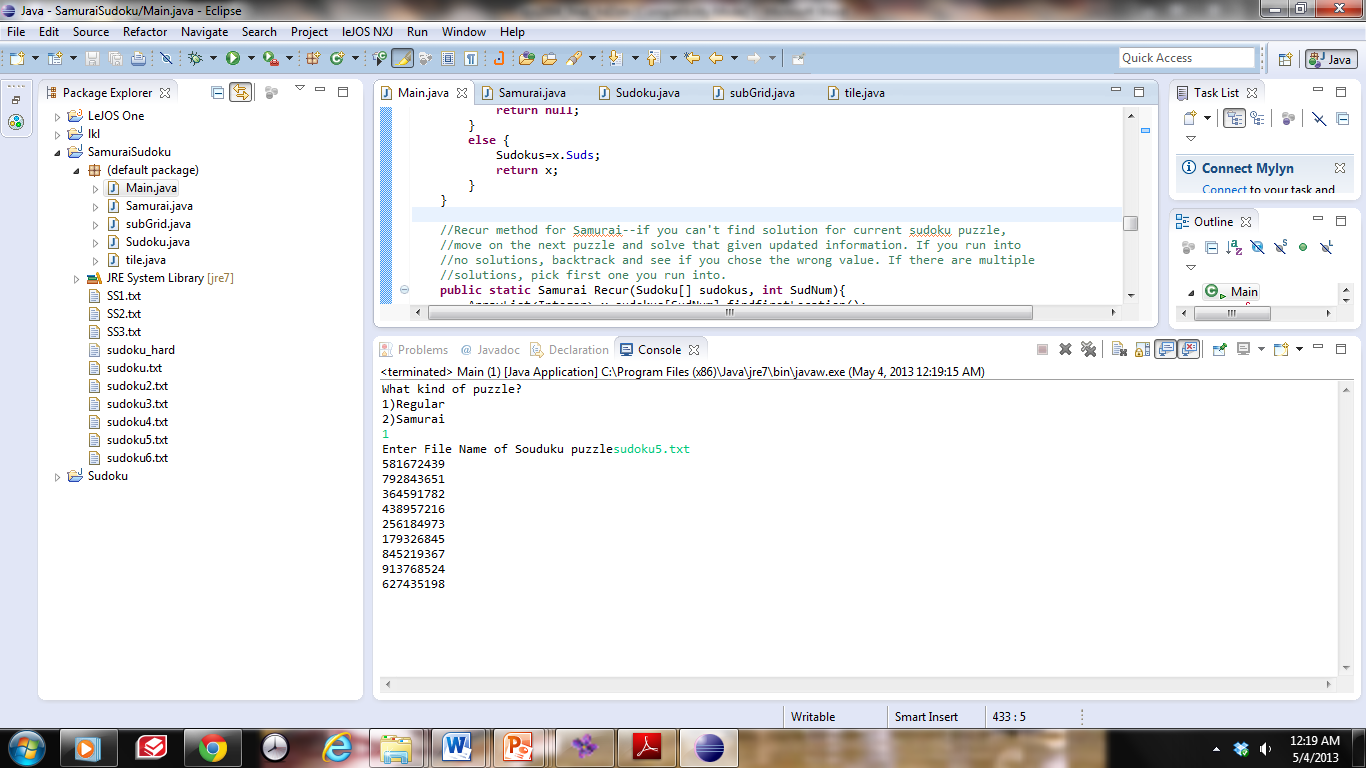
sudoku3.txt Output

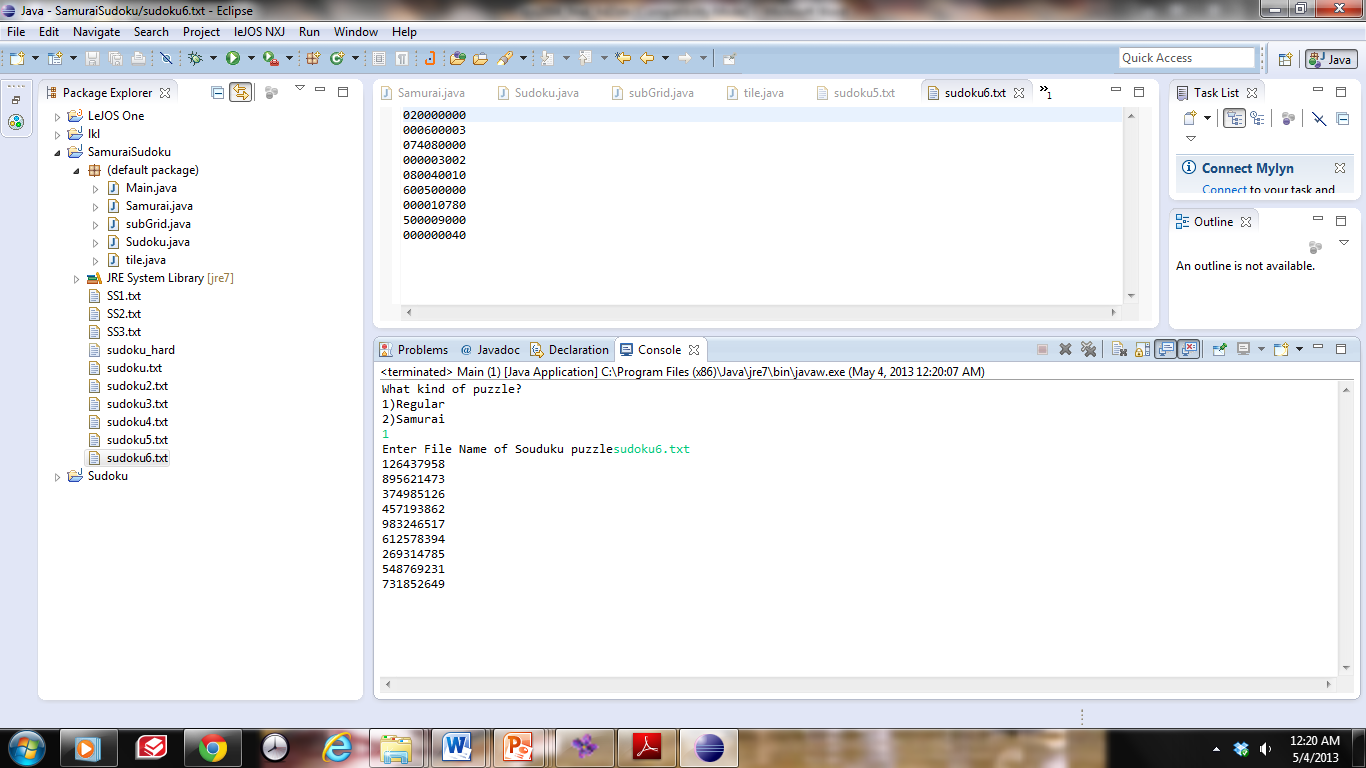
