TASK 5:

```
package ELliteTech;
import java.util.Scanner;

public class SudokuSolver {

	public static void main(String[] args) {
	    int[][] board = new int[9][9];
	    Scanner scanner = new Scanner(System.in);

	    System.out.println("Enter the Sudoku puzzle (use 0 for empty 
cells):");
	    for (int i = 0; i < 9; i++) {
	        for (int j = 0; j < 9; j++) {
	            board[i][j] = scanner.nextInt();
	        }
	    }

	    System.out.println("\nUnsolved Sudoku Puzzle:");
	    printBoard(board);

	    if (solveSudoku(board)) {
	        System.out.println("\nSolved Sudoku Puzzle:");
	        printBoard(board);
	    } else {
	        System.out.println("\nThis Sudoku puzzle cannot be solved.");
	    }

	    scanner.close();
	}

	private static boolean solveSudoku(int[][] board) {
	    for (int row = 0; row < 9; row++) {
	        for (int col = 0; col < 9; col++) {
	            if (board[row][col] == 0) { // Empty cell found
	                for (int num = 1; num <= 9; num++) {
	                    if (isValidMove(board, row, col, num)) {
	                        board[row][col] = num;

	                        if (solveSudoku(board)) { // Recursively solve 
the rest of the board
	                            return true;
	                        }

	                        board[row][col] = 0; // Backtrack
	                    }
	                }
	                return false; // No valid number found, trigger 
backtracking
	            }
	        }
	    }
	    return true; // Puzzle solved
	}

	private static boolean isValidMove(int[][] board, int row, int col, int 
num) {
```

```
            // Check row
            for (int i = 0; i < 9; i++) {
                if (board[row][i] == num) {
                    return false;
                }
            }

            // Check column
            for (int i = 0; i < 9; i++) {
                if (board[i][col] == num) {
                    return false;
                }
            }

            // Check 3x3 grid
            int startRow = row - row % 3;
            int startCol = col - col % 3;
            for (int i = 0; i < 3; i++) {
                for (int j = 0; j < 3; j++) {
                    if (board[startRow + i][startCol + j] == num) {
                        return false;
                    }
                }
            }

            return true;
        }

        private static void printBoard(int[][] board) {
            for (int i = 0; i < 9; i++) {
                for (int j = 0; j < 9; j++) {
                    System.out.print(board[i][j] + " ");
                }
                System.out.println();
            }
        }
    }
```

OUTPUT

```
Enter the Sudoku puzzle (use 0 for empty cells):
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9

Unsolved Sudoku Puzzle:
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
```

```
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9

Solved Sudoku Puzzle:
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```