姓名：张煜；学号：2208300002

第五章【面向对象基础】- 作业

# 谈谈你对面向对象和面向过程的理解

**面向过程**

**面向过程的编程是一种以过程为中心的编程方式。它将程序视为一系列函数或子程序的集合，每个函数完成特定的任务。这种编程方式通常强调的是“做什么”以及“怎么做”。在面向过程的语言中，数据和操作数据的过程是分开的。例如，在C语言中，数据通常存储在变量或数组中，而处理这些数据的逻辑则封装在函数里。**

**面向过程的优点在于它的简单性和直接性，对于一些小型项目或者不需要太多抽象层次的应用来说，它是一个有效率的选择。然而，随着项目的复杂度增加，维护性和可扩展性可能会成为一个问题。**

**面向对象**

**面向对象编程则是以对象为中心的编程方式。在OOP中，对象是数据（属性）及其相关行为（方法）的封装体。通过这种方式，OOP提供了更好的模块化，并且使得代码更容易理解和维护。面向对象的主要特征包括封装、继承、和抽象。**

* **封装：封装是指隐藏对象的具体实现细节，并仅暴露必要的接口或方法给外部世界。这有助于减少不同部分之间的影响，提高系统的灵活性。**
* **继承：继承允许创建新的类来继承现有类的属性和方法。这可以用来表达类之间的“is-a”关系，提高了代码的复用性。**
* **多态：多态意味着一个接口可以有多种实现方式。它允许使用一个统一的界面来表示不同的类型，从而提高了程序的灵活性。**
* **抽象：抽象是通过接口或抽象类来定义一组必须的方法签名而不提供具体的实现。它可以帮助隐藏复杂的实现细节，同时提供清晰、简单的接口供其他开发人员使用。**

1. **设计一款五子棋游戏**

请分别用面向过程和面向对象思想实现，请大致描述你的思路。

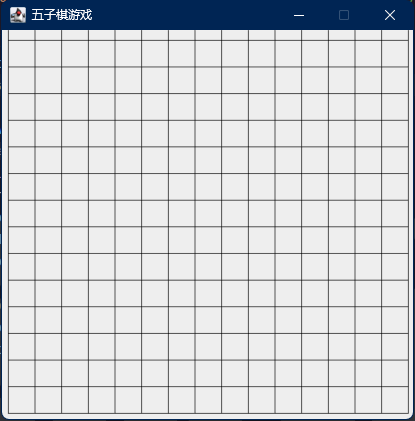
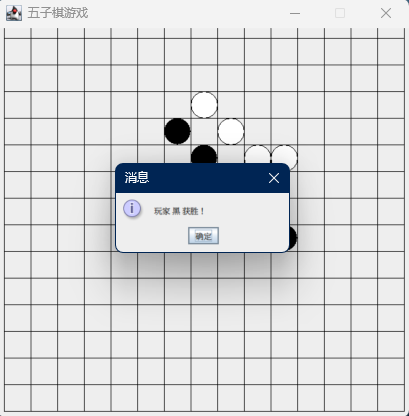
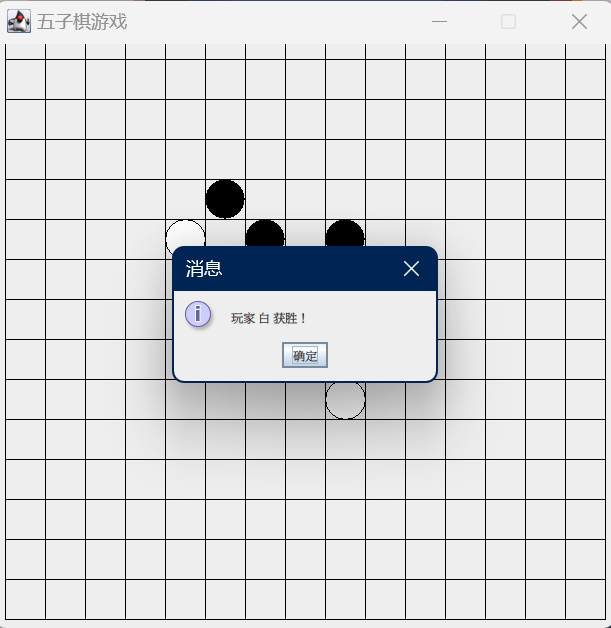
**五子棋游戏过程描述：**

游戏启动，用户点击棋盘指定位置（第几行、第几列），该位置会被放置一颗棋子（黑子先，白子后，轮流落），然后系统自动判断输赢，如果判定某方赢， 则游戏结束，如果没有赢，则继续游戏。