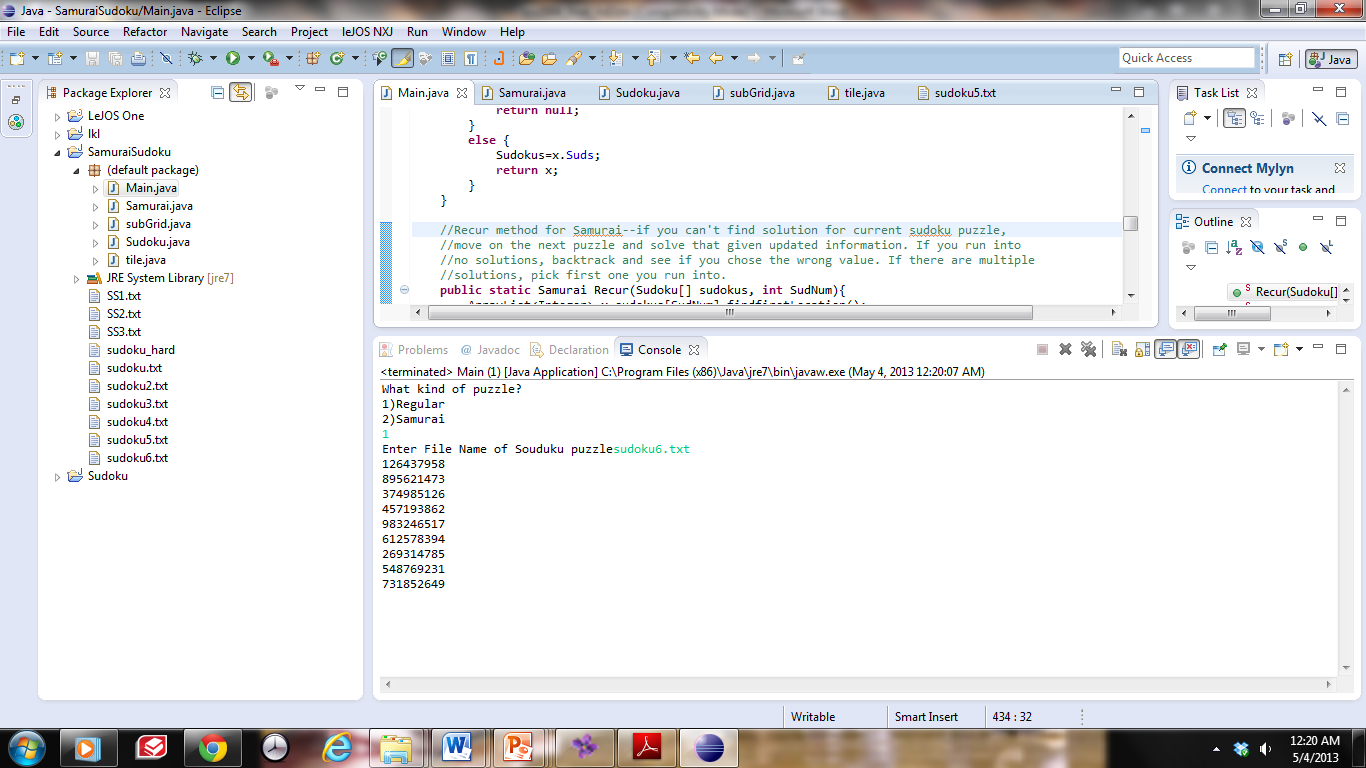
sudoku4.txt Output

