JAVA

A卷：

一、

C,B,B,A,B

二、

enum，

import,

thread

三：1. 如何理解面向对象和面向过程？

- **面向过程（Procedural Programming）**：
  - 主要关注程序的步骤和流程，通过函数和过程来进行代码组织。
  - 代码通常是线性的，数据和函数分开，强调算法的实现。
  - 优点包括易于理解和实现简单问题，但在面对复杂系统时可维护性差。
- **面向对象（Object-Oriented Programming, OOP）**：
  - 强调通过对象来组织代码，对象是数据和操作这些数据的函数的封装体。
  - 通过类和对象的概念，使代码更易于重用和扩展。
  - 提供了封装、继承和多态等特性，有助于管理复杂性，提高代码的可维护性和可扩展性。

2. 请解释 Java 中的继承是什么，以及为什么它对面向对象编程很重要？

- **继承（Inheritance）**：
  - 继承是面向对象编程中的一个核心概念，允许一个类（子类）继承另一个类（父类）的属性和方法。
  - 在 Java 中，可以使用 `extends` 关键字实现继承。
- **重要性**：
  - **代码重用**：继承使得子类可以重用父类的代码，减少重复的代码。
  - **层次结构**：支持创建类的层次结构，使得设计和管理复杂系统变得更加高效。
  - **多态性**：通过继承实现多态，允许子类对象在父类引用中使用，增强了代码的灵活性。

3. 三点 JAVA 语言相较 Python 语言的优势和劣势

- **优势**：
  1. **性能**：Java 是编译型语言，通常比 Python 的解释性执行速度快。
  2. **强类型**：Java 是静态类型语言，类型检查在编译时完成，有助于捕捉错误。
  3. **多线程支持**：Java 内建对多线程的支持，适合高并发的应用程序。
- **劣势**：
  1. **语法复杂性**：Java 的语法相对较复杂，代码量较大，初学者学习曲线较陡。
  2. **开发速度**：由于需要更多的样板代码，Java 的开发速度通常较慢。
  3. **内存管理**：尽管 Java 有垃圾回收机制，但在某些情况下，内存管理仍不如 Python 来得灵活和简单。

四:

1.

```
public class TripleArray {
    public static void main(String[] args) {
        int[] array = new int[8]; // 创建一个长度为 8 的数组

        for (int i = 0; i < array.length; i++) {
            array[i] = i * 3; // 第 i 个元素是 i 的三倍
        }

        // 打印输出数组
        for (int num : array) {
            System.out.println(num);
        }
    }
}
```

2.

```
import java.util.Scanner;

public class FibonacciSeries {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("请输入要生成的斐波那契数列的数量 n: ");
        int n = scanner.nextInt();

        int a = 0, b = 1;
        System.out.print("斐波那契数列: ");
        for (int i = 0; i < n; i++) {
            System.out.print(a + " ");
            int next = a + b;
            a = b;
            b = next;
        }
    }
}
```

3.

```
import java.util.Scanner;

class Employee {
    private String name;
    private int age;
    private String position;
    private double salary;

    // 默认构造器，给所有属性指定初始值
    public Employee() {
        this.name = "";
        this.age = 18;
```

```
        this.position = "售后服务";
        this.salary = calculateSalary(age);
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setAge(int age) {
        if (age < 18) {
            this.age = 18; // 年龄无效时，强制赋值为18
        } else {
            this.age = age;
        }
        this.salary = calculateSalary(this.age); // 更新工资
    }

    public void setPosition(String position) {
        if ("售后服务".equals(position) || "销售员".equals(position)) {
            this.position = position;
        } else {
            this.position = "售后服务"; // 不符合要求强制赋值
        }
    }

    public void setSalary(double salary) {
        this.salary = salary; // 实际不需要外部设置，工资根据年龄计算
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public String getPosition() {
        return position;
    }

    public double getSalary() {
        return salary;
    }

    // 根据年龄段计算工资
    private double calculateSalary(int age) {
        if (age >= 18 && age <= 20) return 1000;
        else if (age >= 21 && age <= 25) return 1500;
        else if (age >= 26 && age <= 30) return 2000;
        else if (age >= 31 && age <= 40) return 3000;
        else if (age >= 41 && age <= 50) return 3500;
        else return 4000; // 50岁以上
    }
}

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

```
        Scanner scanner = new Scanner(System.in);
        Employee[] employees = new Employee[3];

        for (int i = 0; i < 3; i++) {
            employees[i] = new Employee(); // 创建新员工对象
            System.out.print("请输入员工姓名: ");
            employees[i].setName(scanner.nextLine());
            System.out.print("请输入员工年龄: ");
            employees[i].setAge(scanner.nextInt());
            scanner.nextLine(); // 清空扫描器
            System.out.print("请输入员工职位（售后服务/销售员）: ");
            employees[i].setPosition(scanner.nextLine());
            System.out.println(); // 空行
        }

        // 显示所有员工的信息
        System.out.println("员工信息:");
        for (Employee employee : employees) {
            System.out.println("姓名: " + employee.getName());
            System.out.println("年龄: " + employee.getAge());
            System.out.println("职位: " + employee.getPosition());
            System.out.println("工资: " + employee.getSalary());
            System.out.println(); // 空行
        }
    }
}
```