1. import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
  
public class FiveInARowGUI extends JFrame implements MouseListener {  
 private static final int BOARD\_SIZE = 15;  
 private static final int CELL\_SIZE = 40;  
 private static final int PADDING = 20;  
 private static final int WINDOW\_WIDTH = (CELL\_SIZE \* BOARD\_SIZE) + (2 \* PADDING);  
 private static final int WINDOW\_HEIGHT = (CELL\_SIZE \* BOARD\_SIZE) + (2 \* PADDING);  
  
 private char[][] board;  
 private Graphics2D g2d;  
 private boolean isBlackTurn = true;  
  
 public FiveInARowGUI() {  
 super("五子棋游戏");  
 setSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 setLocationRelativeTo(null);  
 setResizable(false);  
 setVisible(true);  
  
 board = new char[BOARD\_SIZE][BOARD\_SIZE];  
 initializeBoard();  
 addMouseListener(this);  
 }  
  
 private void initializeBoard() {  
 for (int i = 0; i < BOARD\_SIZE; i++) {  
 for (int j = 0; j < BOARD\_SIZE; j++) {  
 board[i][j] = ' ';  
 }  
 }  
 }  
  
 @Override public void paint(Graphics g) {  
 super.paint(g);  
 g2d = (Graphics2D) g;  
  
 drawBoard();  
 drawPieces();  
 }  
  
 private void drawBoard() {  
 g2d.setColor(Color.BLACK);  
 for (int i = 0; i <= BOARD\_SIZE; i++) {  
 g2d.drawLine(PADDING, PADDING + i \* CELL\_SIZE, PADDING + BOARD\_SIZE \* CELL\_SIZE, PADDING + i \* CELL\_SIZE);  
 g2d.drawLine(PADDING + i \* CELL\_SIZE, PADDING, PADDING + i \* CELL\_SIZE, PADDING + BOARD\_SIZE \* CELL\_SIZE);  
 }  
 }  
  
 private void drawPieces() {  
 g2d.setColor(Color.BLACK);  
 for (int i = 0; i < BOARD\_SIZE; i++) {  
 for (int j = 0; j < BOARD\_SIZE; j++) {  
 if (board[i][j] == 'B') {  
 g2d.fillOval(PADDING + j \* CELL\_SIZE, PADDING + i \* CELL\_SIZE, CELL\_SIZE, CELL\_SIZE);  
 } else if (board[i][j] == 'W') {  
 g2d.setColor(Color.WHITE);  
 g2d.fillOval(PADDING + j \* CELL\_SIZE, PADDING + i \* CELL\_SIZE, CELL\_SIZE, CELL\_SIZE);  
 g2d.setColor(Color.BLACK);  
 g2d.drawOval(PADDING + j \* CELL\_SIZE, PADDING + i \* CELL\_SIZE, CELL\_SIZE, CELL\_SIZE);  
 }  
 }  
 }  
 }  
  
 @Override public void mouseClicked(MouseEvent e) {  
 int x = e.getX() - PADDING;  
 int y = e.getY() - PADDING;  
  
 int col = x / CELL\_SIZE;  
 int row = y / CELL\_SIZE;  
  
 if (x < 0 || y < 0 || x >= BOARD\_SIZE \* CELL\_SIZE || y >= BOARD\_SIZE \* CELL\_SIZE) {  
 return;  
 }  
  
 if (board[row][col] != ' ') {  
 return;  
 }  
  
 board[row][col] = isBlackTurn ? 'B' : 'W';  
 repaint();  
  
 if (checkWin(row, col, isBlackTurn ? 'B' : 'W')) {  
 JOptionPane.showMessageDialog(this, "玩家 " + (isBlackTurn ? "黑" : "白") + " 获胜！");  
 System.exit(0);  
 }  
  
 isBlackTurn = !isBlackTurn;  
 }  
  
 @Override public void mousePressed(MouseEvent e) {}  
  
 @Override public void mouseReleased(MouseEvent e) {}  
  
 @Override public void mouseEntered(MouseEvent e) {}  
  
 @Override public void mouseExited(MouseEvent e) {}  
  
 private boolean checkWin(int row, int col, char player) {  
 return checkHorizontal(row, col, player) ||  
 checkVertical(row, col, player) ||  
 checkDiagonalLeft(row, col, player) ||  
 checkDiagonalRight(row, col, player);  
 }  
  
 private boolean checkHorizontal(int row, int col, char player) {  
 int count = 1;  
 int left = col - 1;  
 int right = col + 1;  
  
 while (left >= 0 && board[row][left] == player) {  
 count++;  
 left--;  
 }  
  
 while (right < BOARD\_SIZE && board[row][right] == player) {  
 count++;  
 right++;  
 }  
  
 return count >= 5;  
 }  
  
 private boolean checkVertical(int row, int col, char player) {  
 int count = 1;  
 int up = row - 1;  
 int down = row + 1;  
  
 while (up >= 0 && board[up][col] == player) {  
 count++;  
 up--;  
 }  
  
 while (down < BOARD\_SIZE && board[down][col] == player) {  
 count++;  
 down++;  
 }  
  
 return count >= 5;  
 }  
  
 private boolean checkDiagonalLeft(int row, int col, char player) {  
 int count = 1;  
 int leftUp = col - 1;  
 int rightDown = col + 1;  
  
