

**Course Experiment Report**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course:** | Java Language | | | | | | |
|  |  | | | | | | |
| **Semester:** | 1-18th | **week** | 2nd | **year** | | 1st | **term** |
|  |  |  |  |  | |  |  |
| **Major:** | Software Engineering | | | | | **Class:** | 2020 |
|  |  | | | | |  |  |
| **Student name:** | 温长锟 | | **Student No.:** | | 222020321062106 | | |
|  |  | |  | |  | | |
| **Teacher:** | 詹成 | | | | | | |

College of Computer and Information Science

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Exp4 Objects and Classes | | |
| Time | 2021.10.28 | Type | □Verification □Design □Synthetical |
| 1. Answer the questions  (1) When will the no-arg constructor be automatically added?  If the class does not have a constructor, a default parameterless constructor is automatically generated and the object field is initialized with the default value  (2) What is the difference between static members and object members?  Static members, also known as class members, refer to variables or methods modified with static modifiers.  Instance members are variables or methods that are not decorated with static.  The main difference is whether there is a static modification. The result is that static members can be called directly by the class name, and instance members need to be instantiated with the new identifier, so that the immortal object can call methods and variables   1. What do you think is the difference between procedural-oriented programming and object-oriented programming?   Process oriented programming is carried out step by step according to the process and process oriented; In short, it is to analyze the steps needed to solve the problem, and then call the function step by step. Object oriented programming is a process that faces various independent individuals in the problem and decomposes the program into the interaction between different objects.   1. Other experience.   Java is an object-oriented programming language, which not only absorbs various advantages of C + + language, but also discards the concepts of multi inheritance and pointer which are difficult to understand in C + +. Therefore, Java language has two characteristics: powerful function and easy to use. As the representative of static object-oriented programming language, Java language perfectly realizes the object-oriented theory and allows programmers to program complex programs in an elegant way of thinking  2. All Codes  3.1  public class RegularPolygon {  private int n = 3;  private double side = 1;  private double x = 0;  private double y = 0;    public RegularPolygon() {};    public RegularPolygon(int \_n, double \_side)  {  setN(\_n);  setSide(\_side);  };    public RegularPolygon(int \_n, double \_side, double \_x, double \_y)   {  setN(\_n);  setSide(\_side);  setX(\_x);  setY(\_y);  };    public void setN(int \_n)  {  this.n = \_n;  }    public int getN()  {  return this.n;  }    public void setSide(double \_side)  {  this.side = \_side;  }    public double getSide()  {  return this.side;  }    public void setX(double \_x)  {  this.x = \_x;  }    public double getX()  {  return this.x;  }    public void setY(double \_y)  {  this.y = \_y;  }    public double getY()  {  return this.y;  }    public double getPerimeter()  {  return (n \* side);  }    public double getArea()  {  return (n \* side \* side / (4 \* Math.*tan*(Math.*PI* / n)));  }    public static class test  {  public static void main(String[] args)   {  RegularPolygon rp1 = new RegularPolygon();  RegularPolygon rp2 = new RegularPolygon(6, 2.0);  RegularPolygon rp3 = new RegularPolygon(4, 3.0, 1.0, 1.0);  System.*out*.printf("The perimeter and the area of the "  + "first polygon are %4.2f and %4.2f\n", rp1.getPerimeter(),  rp1.getArea());  System.*out*.printf("The perimeter and the area of the "  + "second polygon are %4.2f and %4.2f\n", rp2.getPerimeter(),  rp2.getArea());  System.*out*.printf("The perimeter and the area of the "  + "third polygon are %4.2f and %4.2f\n", rp3.getPerimeter(),  rp3.getArea());  }  } }  Result：    3.2  public class MyInteger {  private final int value;  public MyInteger(int value) {  this.value = value;  }  public int getValue() {  return value;  }  public boolean isEven() {  if (getValue() % 2 == 0)  return true;  else  return false;  }  public static boolean isEven(int v) {  return v%2==0;  }  public static boolean isEven(MyInteger v) {  return *isEven*(v.value);  }  public boolean equals(int v) {  return v == value;  }  public boolean equals(MyInteger v) {  return v.value == value;  } }    3.3  Board  public class Board {  public final static int *SIZE* = 15;  private Stone[][] board;  private int remain;  private int whichColorToPlay;  private int winColor = -1;    public Board()   {  board = new Stone[*SIZE*][*SIZE*];  remain = *SIZE* \* *SIZE*;  whichColorToPlay = (int)(Math.*random*() \* 2);  }    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();  }  }    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;  }  }    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; i < row + column + 1; 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;  }   public int getRemain()   {  return remain;  }    public void setRemain(int remain)   {  this.remain = remain;  }    public int getWhichColorToPlay()   {  return whichColorToPlay;  }    public void setWhichColorToPlay(int whichColorToPlay)   {  this.whichColorToPlay = whichColorToPlay;  }    public int getWinColor()   {  return winColor;  }    public void setWinColor(int winColor)   {  this.winColor = winColor;  }  }  Player  public class Player {  private String name;  private int color;    public Player(String name, int color)   {  this.setName(name);  this.setColor(color);  }   public String getName()   {  return name;  }    public void setName(String name) {  this.name = name;  }    public int getColor() {  return color;  }    public void setColor(int color) {  this.color = color;  }    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);  }  }   }  Stone  public class Stone {  public final static int *WHITE* = 0;  public final static int *BLACK* = 1;  private int color;    public Stone(int color)   {  this.setColor(color);  }    public int getColor()   {  return color;  }    public void setColor(int color)   {  this.color = color;  }    public static String colorToString(int color)   {  return color == *WHITE* ? "white": "black";  }    public char getShape()   {  return color == *WHITE* ? '\u25CB': '\u25CF';  }  }  Gomoku  import java.util.Scanner;  public class Gomoku {  public static void main(String[] args)   {  Scanner input = new Scanner(System.*in*);  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Welcome to GomokuGame\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Board board = new Board();  System.*out*.print("Enter player1 name:");  String playerName1 = input.next();  System.*out*.print("Enter player2 name:");  String playerName2 = input.next();  int playerColor1 = (int)(Math.*random*() \* 2);  int playerColor2 = Math.*abs*(playerColor1 - 1);  Player player1 = new Player(playerName1, playerColor1);  Player player2 = new Player(playerName2, playerColor2);  System.*out*.printf("Player1[%s, %s] VS Player2[%s, %s]\n", player1.getName(), Stone.*colorToString*(player1.getColor()), player2.getName(), Stone.*colorToString*(player2.getColor()));    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()));    input.close();  }  }  Result: | | | |

|  |  |  |
| --- | --- | --- |
| Evaluation | Code Correctness (50%): |  |
| Experience (50%): |  |
| Score： | |