Java课后题

面向过程（2,3章）

1. 利用 `Scanner` 输入正整数 `n`，计算多项式 `1! + 2! + 3! + … + n!`，如果多项式之和超过 `2000` 时需中止后续项的相加操作，并输出累加之和以及停止时累加项（`a!`）的 `a` 值。

输出格式参考：

```
System.out.printf("the sum is %d, and the last item is %d", sum, i);
```

代码：

```
import java.util.Scanner;

public class Chapter32 {
    public static void main(String[] args) {
        Scanner sc3 = new Scanner(System.in);

        int n = sc3.nextInt();
        int i, sum = 0, front = 1, count = 0;

        for (i = 1; i <= n; i++) {
            front *= i;
            sum += front;
            count++;
            if (sum >= 2000)
                break;
        }

        System.out.printf("the sum is %d, and the last item is %d", sum, count);
        sc3.close();
    }
}
```

---

2. 通过 `Scanner` 输入两个正整数，利用辗转相除法（欧几里得算法）求两个正整数的最大公约数

代码：

```
import java.util.Scanner;

public class Chapter33 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int num1 = scanner.nextInt();
        int num2 = scanner.nextInt();
```

```
        if (num1 < num2) {
            int temp = num1;
            num1 = num2;
            num2 = temp;
        }

        int gcd = findGCD(num1, num2);
        System.out.println("最大公约数是: " + gcd);
        scanner.close();
    }

    // 使用辗转相除法求最大公约数
    public static int findGCD(int num1, int num2) {
        while (num2 != 0) {
            int temp = num1 % num2;
            num1 = num2;
            num2 = temp;
        }
        return num1;
    }
}
```

---

3. (程序题) 从键盘输入一个字符，用程序来判断这个字符是属于数字，西文字母还是其他字符。

用例 1:

- 输入 `s` 输出 `s 是西文字符`

用例 2:

- 输入 `4` 输出 `4 是数字`

用例 3:

- 输入 `中` 输出 `中 是其他字符`

用例 4:

- 输入 `!` 输出 `! 是其他字符`

代码:

```
import java.util.Scanner;

public class Chapter36 {
    public static void main(String[] args) {
        Scanner sc4 = new Scanner(System.in);

        char str = sc4.next().charAt(0);

        if ((str >= 65 && str <= 90) || (str >= 97 && str <= 122))
            System.out.println(str + " 是西文字符");
        else if (str >= 48 && str <= 57)
            System.out.println(str + " 是数字");
```

```
        else
            System.out.println(str + " 是其他字符");

        sc4.close();
    }
}
```

---

4. (程序题) 输入一个数组，其中只有一个数出现奇数次，其他数均出现偶数次，用异或运算找出该数。

用例 1:

- 输入：`5 2 3 6 3 2`
- 输出：`奇数次出现的数为：6`

用例 2:

- 输入：`7 3 3 3 3 4 4 5`
- 输出：`奇数次出现的数为：5`

代码:

```
import java.util.Scanner;

public class Chapter38 {
    public static void main(String[] args) {
        Scanner sc4 = new Scanner(System.in);

        int n = sc4.nextInt();
        int res[] = new int[n];
        int result = 0;

        for (int i = 0; i < n; i++) {
            res[i] = sc4.nextInt();
            result ^= res[i];
        }

        System.out.println("奇数次出现的数为：" + result);
        sc4.close();
    }
}
```

面向对象程序设计（4,5章）

1. (填空题) 第五章课后第二题

代码:

```
class sup {}

public class sub extends sup {
    public static void main(String[] args) {
        sub sb1 = new sub();
        sup sp1 = new sub();
        sup sp2 = new sup();

        System.out.println("sp1 instanceof sub: " + (sp1 instanceof sub));
        System.out.println("sp2 instanceof sub: " + (sp2 instanceof sub));
    }
}
```