 while (leftUp >= 0 && row > 0 && board[row - 1][leftUp] == player) {  
 count++;  
 leftUp--;  
 row--;  
 }  
  
 while (rightDown < BOARD\_SIZE && row < BOARD\_SIZE - 1 && board[row + 1][rightDown] == player) {  
 count++;  
 rightDown++;  
 row++;  
 }  
  
 return count >= 5;  
 }  
  
 private boolean checkDiagonalRight(int row, int col, char player) {  
 int count = 1;  
 int leftDown = col - 1;  
 int rightUp = col + 1;  
  
 while (leftDown >= 0 && row < BOARD\_SIZE - 1 && board[row + 1][leftDown] == player) {  
 count++;  
 leftDown--;  
 row++;  
 }  
  
 while (rightUp < BOARD\_SIZE && row > 0 && board[row - 1][rightUp] == player) {  
 count++;  
 rightUp++;  
 row--;  
 }  
  
 return count >= 5;  
 }  
  
 public static void main(String[] args) {  
 new FiveInARowGUI();  
 }  
}

2.

**运行结果：**

 **** 

# 谈谈你对类、对象、引用的理解

# 类 (Class)

类是一种用户定义的数据类型，它描述了一组具有相同属性（成员属性）和行为（成员函数或方法）的对象的模板或蓝图。类定义了对象的结构和行为模式。它的本质是抽象化的，仅为后期生成的对象提供属性和行为的模板。

**对象 (Object)**

对象是类的一个实例。它是根据类定义创建的具体实体，拥有该类的所有属性和行为。可以以一个类创建多个对象，他们可以相同的属性类型，不同的类型数据。

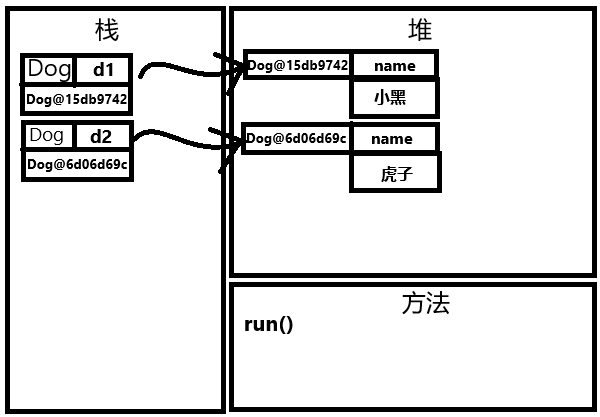
**引用 (Reference)**

引用在不同的编程语言中有不同的含义，但一般而言，它指的是指向某个对象的一个别名或指针。在面向对象的语言中，引用允许你通过一个变量来访问对象的状态和行为。例如，在Java中，你可以声明一个指向Car对象的引用变量Car myCar;，然后通过new Car()创建一个新的Car对象，并将其地址赋值给myCar引用，这样就可以通过myCar来访问对象的方法和属性了。

1. **根据下面案例，绘制对象内存图**

|  |  |  |
| --- | --- | --- |
| 1 | class Dog { |  |
| 2 | String name; |
| 3 |  |
| 4 | public void run() { |
| 5 | System.out.println(name + " | 在随风奔跑"); |
| 6 | } |  |
| 7 | } | |
| 8 |  | |
| 9 | public static void main(String[] args) { | |
| 10 | Dog d1 = new Dog("小黑"); | |

|  |  |  |
| --- | --- | --- |
| 11 |  | Dog d2 = new Dog("虎子"); |
| 12 |  |  |
| 13 |  | d1.run(); |
| 14 |  | d2.run(); |
| 15 | } |  |

****

# 面向对象基础编程

## 需求描述：

定义一个汽车类 Car

属性有：颜色、品牌、价格

行为有：前进、后退、转弯和输出对象属性

测试类实现功能：

实例化一个Car类对象，并给元素赋值调用这个类中方法，输出对象属性信息调用对象前进、后退、转弯等方法

1. class Car{

2.     String color;

3.     String brand;

4.     String price;

5.

6.     public void moveForward(){

7.         System.out.println("前进");

8.     }

9.

10.     public void back(){

11.         System.out.println("后退");

12.     }

13.

14.     public void turn(){

15.         System.out.println("转弯");

16.     }

17.

18.     public void show(){

19.         System.out.println("颜色:" + color + ";品牌:" + brand +";价格:" + price);

20.     }

21. }

22. public class TestCar{

23.     public static void main(String[] args) {

24.         Car car = new Car();

25.         car.brand = "XiaoMi";

26.         car.price = "280000" ;

27.         car.color = "海湾蓝";

28.         car.show();

29.         car.moveForward();

30.         car.back();

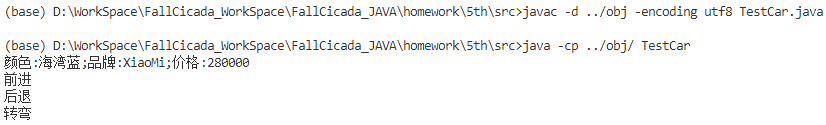
31.         car.turn();

32.     }

33. }

34.

