Objects and Classes

1 Introduction

Through this training, to make the students skilled in the use of class, object, inheritance, polymorphism to programming. To understand the definition of class, the definition of an instance variable, the definition of the method, the difference between the formal parameters and the real parameters, and the method of the parameters in the Java language. To understand abstract classes and interfaces and grasp how to program with them.

1.1 Evaluation

- Code Correctness: 60%
- Experimental Report: 40%

1.2 Knowledge Points

- The Concepts of Object and Class
- Class Structure
- Constructor and Object Creating
- Members Accessing
- Class Members and Instance Members
- Visibility Modifiers
- Encapsulation
- Passing Objects to Methods
- The Scope of Variables
- The keyword `this`
- Object Relationships
- Wrapper Classes of Primitive Data Types
- Auto-boxing and Auto-unboxing
- Procedural Paradigm vs Object-oriented Paradigm

2 Demonstration

2.1 Class Definition and Object Creating

Define a `Circle` class which contains an `attribute`, two `constructors` and one `method`. We can use its constructor to create a instance of the `Circle` class.

```
class Circle {
  /* The radius of this circle */
  double radius = 1.0;
  /* Construct a circle object */
  Circle() {}
  /* Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }
  /* Return the area of this circle */
  double getArea() {
    return radius * radius * 3.14159;
  }
}
public class Test{
    public static void main(String[] args){
        /* Create a circle using the no-arg constructor */
        Circle c1 = new Circle();
        /* Create a circle using the constructor with arguments*/
        Circle c2 = new Circle(2.0);
        System.out.println("The area of the circle [1] is " + c1.getArea());
        System.out.println("The area of the circle [2] is " + c2.getArea());
    }
}
```

2.2 Class Member Application

Declare the class variable `numberOfCircles` using the `static` keyword. This variable is shared by all objects of the class `Circle`. Whenever the constructor is called to create a circle object, the variable `numberOfCircles` is incremented by 1.

```
class Circle {
  double radius = 1.0;
  static int numberOfCircles;
  Circle() {
      numberOfCircles++;
  }
  Circle(double newRadius) {
    radius = newRadius;
    numberOfCircles++;
  }
  double getArea() {
    return radius * radius * 3.14159;
  }
}
public class Test{
    public static void main(String[] args){
        Circle c1 = new Circle();
        /* One circle instance has been created*/
        System.out.println("The total number of the circle that have been created
is " + Circle.numberOfCircles);
        Circle c2 = new Circle(2.0);
        /* Two circle instance has been created*/
        System.out.println("The total number of the circle that have been created
is " + Circle.numberOfCircles);
    }
}
```

2.3 Data Field Encapsulation

The visibility of the variables of classes are set to `private` for data field encapsulation, and the public `getter` and `setter` methods are provided for data field accessing and modifying.

```java
class Circle {
  private double radius = 1.0;
  private static int numberOfCircles;
  public Circle() {
      numberOfCircles++;
  }
  public Circle(double newRadius) {
    radius = newRadius;
    numberOfCircles++;
  }
  /* Provide public methods to access the private data fields*/
  public double getRadius(){
      return radius;
  }
  public static double getNumberOfCircles(){
      return numberOfCircles;
  }
  /* Provide public methods to modify the private data fields*/
  public void setRadius(double r){
      radius = r;
  }
  double getArea() {
    return radius * radius * 3.14159;
  }
}
public class Test{
    public static void main(String[] args){
        Circle c1 = new Circle();
        System.out.printf("The radius of the circle is %4.2f\n", c1.getRadius());
        System.out.printf("The number of circles is %d\n",
Circle.getNumberOfCircles());
    }
}
```

3 Experiment Content

3.1 Geometry: n-sided regular polygon

In an n-sided regular polygon, all sides have the same length and all angles have the same degree (i.e., the polygon is both equilateral and equiangular). Design a class named `RegularPolygon` that contains:

- A private `int` data field named `n` that defines the number of sides in the polygon with default value `3`.
- A private double data field named `side` that stores the length of the side with default value `1`.
- A private double data field named `x` that defines the x-coordinate of the polygon's center with default value `0`.