sudoku5.txt Output

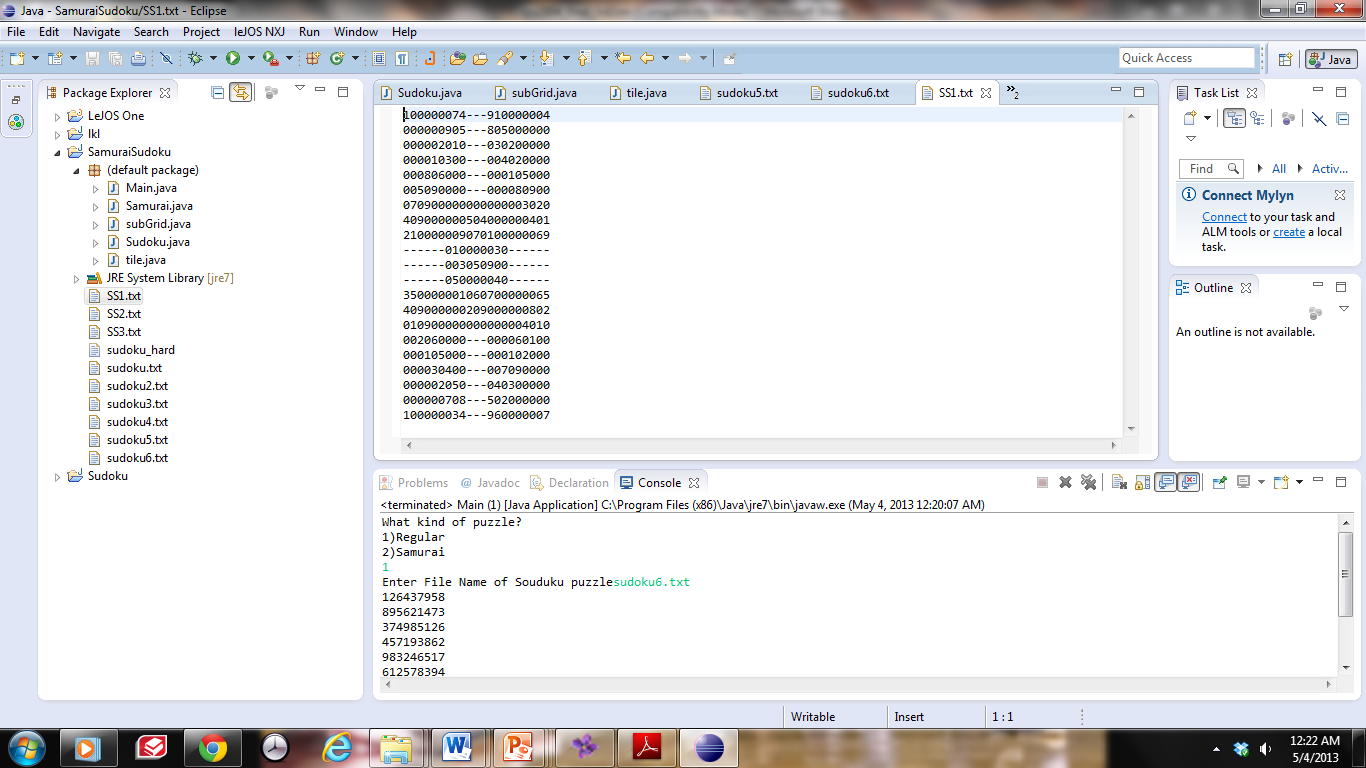
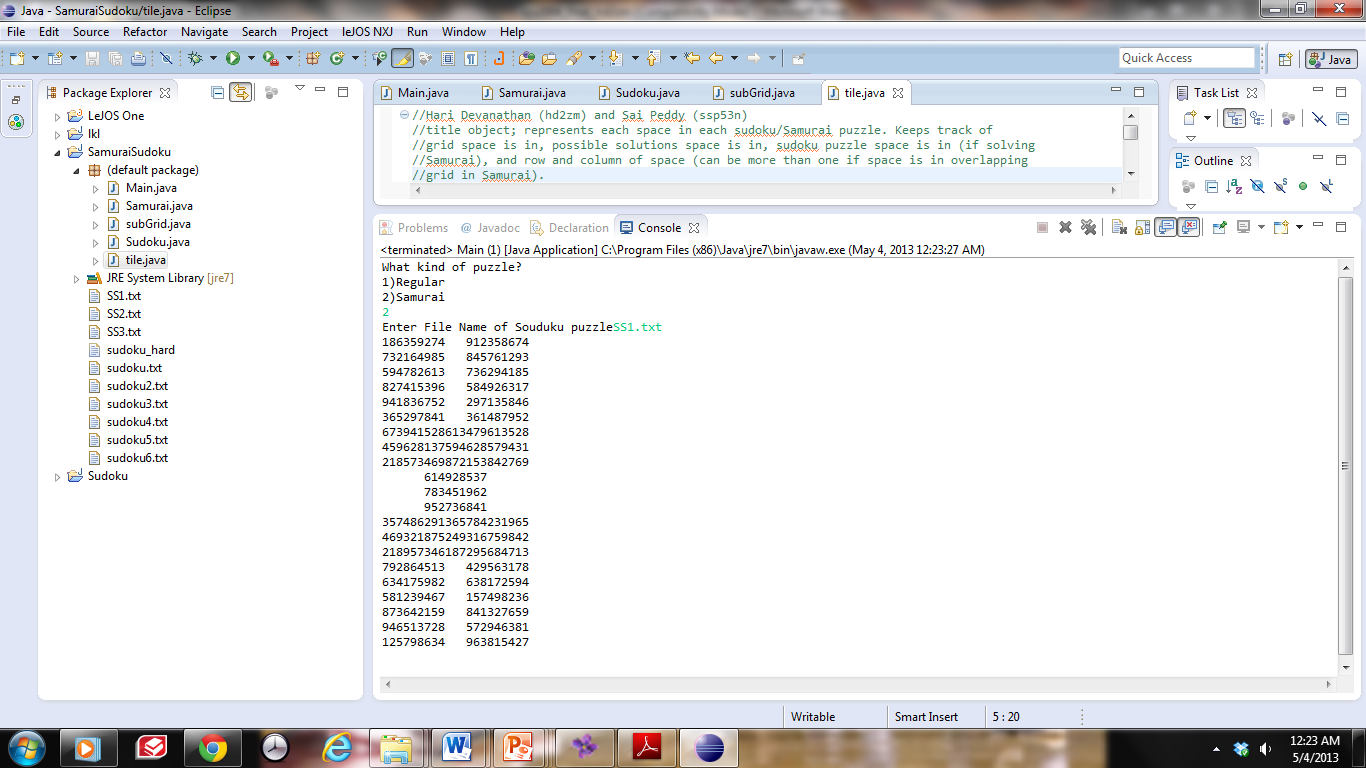
 