**运行结果：**



# 谈谈你对封装的理解

**封装是面向对象编程的一个核心概念，它涉及将对象的属性和行为（方法）捆绑在一个单独的单元内。这样做的主要好处包括：**

* **隐藏内部细节：外界不需要知道对象内部是如何工作的，只需知道如何与之交互。**
* **增强安全性：通过限制直接访问内部状态，可以防止意外或恶意的数据篡改。**
* **简化接口：通过提供清晰的方法调用，使对象的使用变得简单直观。**
* **减少耦合：使对象更加独立，便于修改和重用。**

**封装是通过使用访问控制（如private、protected、public）和提供访问器（getters和setters）来实现的，目的是为了提高代码的可维护性和可扩展性。**

1. **列举构造方法注意事项**

构造方法（Constructor）是在创建对象时被调用的特殊方法。它用于初始化新创建的对象实例，并确保对象处于有效的初始状态。以下是使用构造方法时应注意的一些事项：

1. **名称匹配**：构造方法的名字必须与类名完全相同。
2. **无返回类型**：构造方法不应该有返回类型声明，即使是void也不行。这是构造方法与普通方法的一个重要区别。
3. **自动调用**：当创建一个类的新实例时，构造方法会被自动调用。如果没有显式地为类定义构造方法，编译器会自动提供一个默认的无参构造方法。
4. **链式调用**：可以使用this关键字在一个构造方法中调用另一个构造方法，但这必须是构造方法的第一条语句。
5. **super关键字**：如果一个类继承了另一个类，在子类的构造方法中必须显式调用父类的构造方法（通过super关键字）。这也是构造方法的第一条语句。
6. **初始化必填字段**：构造方法应确保对象的所有必需字段都得到适当的初始化，以保证对象的有效性。
7. **参数化构造方法**：可以根据需要定义多个构造方法，以接受不同数量或类型的参数，从而支持不同的初始化策略。
8. **异常处理**：在构造方法中进行可能抛出异常的操作时，应适当地处理这些异常，或者声明抛出这些异常。
9. **避免复杂的初始化逻辑**：构造方法应尽量保持简单，避免执行复杂的逻辑或耗时的操作。如果有必要执行复杂的初始化工作，可以考虑将这部分逻辑移到一个单独的方法中，并在构造方法完成后调用。
10. **私有构造方法**：对于不可实例化的类（如工具类或抽象类），可以将构造方法声明为私有的，以防止外部创建其实例。
11. **final构造方法**：虽然Java中不能将构造方法声明为final，但可以采用其他手段来禁止继承，比如在类声明前加上final关键字。

# 构造方法编程

定义一个人类Person

描述：属性： 姓名 （name） 、年龄 （age）； 要求：

* 1. 定义无参构造器
  2. 定义两参构造器，参数为name和age
  3. 定义一参构造器，参数为name，要求该构造器借助借助两参构造器实现功 能
  4. 定义show方法，输出类对象基本信息

答案：

1. class Person{

2.     private String name;

3.     private int age;

4.     public Person(String name){

5.         this.name = name;

6.     }

7.     public Person(String name,int age){

8.         this(name);

9.         this.age = age;

10.     }

11.     public void show(){

12.         System.out.println("姓名:" + name + ";年龄:" + age);

13.     }

14. }

15. public class Test08\_person {

16.      public static void main(String[] args) {

17.         Person p1 = new Person("jack",21);

18.         p1.show();

19.         System.out.println("------------------");

20.         Person p2 = new Person("lucy");

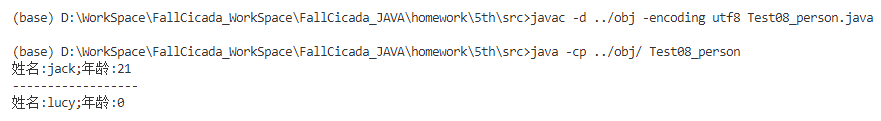
21.         p2.show();

22.      }

23. }

24.

运行结果：



# 简述this关键字的用法

* 1. **区分成员变量与局部变量** 当类的成员变量与方法中的局部变量同名时，可以通过this来明确地引用成员变量。
  2. **再有继承类时对与父类同名变量**在类中有与父类同名变量时区分与父类同名变量数据。
  3. **传递当前对象作为参数** 当一个方法需要将当前对象作为参数传递给另一个方法时，可以使用this
  4. **在构造函数中调用其他构造函数** this也可以在一个类的构造函数中调用该类的另一个构造函数，这称为构造函数重载。这种情况下，this必须是构造函数的第一条语句
  5. **返回当前对象实例** 在一些设计模式中，如链式调用，this被用来返回当前对象实例，使得可以在方法调用之间连续添加更多的方法调用

1. **程序阅读题**

下面代码无法通过编译，要求：

修改代码使得程序能够正常运行分析其输出结果

提示：该题主要用于考核this关键字的作用

答案：

1. public class Test10\_This {

2.     public static void main(String[] args) {

3.         Teacher t = new Teacher();

4.         t.show();

5.         System.out.println("-----------");

6.         t.disp();

7.     }

8. }

9.