---

2. (填空题) 第五章课后第3题

代码:

```
class A {
    public String Show(D obj) {
        return "A and D";
    }
    public String Show(A obj) {
        return "A and A";
    }
}

class B extends A {
    public String Show(B obj) {
        return "B and B";
    }
    public String Show(A obj) {
        return "B and A";
    }
}

class C extends B {
    public String Show(C obj) {
        return "C and C";
    }
    public String Show(B obj) {
        return "C and B";
    }
}

class D extends B {
    public String Show(D obj) {
        return "D and D";
    }
    public String Show(B obj) {
        return "D and B";
    }
}
```

```
public class mainTest {
    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new B();
        B b = new B();
        C c = new C();
        D d = new D();

        System.out.println(a1.Show(b));
        System.out.println(a1.Show(c));
        System.out.println(a1.Show(d));
        System.out.println(a2.Show(b));
        System.out.println(a2.Show(c));
        System.out.println(a2.Show(d));
        System.out.println(b.Show(b));
        System.out.println(b.Show(c));
        System.out.println(b.Show(d));
    }
}
```

---

3. 第五章课后第4题

代码:

```
class Father {
    public void Show(Father obj) {
        System.out.println("in Father.show-Father");
    }
}

class Son extends Father {
    public void Show(Son obj) {
        System.out.println("in Son.show-Son");
    }
    public void Show(Father obj) {
        System.out.println("in Son.show-Father");
    }
    public void Show(GrandSon obj) {
        System.out.println("in Son.show-GrandSon");
    }
}

class GrandSon extends Son {}

public class mainTest1 {
    public static void main(String[] args) {
        Father f2 = new Son();
        GrandSon gs1 = new GrandSon();
        f2.Show(gs1); // 引用类型确定函数，然后动态绑定该函数
    }
}
```

第五章课后第5题

代码：

```
class superc {
    int i = 5;

    void show() {
        System.out.println("the i is :" + i);
    }
}

public class subc extends superc {
    int i = 6;

    public static void main(String[] args) {
        subc s = new subc();

        System.out.println(s.i); // 子类的 i
        s.show(); // 调用父类定义的 show 方法
    }
}
```

第五章课后第6题

代码：

```
class Base {
    private String name = "base";

    public Base() {
        tellName(); // 调用可能被子类重写的方法
    }

    public void tellName() {
        System.out.println("Base tell name: " + name);
    }
}

public class Dervied extends Base {
    private String name = "dervied";

    public Dervied() {
        tellName();
    }

    public void tellName() {
        System.out.println("Dervied tell name: " + name);
    }

    public static void main(String[] args) {
        new Dervied();
    }
```

```
}
```

---

6. (填空题) 编写程序，声明一个 `Student` 类，找出总成绩最高的学生和数学成绩最低的学生。

代码：

```
class Student {
    private int id;
    private String name;
    private int english;
    private int math;
    private int computer;

    public Student(int id, String name, int english, int math, int computer) {
        this.id = id;
        this.name = name;
        this.english = english;
        this.math = math;
        this.computer = computer;
    }

    public int getTotalScore() {
        return english + math + computer;
    }

    public int getMathScore() {
        return math;
    }

    public String getName() {


        return name;
    }
}

public class Main {
    public static void main(String[] args) {
        Student[] students = {
            new Student(1, "Alice", 85, 92, 78),
            new Student(2, "Bob", 88, 75, 91),
            new Student(3, "Charlie", 90, 88, 84)
        };

        Student highestTotal = students[0];
        Student lowestMath = students[0];

        for (Student s : students) {
            if (s.getTotalScore() > highestTotal.getTotalScore()) {
                highestTotal = s;
            }
            if (s.getMathScore() < lowestMath.getMathScore()) {
```

```
                lowestMath = s;
            }
        }

        System.out.println("总成绩最高的学生是：" + highestTotal.getName());
        System.out.println("数学成绩最低的学生是：" + lowestMath.getName());
    }
}
```

6-10章

1. 文件读写方法：利用字节输入流、字节字符转换流、缓冲区流实现纯文本文件内容的复制

代码：

```
import java.io.*;

public class FileCopy {
    public static void main(String[] args) {
        String sourceFile = "source.txt";
        String destFile = "destination.txt";

        try (BufferedReader br = new BufferedReader(new FileReader(sourceFile));
             BufferedWriter bw = new BufferedWriter(new FileWriter(destFile))) {

            String line;
            while ((line = br.readLine()) != null) {
                bw.write(line);
                bw.newLine();
            }

            System.out.println("文件复制完成！");
        } catch (IOException e) {
            System.out.println("发生错误：" + e.getMessage());
        }
    }
}
```

