

**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 | Exp4 Objects and Classes | | |
| Time | 2020.11.8 | Type | □Verification □Design □Synthetical |
| 1. Answer the questions  (1) When will the no-arg constructor be automatically added?  A: There are no constructors defined in the class.  (2) What is the difference between static members and object members?  A: Static members belong to classes, while object members belong to objects. Static members can be called by their class name, while object members can only be called by their object name. Static members are created when the class is loaded, while object members are created as the object is instantiated.  (3) What do you think is the difference between procedural-oriented programming and object-oriented programming?  A: Procedural-oriented programming like C is accomplished by calling different subfunctions of the main function, and the program is executed in a predetermined order. object-oriented programming like Java and C++, the program instantiates the class in the main function and executes the member functions of the class to accomplish the desired goal. Object-oriented programming emphasizes abstraction, encapsulation, inheritance and polymorphism.  (4) Other experience.  In this experiment, I learned how to debug a Java program using the debug mode of the eclipse IDE. The debugging mode helps us to know how the program is running by monitoring the member variables and functions during the process, and helps us to make changes to the program to achieve the desired purpose.  2. All Codes  Implement of play() in Player class:  **public** **void** **play**(Board board, Scanner input)  {  System.***out***.printf("Player %s[%s] put a stone at:", **this**.getName(), Stone.*colorToString*(color));  **int** **row** = input.nextInt() - 1;  **int** **column** = input.nextInt() - 1;  **boolean** **success** = board.putAStone(row, column, **new** Stone(color));  **if**(!success)  {  System.***out***.println("Illegal Input");  **this**.play(board, input);  }  }  Implement of printBoard() in Board class:  **public** **void** **printBoard**()  {  System.***out***.println(" 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15");  **for**(**int** **i** = 0; i < 15; i++)  {  System.***out***.printf("%2d", i + 1);  **for** (**int** **j** = 0; j < 15; j++)  **if** (board[i][j] != **null**)  System.***out***.printf("%3c", board[i][j].getShape());  **else**  System.***out***.print(" +");  System.***out***.println();  }  }  Implement of putAStone() in Board class:  **public** **boolean** **putAStone**(**int** row, **int** column, Stone stone) {  **if** (row < 0 || row > ***SIZE*** - 1 || column < 0 || column > ***SIZE*** - 1 || board[row][column] != **null**)  {  **return** **false**;  }  **else**  {  board[row][column] = stone;  remain--;  whichColorToPlay = stone.getColor() == Stone.***WHITE*** ? Stone.***BLACK***:Stone.***WHITE***;  winColor = judge(row, column);  **return** **true**;  }  }  Implement of judge() in Board class:  **public** **int** **judge**(**int** row, **int** column)  {  String **kernel** = **new** String(**new** **char**[5]).replace("\0", String.*valueOf*(board[row][column].getColor()));    StringBuffer **lineX** = **new** StringBuffer();  **for**(**int** **i** = 0; i < ***SIZE***; i++)  **if**(board[row][i] != **null**)  lineX.append(board[row][i].getColor());  **else**  lineX.append("N");  **if**(lineX.indexOf(kernel) >= 0)  **return** board[row][column].getColor();    StringBuffer **lineY** = **new** StringBuffer();  **for**(**int** **i** = 0; i < ***SIZE***; i++)  **if**(board[i][column] != **null**)  lineY.append(board[i][column].getColor());  **else**  lineY.append("N");  **if**(lineY.indexOf(kernel) >= 0)  **return** board[row][column].getColor();    StringBuffer **lineDiagA** = **new** StringBuffer();  **if**(row >= column)  **for** (**int** **i** = row - column, **j** = 0; i < ***SIZE***; i++, j++)  **if**(board[i][j] != **null**)  lineDiagA.append(board[i][j].getColor());  **else**  lineDiagA.append("N");  **else**  **for** (**int** **i** = 0, **j** = column - row; j < ***SIZE***; i++, j++)  **if**(board[i][j] != **null**)  lineDiagA.append(board[i][j].getColor());  **else**  lineDiagA.append("N");  **if**(lineDiagA.indexOf(kernel) >= 0)  **return** board[row][column].getColor();    StringBuffer **lineDiagB** = **new** StringBuffer();  **if**(row + column < ***SIZE*** - 1)  **for** (**int** **i** = 0, **j** = row + column; j > 0; i++, j--)  **if**(board[i][j] != **null**)  lineDiagB.append(board[i][j].getColor());  **else**  lineDiagB.append("N");  **else**  **for** (**int** **i** = row + column - ***SIZE*** + 1, **j** = ***SIZE*** - 1; i < ***SIZE***; i++, j--)  **if**(board[i][j] != **null**)  lineDiagB.append(board[i][j].getColor());  **else**  lineDiagB.append("N");  **if**(lineDiagB.indexOf(kernel) >= 0)  **return** board[row][column].getColor();    **return** -1;  }  Implement of main class:  **while**(board.getWinColor() == -1 && board.getRemain() != 0)  {  **if**(board.getWhichColorToPlay() == player1.getColor())  player1.play(board, input);  **else**  player2.play(board, input);  board.printBoard();  }    **if**(board.getRemain() == 0 && board.getWinColor() == -1)  System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Game Draw\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  **else** **if**(board.getWinColor() == player1.getColor())  System.***out***.printf("Player1 %s[%s] won the game", player1.getName(), Stone.*colorToString*(player1.getColor()));  **else**  System.***out***.printf("Player2 %s[%s] won the game", player2.getName(), Stone.*colorToString*(player2.getColor()));  Screenshot of the program running:   1. The program ends in various situations where a player wins: 2. In row      1. In column      1. In forward diagonals 2. General case      1. Critical case      1. In reverse diagonals 2. General case      1. Critical case      1. The case of illegal operation: 2. The selected coordinates already have Stone present.      1. The selected coordinates are outside the Board's range. | | | |
|  | | | |

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