sudoku6.txt Output

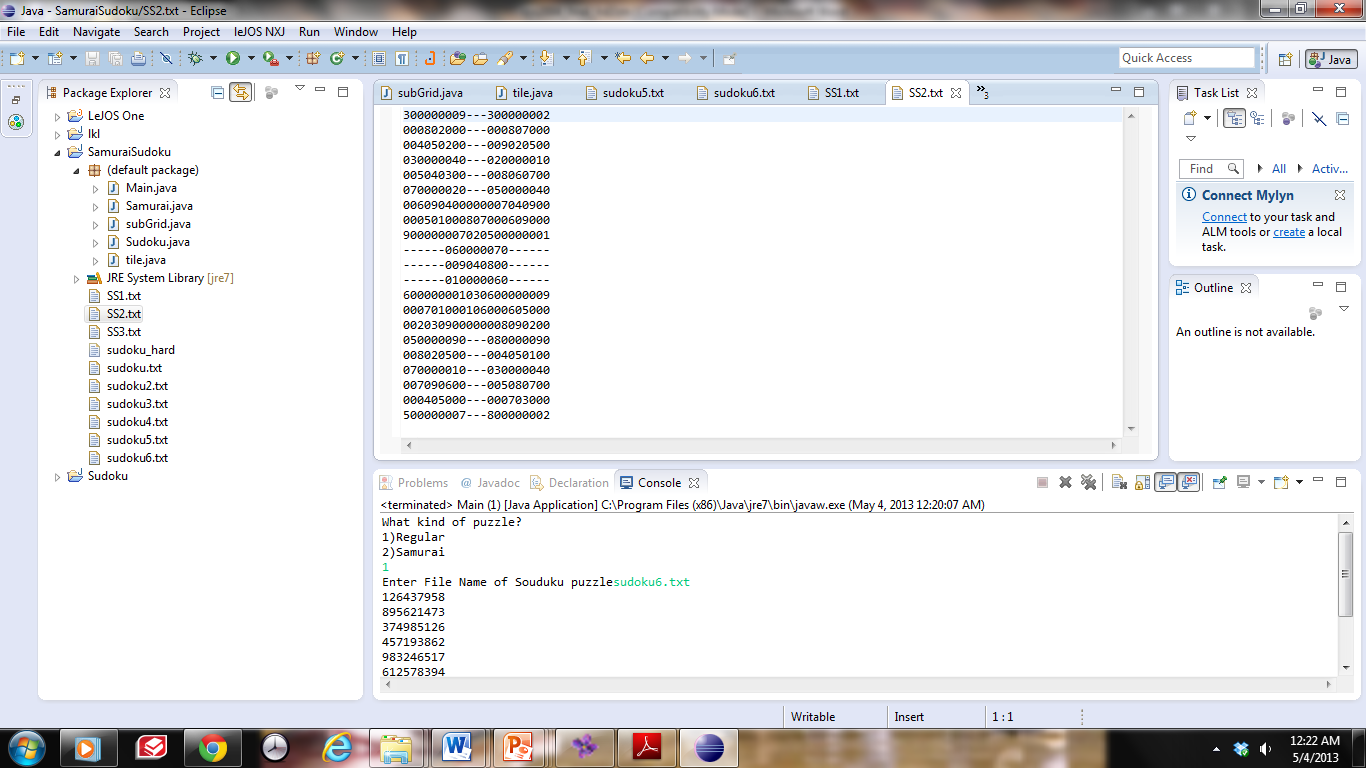
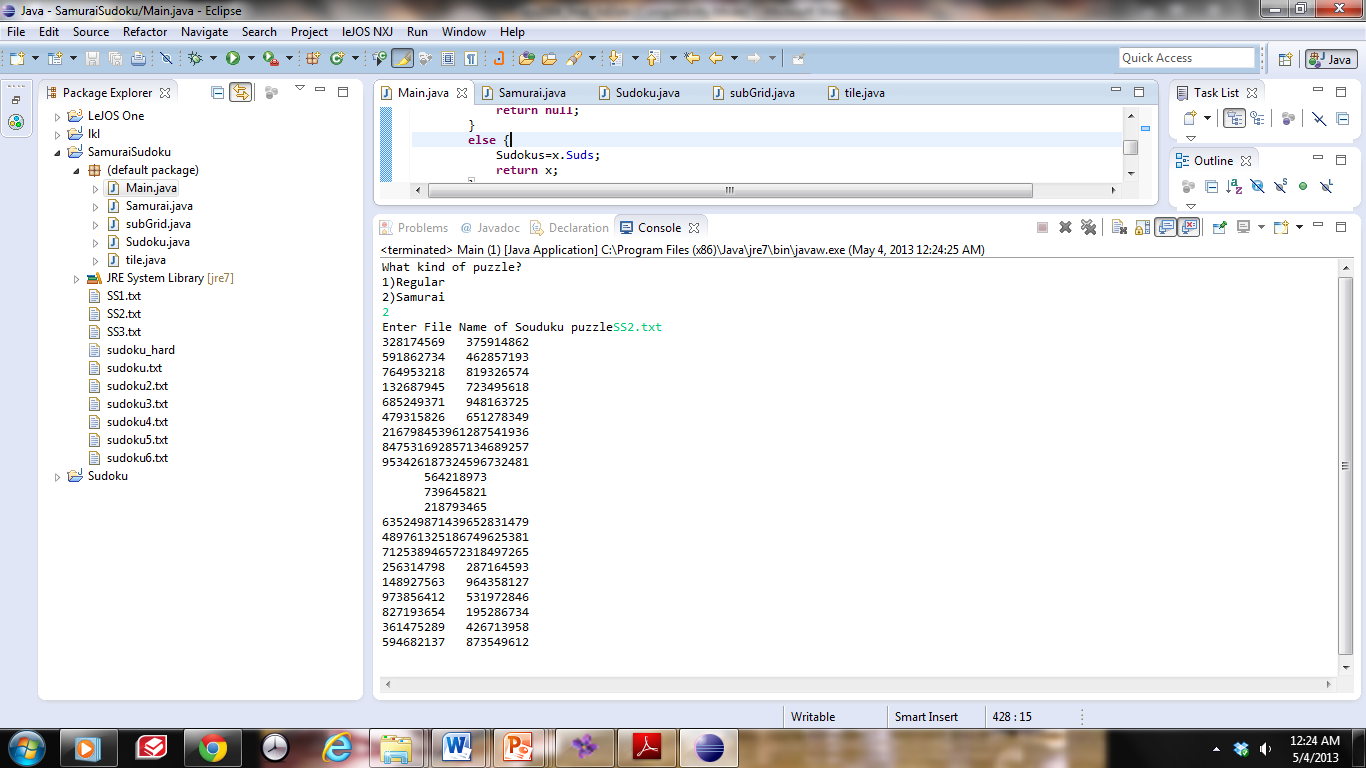
 

