1.杨辉三角

```
import java.util.Scanner;

public class PascalTriangle {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 输入生成杨辉三角的行数
        System.out.print("请输入杨辉三角的行数: ");
        int rows = scanner.nextInt();

        // 创建二维数组存储杨辉三角
        int[][] triangle = new int[rows][rows];

        // 生成杨辉三角
        for (int i = 0; i < rows; i++) {
            triangle[i][0] = 1; // 每行的第一个元素是1
            for (int j = 1; j <= i; j++) {
                // 当前元素等于上一行的左上和正上方之和
                triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
            }
        }

        // 打印杨辉三角
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j <= i; j++) {
                System.out.print(triangle[i][j] + " ");
            }
            System.out.println();
        }

        scanner.close();
    }
}
```

2.多线程存钱

```
import java.util.Random;

class BankAccount {
    private int balance = 0; // 初始余额为0

    // 同步方法，确保线程安全
    public synchronized void deposit(int amount) {
        balance += amount;
        System.out.println(Thread.currentThread().getName() + " 存入: " + amount +
" 当前余额: " + balance);
    }
}

class DepositTask implements Runnable {
    private final BankAccount account;
```

```
    public DepositTask(BankAccount account) {
        this.account = account;
    }

    @Override
    public void run() {
        Random random = new Random();
        for (int i = 0; i < 5; i++) { // 每个线程存钱5次
            int amount = random.nextInt(100) + 1; // 随机金额 [1, 100]
            account.deposit(amount);
            try {
                Thread.sleep(100); // 模拟存款操作的延迟
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class BankSystem {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(); // 创建银行账户

        // 创建两个存钱线程
        Thread t1 = new Thread(new DepositTask(account), "线程1");
        Thread t2 = new Thread(new DepositTask(account), "线程2");

        // 启动线程
        t1.start();
        t2.start();
    }
}
```

（1）使用Arr记录一个student对象，三个属性(姓名，学号，成绩)和两个方法：查询成绩和比较compare成绩

（2）编写一个排序算法进行成绩的排序

```
import java.util.Arrays;

class Student {
    String name;  // 姓名
    int id;       // 学号
    int score;    // 成绩

    // 构造方法
    public Student(String name, int id, int score) {
        this.name = name;
        this.id = id;
        this.score = score;
    }

    // 查询成绩方法
    public int getScore() {
```

```
        return score;
    }

    // 比较成绩方法
    public int compare(Student other) {
        return Integer.compare(this.score, other.score); // 返回成绩差值
    }

    @Override
    public String toString() {
        return "姓名: " + name + ", 学号: " + id + ", 成绩: " + score;
    }
}

public class StudentSortExample {
    public static void main(String[] args) {
        // 创建学生数组
        Student[] students = {
            new Student("Alice", 1001, 85),
            new Student("Bob", 1002, 92),
            new Student("Charlie", 1003, 78),
            new Student("David", 1004, 88)
        };

        System.out.println("排序前: ");
        printArray(students);

        // 调用排序方法进行成绩排序
        sortStudentsByScore(students);

        System.out.println("\n按成绩排序后: ");
        printArray(students);
    }

    // 排序方法：冒泡排序
    public static void sortStudentsByScore(Student[] students) {
        int n = students.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (students[j].compare(students[j + 1]) > 0) { // 如果前者成绩大于
后者，交换
                    Student temp = students[j];
                    students[j] = students[j + 1];
                    students[j + 1] = temp;
                }
            }
        }
    }

    // 打印数组内容
    public static void printArray(Student[] students) {
        for (Student student : students) {
            System.out.println(student);
        }
    }
}
```