10. class Teacher {

11.     // 初始化成员变量

12.     private String name = "unknown";

13.     private int age = 0;

14.     private String gender = "男";

15.

16.     public void show() {

17.         // 局部变量与成员变量同名，这里使用局部变量

18.         String name = "Tom";

19.         int age = 20;

20.         System.out.println("name：" + name); // 打印局部变量

21.         System.out.println("age：" + age);   // 打印局部变量

22.         System.out.println("gender：" + gender); // 打印成员变量

23.     }

24.

25.     public void disp() {

26.         name = "jack";

27.         age = 18;

28.         // 使用 this 关键字来区分成员变量与方法内的局部变量

29.         System.out.println("this.age：" + this.age); // 打印成员变量

30.         System.out.println("this.name：" + this.name); // 打印成员变量

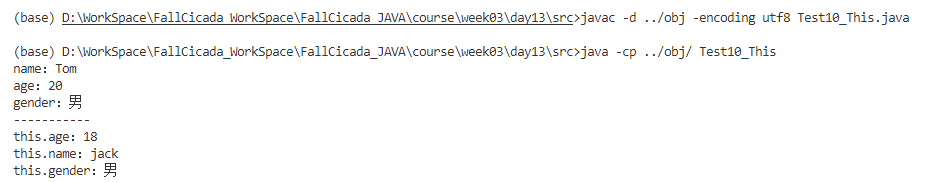
31.         System.out.println("this.gender：" + this.gender); // 打印成员变量

32.     }

33. }

34.

**运行结果：**



# 根据要求编写时间类Duration

时间类Duration 具体要求如下：

* 1. 包含属性hour、min、sec，都是int类型
  2. 提供无参构造器
  3. 提供构造器（三参），对hour、min、sec进行初始化
  4. 提供构造器（一参），参数含义：总秒数int seconds，对hour、min、sec进行初始化，例如3661秒，转为a小时b分钟c秒，就是1小时1分钟1秒
  5. 提供每个属性的get方法
  6. 提供getTotalSeconds()方法，用于返回hour小时min分钟sec秒钟对应的总秒 数seconds
  7. 提供disp()方法，输出对象的属性信息和总秒数测试类：

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

package com.briup.chap05;

public class Test11\_Duration {

public static void main(String[] args) {

//借助三参构造器 实例化对象

Duration d1 = new Duration(1,1,1);

//获取总秒数

int totalSeconds = d1.getTotalSeconds(); System.out.println("1小时1分钟1秒：" + totalSeconds);

System.out.println(" ");

//借助单参构造器 实例化对象

Duration d2 = new Duration(3660);

//获取小时

int hour = d2.getHour(); System.out.println("hour = " + hour);

System.out.println(" ");

//输出对象属性信息及总描述信息d2.disp();

}

}

答案：

1. public class DurationTest{

2.     public static void main(String[] args) {

3.         Duration duration = new Duration(3661);

4.         duration.disp();

5.

6.     }

7. }

8. public class Duration {

9.     int hour;

10.     int min;

11.     int sec;

12.

13.     public Duration() {

14.

15.     }

16.

17.     public Duration(int seconds) {

18.         int hour = seconds/3600;

19.         int min = (seconds%3600)/60;

20.         int sec = (seconds%3600)%60;

21.         this.hour = hour;

22.         this.min = min;

23.         this.sec = sec;

24.     }

25.

26.     public Duration(int hour, int min, int sec) {

27.         this.hour = hour;

28.         this.min = min;

29.         this.sec = sec;

30.     }

31.

32.     public int getHour() {

33.         return hour;

34.     }

35.

36.     public int getMin() {

37.         return min;

38.     }

39.

40.     public int getSec() {

41.         return sec;

42.     }

43.

44.     public int getTotalSeconds() {

45.         return hour\*3600 + min\*60 + sec;

46.     }

47.

48.     public void disp() {

49.         int second = hour\*3600 + min\*60 + sec;

50.         System.out.println(hour + "小时;" + min + "分钟;" + sec + "秒");

51.         System.out.println("总秒数:" + second);

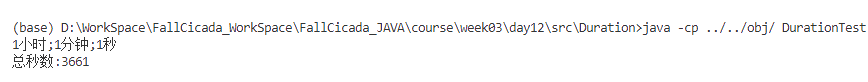
52.     }

53.