**Samurai Puzzles**

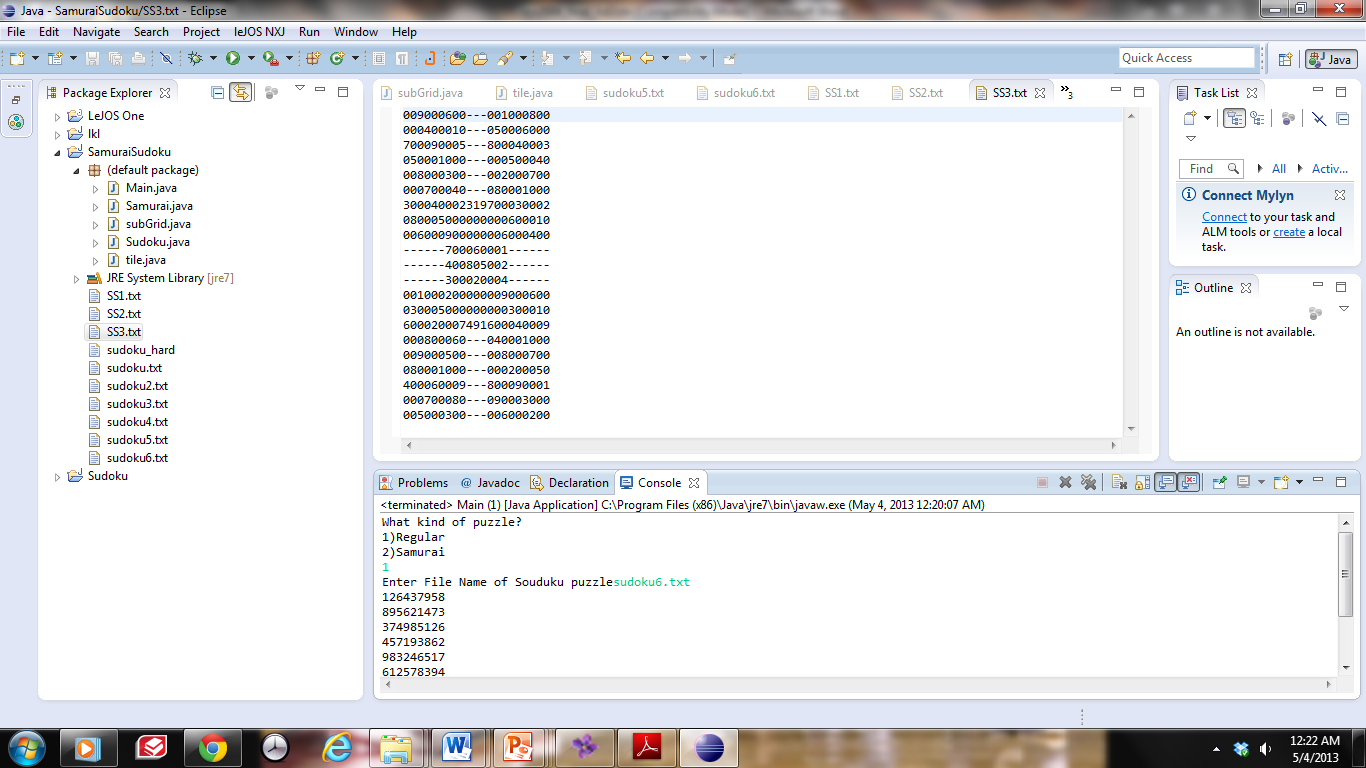
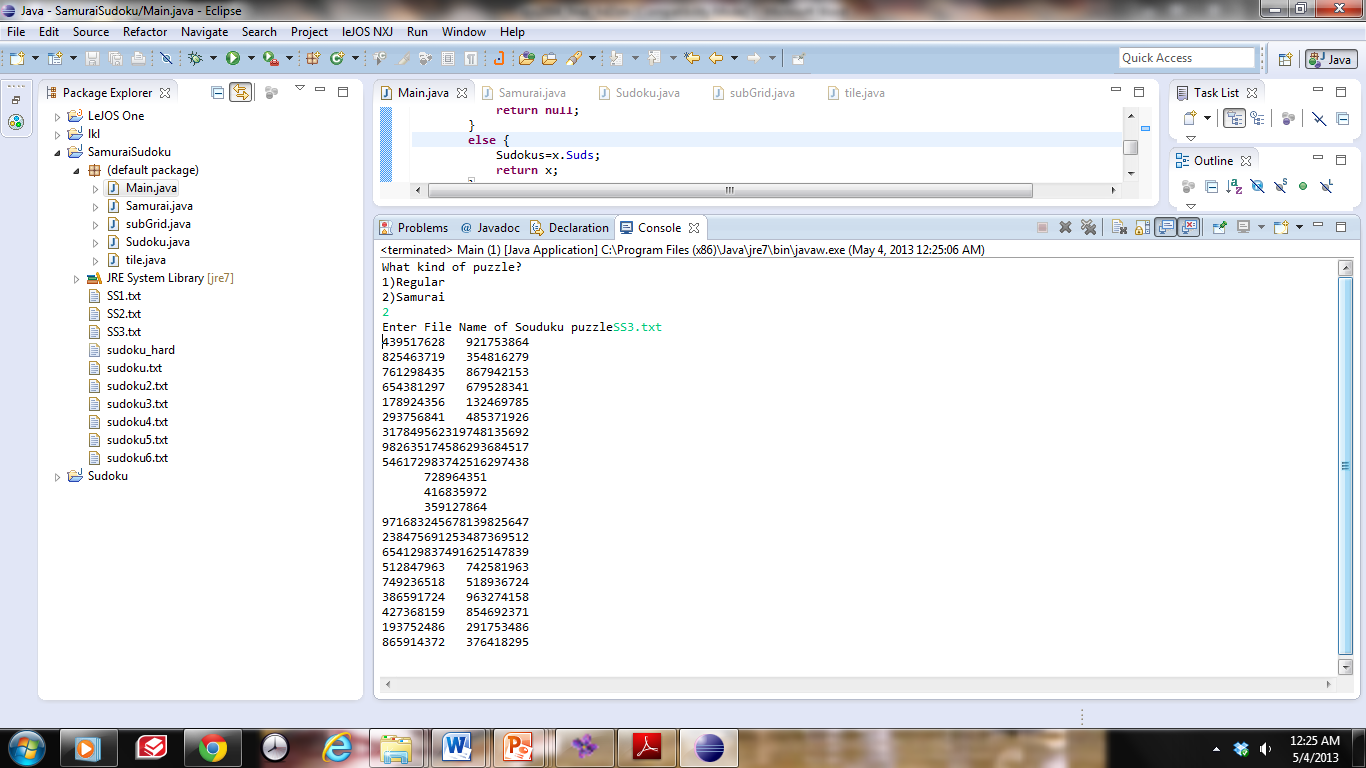
SS1.txt Output