- A private double data field named `y` that defines the y-coordinate of the polygon's center with default value `0`.
- A `no-arg constructor` that creates a regular polygon with `default values`.
- A `constructor` that creates a regular polygon with the specified `number of sides and length of side`, centered at (0, 0).
- A `constructor` that creates a regular polygon with the specified `number of sides, length of side, and x- and y-coordinates`.
- `The accessor` and `mutator` methods for all data fields.
- The method `getPerimeter()`` that returns the perimeter of the polygon.
- The method `getArea()` that returns the area of the polygon. The formula for computing the area of a regular polygon is $Area = \frac{n \times s^2}{4 \times \tan\frac{\pi}{n}}$

(1) Define a class named `RegularPolygon`, and it has 4 private data fields.

```
public class RegularPolygon {

    private int n = ...;
    private double side = ...;
    private double x = ...;
    private double y = ...;

}
```

(2) Define `3` constructors for create and initialize a polygon.

```
public class RegularPolygon {

    public RegularPolygon(){
        /* ... */
    }
    public RegularPolygon(...){
        /* data fields initialzing */
    }
    public RegularPolygon(...){
        /* data fields initialzing */
    }
}
```

(3) Add `getter` and `setter` methods for accessing and modifying its data fields.

```
public class RegularPolygon {

    /* The getter and setter methods for the attribute n */
    public int getN() {
        return n;
    }
    public void setN(int n) {
        this.n = n;
    }
    /* The getter and setter methods for other attributes */

}
```

(4) Define a method named `getPerimeter` to calculate and return the perimeter of a polygon.

```
public class RegularPolygon {

    public double getPerimeter() {
        /* calculate and return the perimeter */
    }
}
```

(5) Define a method named `getArea` to calculate and return the area of a polygon.

```
public class RegularPolygon {

    public double getArea() {
        /* calculate and return the area */
    }
}
```

(6) Define a test class to test the class `RegularPolygon`.

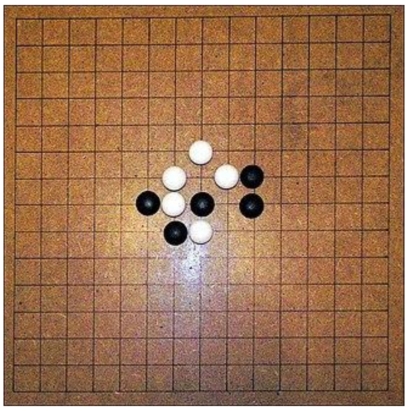
```
public class Test {

    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, 1);
        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 "
                + "first polygon are %4.2f and %4.2f\n", rp2.getPerimeter(),
rp2.getArea());
        System.out.printf("The perimeter and the area of the "
                + "first polygon are %4.2f and %4.2f\n", rp3.getPerimeter(),
rp3.getArea());
    }
}
```

(7) Compile and run the program.

```
$ javac Test.java
$ java Test
The perimeter and the area of the first polygon are 3.00 and 0.43
The perimeter and the area of the first polygon are 12.00 and 10.39
The perimeter and the area of the first polygon are 12.00 and 9.00
```

3.2 Gomoku Game

Gomoku, also called `Five in a Row`, is an abstract strategy board game. It is traditionally played with Go pieces (black and white stones) on a Go board. It can be played using the 15×15 board or the 19×19 board. Players alternate turns placing a stone of their color on an empty intersection. The winner is the first player to form an unbroken chain of `five` stones `horizontally`, `vertically`, or `diagonally`.



(1) Through analysis, this game has at least three classes: `Player`, `Stone` and `Board`. In addition, design a main class named `Gomoku` to run the main game process.

