

**Course Experiment Report**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course:** | Java Language | | | | | | |
|  |  | | | | | | |
| **Semester:** | 1-18th | **week** | 2nd | **year** | | 1st | **term** |
|  |  |  |  |  | |  |  |
| **Major:** | Software Engineering | | | | | **Class:** | 2019 |
|  |  | | | | |  |  |
| **Student name:** | 冯春霖 | | **Student No.:** | | 222019321062074 | | |
|  |  | |  | |  | | |
| **Teacher:** | Wang Xiaomeng | | | | | | |

College of Computer and Information Science

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Exp3 Array | | |
| Time | 2020.10.26 | Type | □Verification □Design □Synthetical |
| 1. Answer the questions  (1) What is the difference between array variables and primitive data type variables?  A: Array is a continuous set of primitive data types in memory space, with support for random access using subscripts.  (2) How to determine the location of the 3 \* 3 box in above experiment.  A: Divide the whole board into nine 3 \* 3 pieces, and divide by 3 to determine which box the input in.  (3) How to understand that a two-dimensional array is an array of one-dimensional arrays.  A: Two-dimensional arrays can be thought of as one-dimensional arrays where each element is a one-dimensional array and each row represents an element of the outer one-dimensional array, a one-dimensional array  (4) Other experience.  Unlike C, the elements of each row of a 2D array in Java can be of different lengths, which makes the use of 2D arrays more flexible and saves memory when large amounts of data are involved.  2. All Codes  **package** week8;  **import** java.util.Scanner;  **public** **class** Ex1 {    **public** **static** **void** **main**(String[] args) {  Scanner **input** = **new** Scanner(System.***in***);  **int**[][] **grid** = {{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},  {0,6,0,0,0,0,0,0,0},  {0,0,0,4,1,9,0,0,5},  {0,0,0,0,8,0,0,7,9}};  System.***out***.println("\*\*\*\*\*\*Game Start!\*\*\*\*\*\*");  *print*(grid);  **while**(*countRemain*(grid) != 0)  {  System.***out***.print("Enter row, column and number([1-9] [1-9] [1-9]):");  **int** **row** = input.nextInt();  **int** **col** = input.nextInt();  **int** **num** = input.nextInt();  **if** (*isValid*(grid, row - 1, col - 1, num))  grid[row - 1][col - 1] = num;  *print*(grid);  }  System.***out***.println("You win!");  input.close();  }    **public** **static** **void** **print**(**int** grid[][])  {  **for** (**int** **i** = 0; i < 9; i++)  {  **for** (**int** **j** = 0; j < 9; j++)  {  System.***out***.printf(" %d", grid[i][j]);  **if** (j == 2 || j ==5)  System.***out***.print(" |");  }  System.***out***.println();  **if** (i == 2 || i == 5)  *printRowDivider*();  }  }    **public** **static** **void** **printRowDivider**()  {  System.***out***.println("-------|-------|-------");  }    **public** **static** **boolean** **isValid**(**int** grid[][], **int** \_row, **int** \_col, **int** \_num)  {  **if** (\_num < 1 || \_num > 9 || \_row < 0 || \_row > 8 || \_col < 0 || \_col > 8)  {  System.***out***.println("Invalid number, try again");  **return** **false**;  }  **if** (grid[\_row][\_col] != 0)  {  System.***out***.println("The cell is not empty, try again");  **return** **false**;  }  **for** (**int** **i** = 0; i < 9; i++)  **if** (grid[i][\_col] == \_num)  {  System.***out***.println("There is same number in this column, try again");  **return** **false**;  }  **for** (**int** **j** = 0; j < 9; j++)  **if** (grid[\_row][j] == \_num)  {  System.***out***.println("There is same number in this row, try again");  **return** **false**;  }  **for** (**int** **i** = \_row - \_row % 3; i < \_row - \_row % 3 + 3; i++)  **for** (**int** **j** = \_col - \_col % 3; j < \_col - \_col % 3 + 3; j++)  **if** (grid[i][j] == \_num)  {  System.***out***.println("There is same number in this block, try again");  **return** **false**;  }  **return** **true**;  }    **public** **static** **int** **countRemain**(**int**[][] grid)  {  **int** **count** = 0;  **for** (**int** **i** = 0; i < 9; i++)  **for**(**int** **j** = 0; j < 9; j++)  **if** (grid[i][j] == 0)  count++;  **return** count;  }    }  **Screenshot of execution:**  Results of various invalid inputs:    Update the array when input is valid: | | | |

|  |  |  |
| --- | --- | --- |
| Evaluation | Code Correctness (60%): |  |
| Experience (40%): |  |
| Score： | |