SS2.txt Output

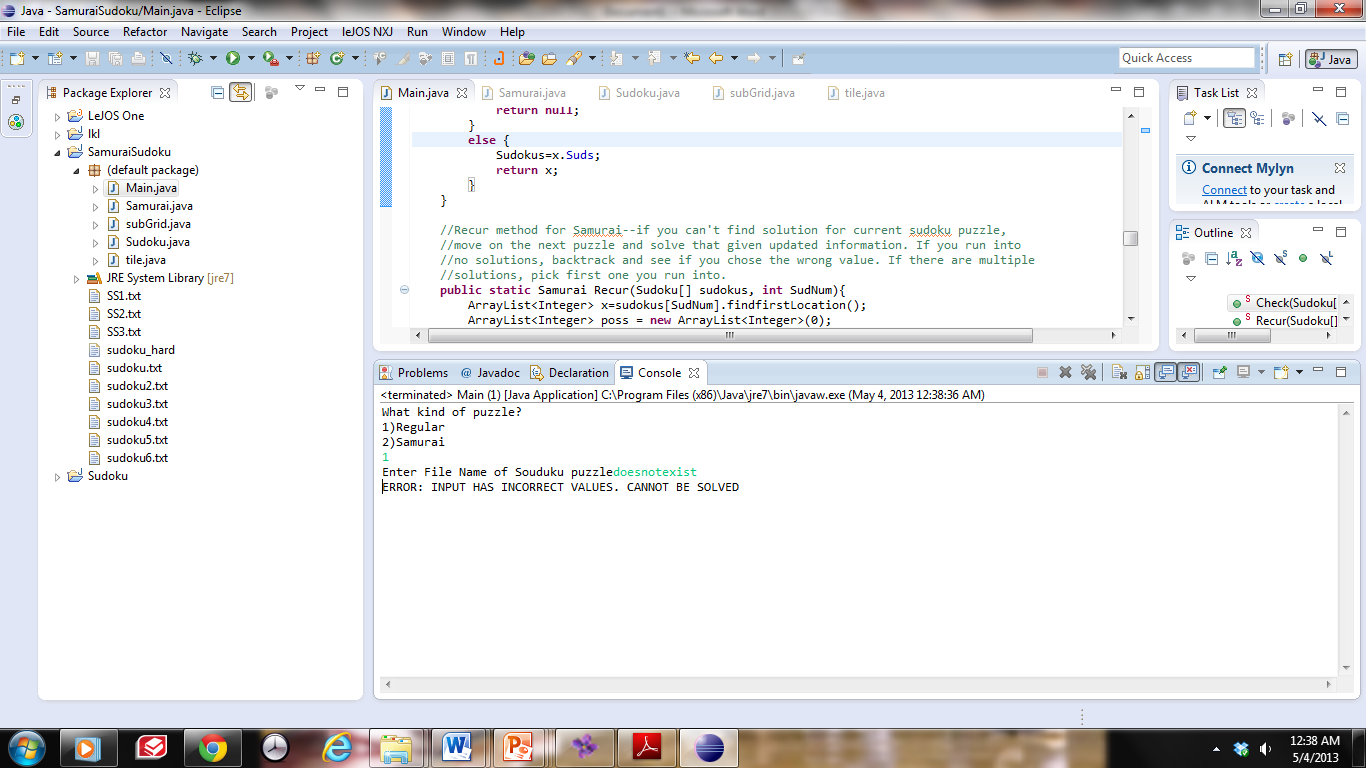
 

SS3.txt Output