(2) For `Stone` class, it mainly has a `color` attribute. The definition of the class is as follows:

```java
public class Stone {
    // The two numbers 0 and 1 are used to indicate white and black
    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;
    }

    // Sometimes it is necessary to convert color numbers into color terms
    public static String colorToString(int color) {
        return color == WHITE ? "white": "black";
    }
    // The shape information of the stone needs to be obtained when printing the
chessboard
    public char getShape() {
        return color == WHITE ? '\u25CB': '\u25CF';
    }
}
```

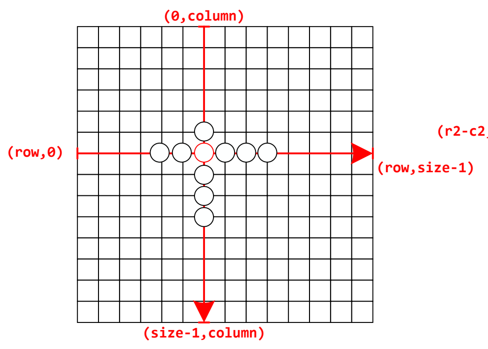
(3) The class `Player` mainly has two attributes `name` and `color` and its main behavior is to `play`. The definition of the class is as follows:

```java
import java.util.Scanner;
public class Player {
    private String name; // The name of the player
    private int color;  // The color chosen by the player

    public Player(String name, int color) {
        this.setName(name);
        this.setColor(color);
    }

    // Getter and setter methods for the two attributes name and 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;
    }

    // Enter the position to place the chess piece
    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;
        // Call the putAStone method of the class Board to place chess pieces
        boolean success = board.putAStone(row, column, new Stone(color));
        if(!success) {
            System.out.println("Illegal Input");
            // Re-entering
        }
    }
}
```

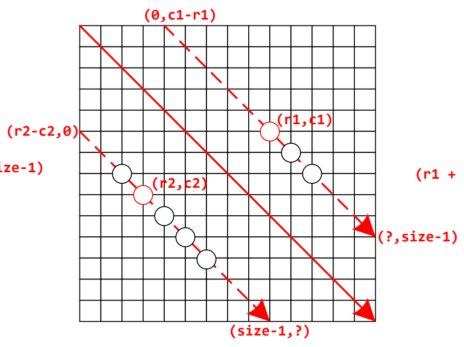
(4) The main attribute of the class `Board` is `board` which reference a two-dimensional array. The framework of the class is as follows:

```
public class Board {

    public final static int SIZE = 15;
    private Stone[][] board;
    private int remain; // How many free positions are left
    private int whichColorToPlay;
    private int winColor = -1; // No one win: -1

    // Create and initialize a chess-board
    public Board() {
        board = new Stone[SIZE][SIZE];
        remain = SIZE * SIZE;
        whichColorToPlay = (int)(Math.random() * 2);
    }

    // Display the chess-board
    public void printBoard() {
    }

    // Put a piece at position (row, column) on the chess-board
    public boolean putAStone(int row, int column, Stone stone) {
        if(...) {
            return false;
        }else {
            board[row][column] = stone;
            remain--;
            whichColorToPlay = stone.getColor() == Stone.WHITE ? Stone.BLACK:
Stone.WHITE;
            winColor = judge(row, column);
            return true;
        }
    }

    // Determine whether changes in the position (row, column) of the board will
affect the game result
    public int judge(int row, int column) {
        // The kernel used for judging: "11111" or "00000"
        String kernel = new String(new char[5]).replace("\0",
String.valueOf(board[row][column].getColor()));

        // Determine whether there are five consecutive colored stones in the
horizontal direction
        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();

        // Determine whether there are five consecutive colors of stones in the
vertical direction
        StringBuffer lineY = new StringBuffer();
        /*...*/
        if(lineY.indexOf(kernel) >= 0)
            return board[row][column].getColor();

        // Determine whether there are five consecutive colors of stones in the
main diagonal direction
        StringBuffer lineDiagA = new StringBuffer();
        /*...*/
        if(lineDiagA.indexOf(kernel) >= 0)
            return board[row][column].getColor();

        // Determine whether there are five consecutive colors of stones in the
back diagonal direction
        StringBuffer lineDiagB = new StringBuffer();
        /*...*/
        if(lineDiagB.indexOf(kernel) >= 0)
            return board[row][column].getColor();

        // There are no consecutive stones of the same color
        return -1;
    }

    /* Getter and setter methods for data fields*/
    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;
    }

}
```