---

2. 多线程编程（第九章，习题4.1）：编写一个多线程程序，演示两个人同时操作一个银行账户，一个人存钱，一个人取钱

代码：

```
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class BankAccount {
    private int balance = 0;
    private final Lock lock = new ReentrantLock();
```

```
    public void deposit(int amount) {
        lock.lock();
        try {
            balance += amount;
            System.out.println(Thread.currentThread().getName() + " 存入: " +
amount + ", 当前余额: " + balance);
        } finally {
            lock.unlock();
        }
    }

    public void withdraw(int amount) {
        lock.lock();
        try {
            if (balance >= amount) {
                balance -= amount;
                System.out.println(Thread.currentThread().getName() + " 取出: " +
amount + ", 当前余额: " + balance);
            } else {
                System.out.println(Thread.currentThread().getName() + " 取款失败,
余额不足! ");
            }
        } finally {
            lock.unlock();
        }
    }
}

public class BankOperation {
    public static void main(String[] args) {
        BankAccount account = new BankAccount();

        Thread depositor = new Thread(() -> {
            for (int i = 0; i < 5; i++) {
                account.deposit(100);
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }, "存款线程");

        Thread withdrawer = new Thread(() -> {
            for (int i = 0; i < 5; i++) {
                account.withdraw(50);
                try {
                    Thread.sleep(150);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }, "取款线程");

        depositor.start();
        withdrawer.start();
```

```
    }
}
```

---

3. 图形化编程：设计一个一元二次方程类，通过输入三个参数初始化方程对象，并提供方程求解方法，考虑用户的非法输入，并给出相应的异常处理。

代码：

```
import java.util.Scanner;

class QuadraticEquation {
    private double a, b, c;

    public QuadraticEquation(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public void solve() {
        double discriminant = b * b - 4 * a * c;
        if (discriminant > 0) {
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("方程有两个实根: " + root1 + " 和 " + root2);
        } else if (discriminant == 0) {
            double root = -b / (2 * a);
            System.out.println("方程有一个实根: " + root);
        } else {
            System.out.println("方程无实根！");
        }
    }
}

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

        try {
            System.out.print("请输入参数 a: ");
            double a = scanner.nextDouble();
            if (a == 0) {
                throw new IllegalArgumentException("参数 a 不能为 0！");
            }

            System.out.print("请输入参数 b: ");
            double b = scanner.nextDouble();

            System.out.print("请输入参数 c: ");
            double c = scanner.nextDouble();

            QuadraticEquation equation = new QuadraticEquation(a, b, c);
```

```
            equation.solve();

        } catch (Exception e) {
            System.out.println("输入错误: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

---

4. 网络读写：利用 TCP 协议实现一对一的客户端/服务器编程。客户端通过标准输入端循环输入数据，服务器获取数据后显示在本地屏幕上，同时回传 "ok" 给客户端。

服务器端：

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(9999)) {
            System.out.println("服务器已启动，等待客户端连接...");
            Socket socket = serverSocket.accept();
            System.out.println("客户端已连接！");

            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));

            String message;
            while ((message = in.readLine()) != null) {
                System.out.println("收到客户端消息: " + message);
                out.write("ok\n");
                out.flush();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

客户端：

```
import java.io.*;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) {
```

```
        try (Socket socket = new Socket("localhost", 9999)) {
            System.out.println("已连接到服务器! ");

            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
            Scanner scanner = new Scanner(System.in);

            String message;
            while (true) {
                System.out.print("请输入消息（输入exit退出）: ");
                message = scanner.nextLine();
                if ("exit".equalsIgnoreCase(message)) {
                    break;
                }

                out.write(message + "\n");
                out.flush();
                System.out.println("服务器回复: " + in.readLine());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```