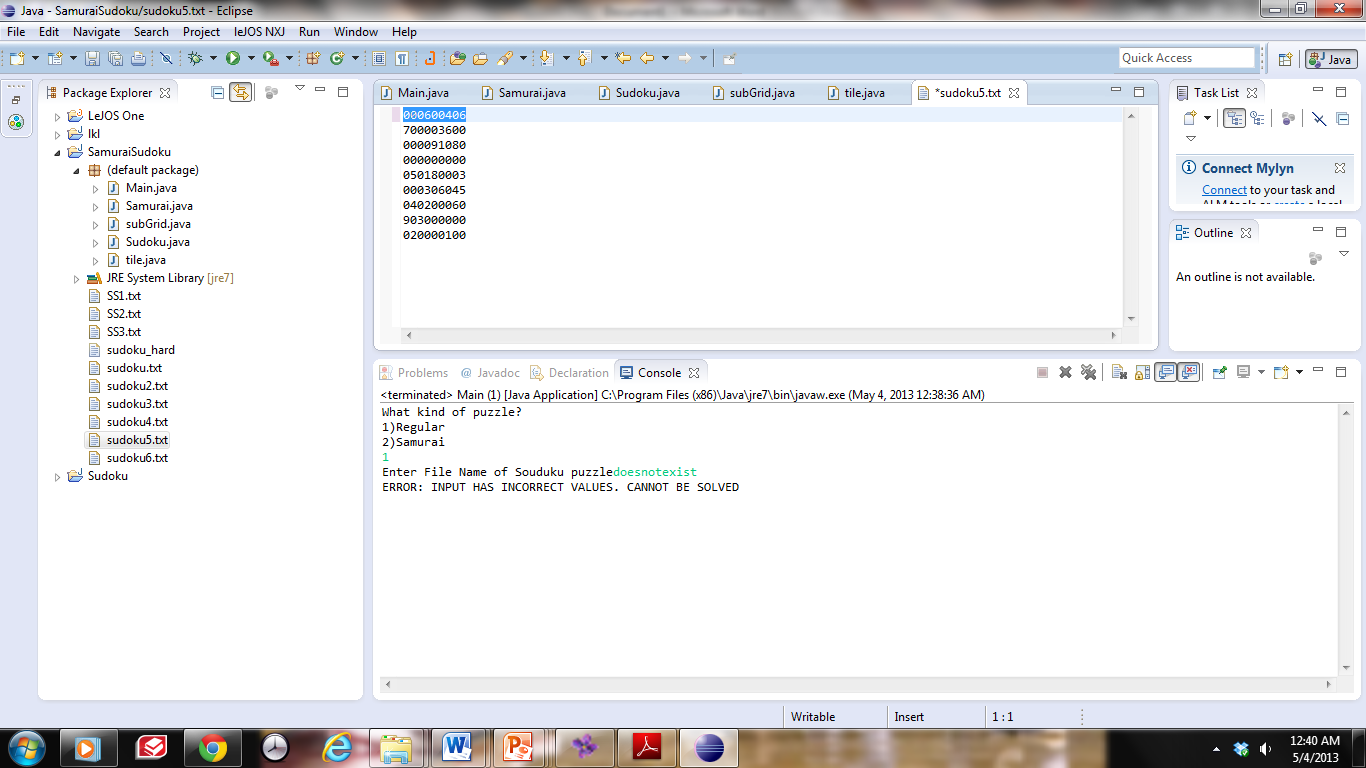
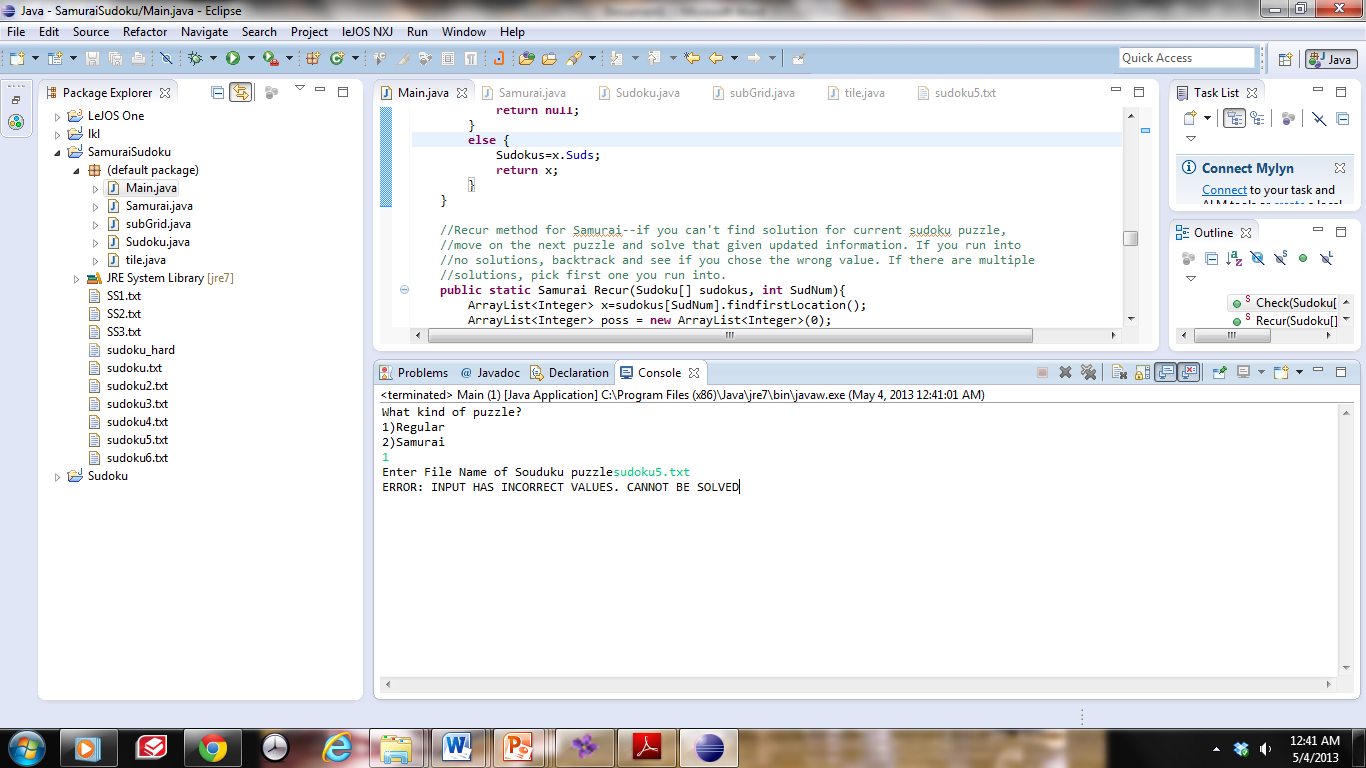
**Error Checking**