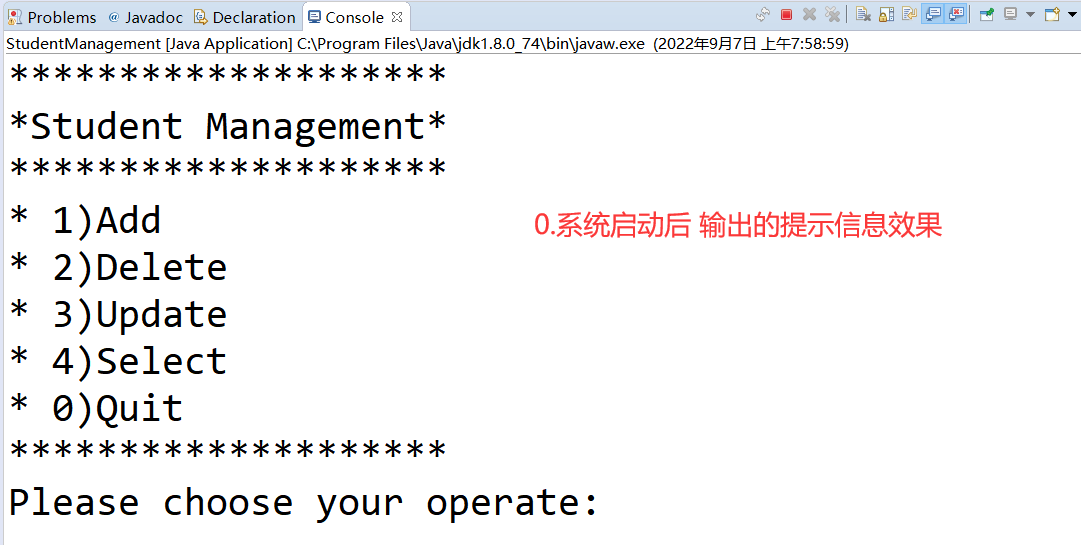
54. }

55.