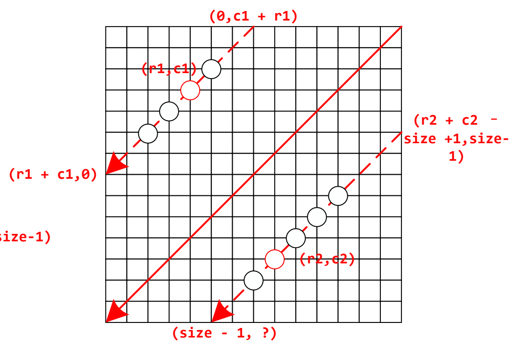
Judgment diagram：



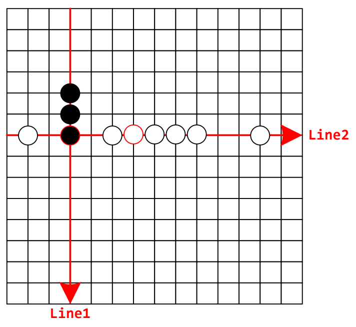
(a) Horizontal and Vertical



(b) Main Diagonal



(c) Back Diagonal



(d) Judgement Method

Line1:NNNN111NNNNNNNN
Line2:N0N1N00000NN0NN
“00000” is a sub-string of Line2
“11111” is not a sub-string of Line1

(5) The main process of the game is as follows:

```
public class Gomoku {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("***************Welcome to Gomoku
Game*****************");
        Board board = new Board();

        // Enter and display the information of two players
        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()));

        // Play the game by the two players
        while(...) {
            if(board.getWhichColorToPlay() == player1.getColor()) {
                player1.play(board, input);
            }else {
                player2.play(board, input);
            }
            board.printBoard();
        }

        if(...) {
            System.out.println("**************Game Draw***************");
        }else if(...){
            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();
    }

}
```

(6) Complete the program and debug it.

```
***************Welcome to Gomoku Game****************
Enter player1 name:xiaoming
Enter player2 name:xiaohong
Player1[xiaoming, white] VS Player2[xiaohong, black]
Player xiaohong[black] put a stone at:5 6
     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
  1  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  2  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  3  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  4  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  5  +  +  +  +  +  ●  +  +  +  +  +  +  +  +  +
  6  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  7  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  8  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  9  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
 10  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
 11  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
 12  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
 13  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
 14  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
 15  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
Player xiaoming[white] put a stone at:7 8
     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
  1  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  2  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  3  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
  4  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
 5 + + + + + ● + + + + + + + + +
 6 + + + + + + + + + + + + + + +
 7 + + + + + + + ○ + + + + + + +
 8 + + + + + + + + + + + + + + +
 9 + + + + + + + + + + + + + + +
10 + + + + + + + + + + + + + + +
11 + + + + + + + + + + + + + + +
12 + + + + + + + + + + + + + + +
13 + + + + + + + + + + + + + + +
14 + + + + + + + + + + + + + + +
15 + + + + + + + + + + + + + + +
```

4 Experiment Report Requirements

4.1 Think and answer the question

(1) When will the no-arg constructor be automatically added?

(2) What is the difference between static members and object members?

(3) What do you think is the difference between procedural-oriented programming and object-oriented programming?

(4) Other experience.

4.2 Experiment report content

(1) Answer the above questions.

(2) All codes.

4.3 Submission method

(1) Upload the report by ftp:(Address:ftp://172.18.5.102; UserName:wangxiaomeng; Password: wangxiaomeng)

(2) File name format: StudentID+Name. For example, `20191234小明.docx`

4.4 Other Instructions

You can obtain experiment course resources through the web platform (URL: https://www.lanqiao.cn; InvitationCode: `ZF0XA4Y1`)