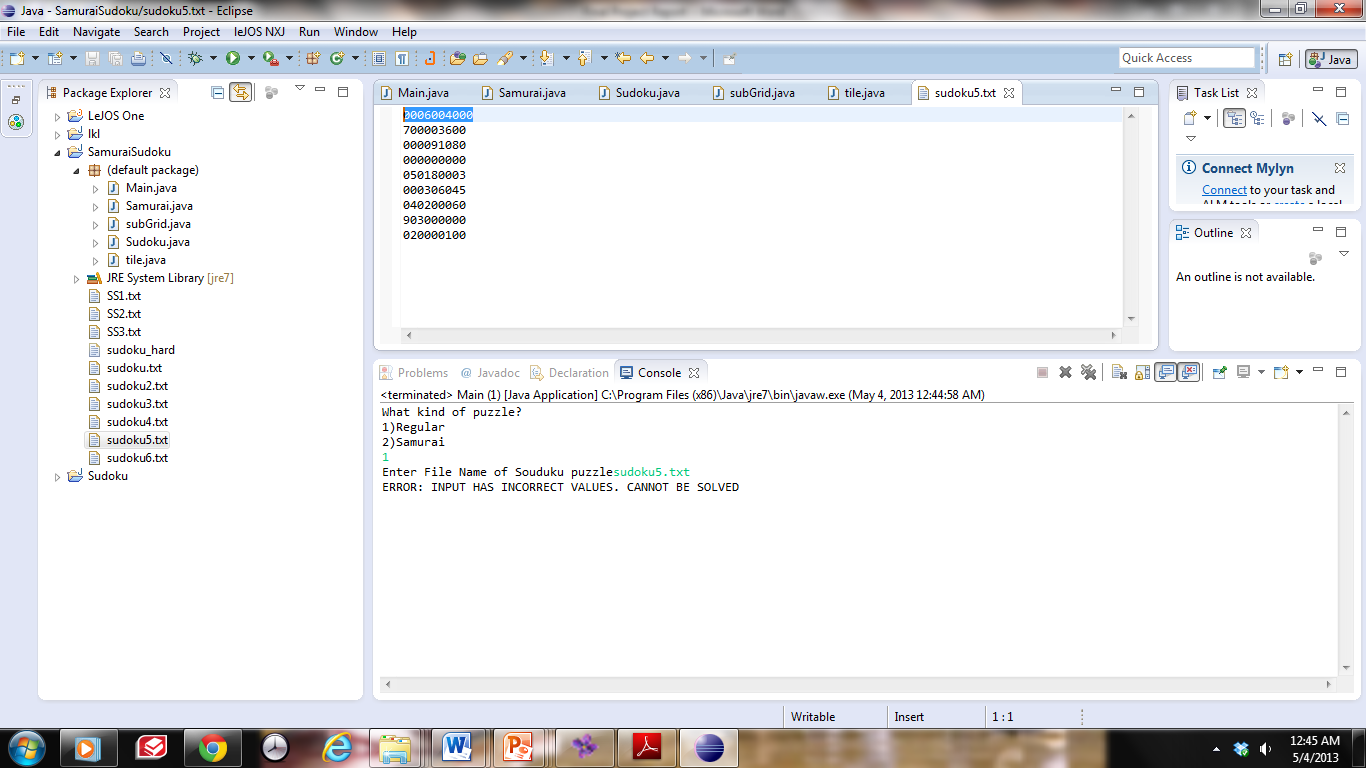
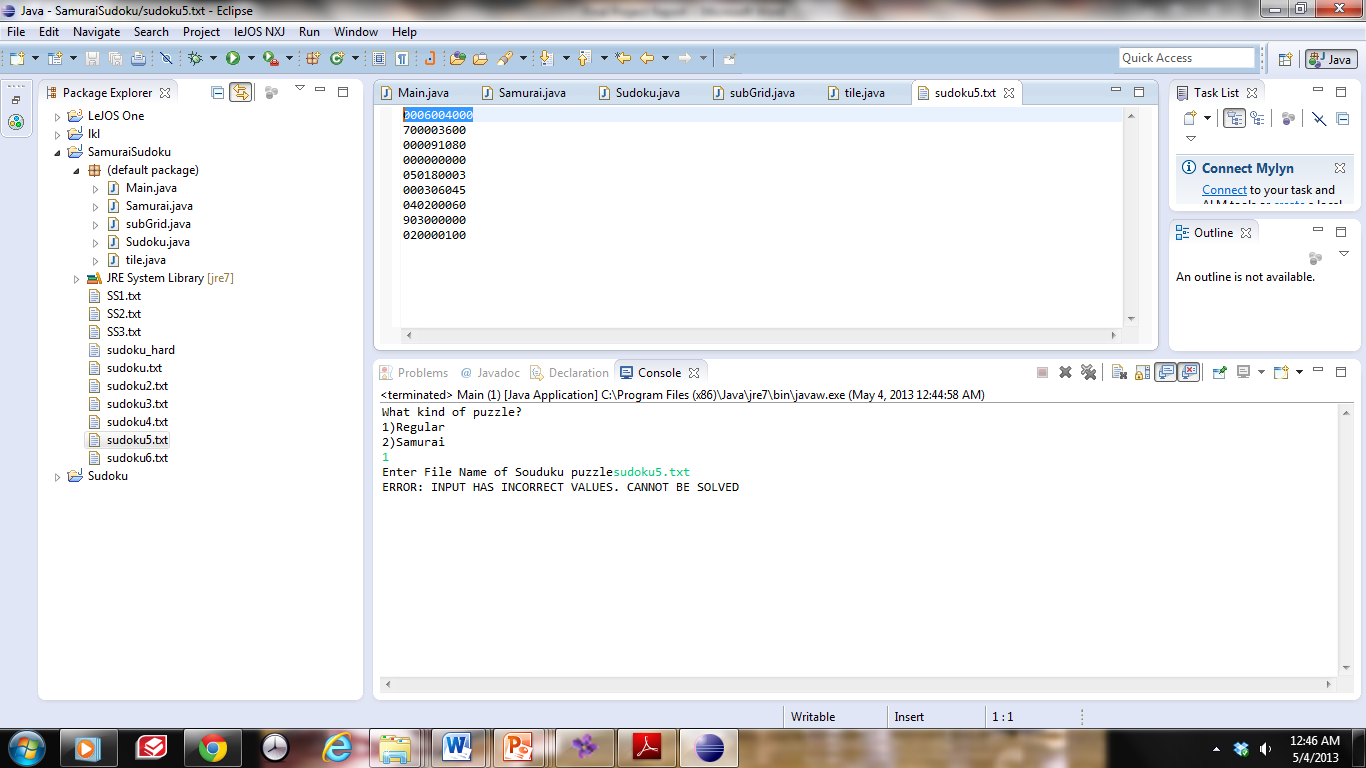
**Invalid text file**



**Error in regular input**

sudoku5.txt Output

**Error in samurai input**

SS2.txt Output