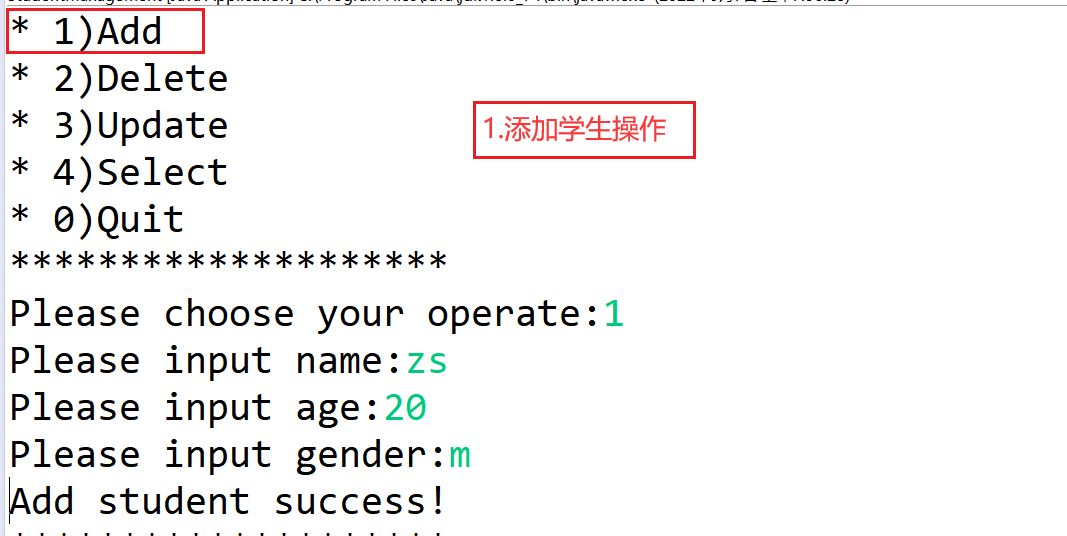
**运行结果：**



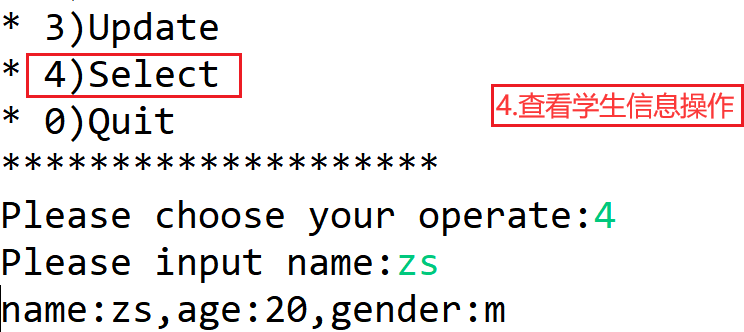
# 扩展题-学生管理系统

现有一个简单的学生管理系统，系统启动后运行效果如下图： 系统启动后提示界面如下图：

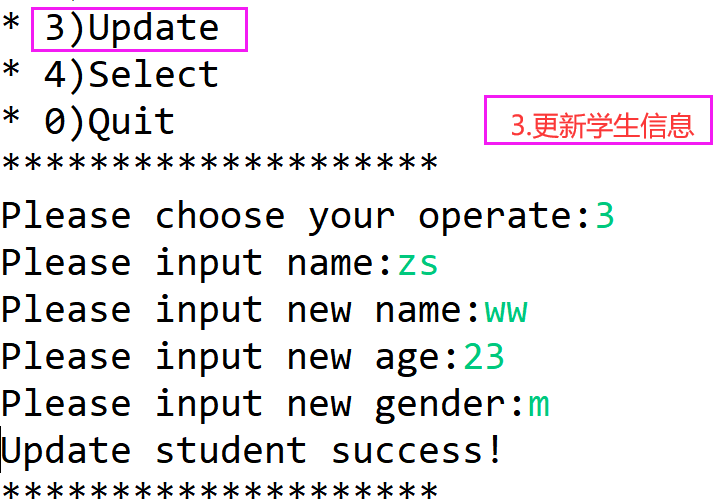
添加学生操作步骤如下：



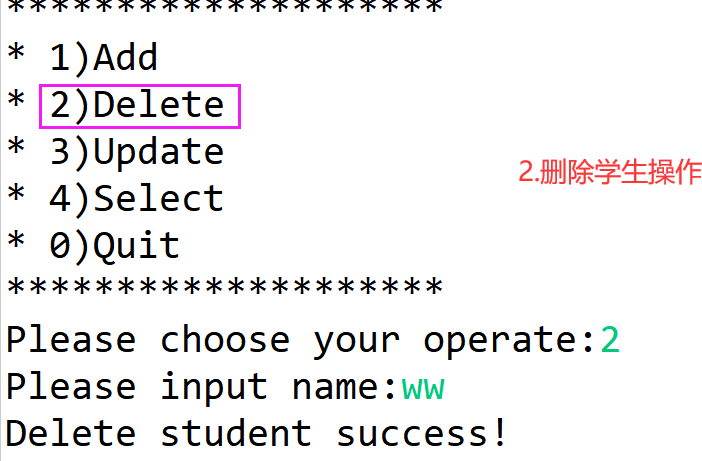
查看学生操作步骤如下：



更新学生操作步骤如下：



删除学生操作步骤如下：



其中学生类Student，学生管理类 StudentManagement代码已经给出，但代码不完整！

请补全代码，使得程序能够实现完整上述完整功能：

1. package staudent;

2.

3. /\*

4.  \* @Description:

5.  \* @Author: FallCicada

6.  \* @Date: 2024-09-11 19:06:01

7.  \* @LastEditors: FallCicada

8.  \* @LastEditTime: 2024-09-11 19:12:19

9.  \*/

10. import java.util.Scanner;

11.

12. // 学生管理系统类

13. public class StudentManagement {

14.     // 学生对象数组

15.     private Student[] stus;

16.     // 实际学生个数

17.     private int counter;

18.     // 键盘录入对象

19.     private Scanner in;

20.

21.     // 无参构造器：默认容量100

22.     public StudentManagement() {

23.         stus = new Student[100];

24.         in = new Scanner(System.in);

25.     }

26.

27.     // 有参构造器：size表示初始容量

28.     public StudentManagement(int size) {

29.         stus = new Student[size];

30.         in = new Scanner(System.in);

31.     }

32.

33.     // 输出 操作提示

34.     public void prompt() {

35.         System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

36.         System.out.println("\*Student Management\*");

37.         System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

38.         System.out.println("\* 1)Add");

39.         System.out.println("\* 2)Delete");

40.         System.out.println("\* 3)Update");

41.         System.out.println("\* 4)Select");

42.         System.out.println("\* 0)Quit");

43.         System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

44.         System.out.print("Please choose your operate:");

45.     }

46.

47.     // 根据用户录入的学生名称，找到学生在数组中的索引返回

48.     private int findStudent() {

49.         int index = -1;

50.         String name;

51.         System.out.print("Please input name:");

52.         // 从键盘获取一个字符串

53.         name = in.next();

54.         for (int i = 0; i < counter; i++) {

55.             if (stus[i].getName().equals(name)) {

56.                 index = i;

57.                 break;

58.             }

59.         }

60.         return index;

61.     }

62.

63.     // 增加学生

64.     public void addStudent() {

65.         String name;

66.         int age;

67.         String gender;

68.         Student s;

69.         // 判断 管理系统容量是否已满

70.         if (counter >= stus.length) {

71.             System.out.println("Add failure, too many students!");

72.             return;

73.         }

74.         System.out.print("Please input name:");

75.         name = in.next();

76.         System.out.print("Please input age:");

77.         age = in.nextInt();

78.         System.out.print("Please input gender:");

79.         gender = in.next();

80.         s = new Student(name, age, gender);

81.         // 关键代码

82.         stus[counter] = s;

83.         counter++;

84.         System.out.println("Add student success!");

85.     }

86.

