学号	姓名	提交时间	总分	成绩
20225102010214	孙正航	2024-03-28 22:14:07	100	90

1.【简答题】

得分：90/100

作业：
1、修饰方法和属性的常用修饰符有哪些？作用是啥？
2、构造方法的特点？
3、要想调用一个类下的方法和属性需要做什么？
4、静态修饰的属性和方法的特点
5、静态语句块的结构与特点
6、封装的特点
7、继承的关键字
8、局部变量和成员变量的区别
9、堆和栈结构特点
10、面向对象的特点
编程题： 编写一个计算器的类，
        计算器类根据用户类输入的+-*/符号和运算数，
        返回相应的运算结果

学生答案：1. 修饰方法和属性的常用修饰符包括 public、private、protected、static、final 等。它们的作用如下：
- public：表示该方法或属性可以被其他类访问。
- private：表示该方法或属性只能在本类中访问。
- protected：表示该方法或属性可以在本类和子类中访问。
- static：表示该方法或属性属于类，而不是属于类的实例。
- final：表示该方法或属性不能被重写或修改。
2. 构造方法的特点包括：
- 构造方法与类同名。
- 没有返回值类型。
- 可以有零个或多个参数。
- 当创建类的实例时，会自动调用构造方法。
3. 要想调用一个类下的方法和属性，需要创建该类的实例，并使用点号运算符来访问方法和属性。
4. 静态修饰的属性和方法的特点包括：
- 静态属性属于类，而不是类的实例，可以通过类名直接访问。
- 静态方法可以不创建类的实例而直接通过类名调用。
- 静态方法中不能访问非静态属性和方法。
- 静态语句块只会在类加载时执行一次。
5. 静态语句块的结构与特点：
- 静态语句块用 static 关键字修饰。
- 它在类加载时执行，且只会执行一次。
- 可以用于初始化类的静态属性。
6. 封装的特点包括：
- 隐藏类的内部实现细节。
- 提供公共的接口来访问类的属性和方法。
- 增强了代码的安全性和可维护性。
7. 继承的关键字是 extends。
8. 局部变量和成员变量的区别包括：
- 作用域不同：局部变量在方法内部定义，成员变量在类中定义。
- 生命周期不同：局部变量随着方法的执行而存在，方法执行结束后销毁；成员变量随着类的实例的创建而存在，实例销毁后销毁。
- 初始化方式不同：局部变量需要显式地初始化，成员变量可以有默认值或在构造方法中初始化。
9. 堆和栈的结构特点：
- 堆：用于存储对象实例，内存空间较大，分配和释放由垃圾回收器管理。
- 栈：用于存储方法内部的变量和引用，内存空间较小，自动分配和释放。
10. 面向对象的特点包括：
- 封装：将数据和对数据的操作封装在一起。
- 继承：实现代码的复用和扩展。
- 多态：允许不同的对象对同一消息做出不同的响应。public class Calculator {
```
    public static int add(int num1, int num2) {
        return num1 + num2;
    }

    public static int subtract(int num1, int num2) {
```

```
public static int subtract(int num1, int num2) {
    return num1 - num2;
}

public static int multiply(int num1, int num2) {
    return num1 * num2;
}

public static int divide(int num1, int num2) {
    if (num2 == 0) {
        throw new IllegalArgumentException("除数不能为零");
    }
    return num1 / num2;
}

public static void main(String[] args) {
    int num1 = 10;
    int num2 = 5 ;
    char operator = '+';
    if (args.length > 0) {
        operator = args[0].charAt(0);
    }

    switch (operator) {
        case '+':
            System.out.println(num1 + " + " + num2 + " = " + add(num1, num2));
            break;
        case '-':
            System.out.println(num1 + " - " + num2 + " = " + subtract(num1, num2));
            break;
        case '*':
            System.out.println(num1 + " * " + num2 + " = " + multiply(num1, num2));
```

参考答案：参考笔记和课堂练习