87.     // 删除学生

88.     public void deleteStudent() {

89.         int index = findStudent();

90.         if (index == -1) {

91.             System.out.println("Student not found!");

92.             return;

93.         }

94.         for (int i = index; i < counter - 1; i++) {

95.             stus[i] = stus[i + 1];

96.         }

97.         stus[--counter] = null;

98.         System.out.println("Delete student success!");

99.     }

100.

101.     // 更新学生信息

102.     public void updateStudent() {

103.         int index = findStudent();

104.         if (index == -1) {

105.             System.out.println("Student not found!");

106.             return;

107.         }

108.         System.out.print("Please input new name:");

109.         String newName = in.next();

110.         System.out.print("Please input new age:");

111.         int newAge = in.nextInt();

112.         System.out.print("Please input new gender:");

113.         String newGender = in.next();

114.         stus[index].setName(newName);

115.         stus[index].setAge(newAge);

116.         stus[index].setGender(newGender);

117.         System.out.println("Update student success!");

118.     }

119.

120.     // 查询学生信息

121.     public void selectStudent() {

122.         int index = findStudent();

123.         if (index == -1) {

124.             System.out.println("Student not found!");

125.             return;

126.         }

127.         System.out.println("Name: " + stus[index].getName());

128.         System.out.println("Age: " + stus[index].getAge());

129.         System.out.println("Gender: " + stus[index].getGender());

130.     }

131.

132.     public static void main(String[] args) {

133.         // 实例化对象

134.         StudentManagement sm = new StudentManagement();

135.         // 用户操作标志

136.         int option = -1;

137.         Loop: while (true) {

138.             // 打印提示信息

139.             sm.prompt();

140.             // 获取用户操作

141.             option = sm.in.nextInt();

142.             if (option < 0 || option > 4)

143.                 continue;

144.             switch (option) { // byte char short int

145.                 case 1:

146.                     sm.addStudent();

147.                     break;

148.                 case 2:

149.                     sm.deleteStudent();

150.                     break;

151.                 case 3:

152.                     sm.updateStudent();

153.                     break;

154.                 case 4:

155.                     sm.selectStudent();

156.                     break;

157.                 case 0:

158.                     break Loop;

159.             } // end switch

160.         } // end while

161.         System.out.println("Game over, byebye");

162.     } // end main

163. }

164.

165. class Student {

166.     // 属性

167.     private String name;

168.     private int age;

169.     private String gender;

170.

171.     // 默认|无参构造器

172.     public Student() {}

173.

174.     // 有参构造器

175.     public Student(String name, int age, String gender) {

176.         this.name = name;

177.         this.age = age;

178.         this.gender = gender;

179.     }

180.

181.     // getter/setter 方法

182.     public String getName() {

183.         return name;

184.     }

185.

186.     public void setName(String name) {

187.         this.name = name;

188.     }

189.

190.     public int getAge() {

191.         return age;

192.     }

193.

194.     public void setAge(int age) {

195.         this.age = age;

196.     }

197.

198.     public String getGender() {

199.         return gender;

200.     }

201.

202.     public void setGender(String gender) {

203.         this.gender = gender;

204.     }

205. }

206.

Cmd：

1. (base) d:\WorkSpace\FallCicada\_WorkSpace>

2. (base) d:\WorkSpace\FallCicada\_WorkSpace> d: && cd d:\WorkSpace\FallCicada\_WorkSpace && cmd /C ""C:\Program Files\Java\jdk1.8.0\_241\bin\java.exe" -cp C:\Users\86138\AppData\Roaming\Code\User\workspaceStorage\5e64c956c6ef5aa4f47184cf76ebe205\redhat.java\jdt\_ws\FallCicada\_WorkSpace\_1526d045\bin staudent.StudentManagement "

3. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4. \*Student Management\*

5. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6. \* 1)Add

7. \* 2)Delete

8. \* 3)Update

9. \* 4)Select

10. \* 0)Quit

11. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

12. Please choose your operate:1

13. Please input name:mr

14. Please input age:9

15. Please input gender:m

16. Add student success!

17. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18. \*Student Management\*

19. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20. \* 1)Add

21. \* 2)Delete

22. \* 3)Update

23. \* 4)Select

24. \* 0)Quit

25. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

26. Please choose your operate:4

27. Please input name:mr

28. Name: mr

29. Age: 9

30. Gender: m

31. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

32. \*Student Management\*

33. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

34. \* 1)Add

35. \* 2)Delete

36. \* 3)Update

37. \* 4)Select

38. \* 0)Quit

39. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

40. Please choose your operate:2

41. Please input name:mr

42. Delete student success!

43. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

44. \*Student Management\*

45. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

46. \* 1)Add

47. \* 2)Delete

48. \* 3)Update

49. \* 4)Select

50. \* 0)Quit

51. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

52. Please choose your operate:0

53. Game over, byebye

54.