# 第二章 标识符、关键字、变量-答案

## 1. 程序阅读改错

* **题目要求**

1. 请阅读并分析以下四个源⽂件中的内容，找出其中会导致程序⽆法通过编译或运⾏的错误
2. 修改代码使它们能够编译和运⾏

* **Test1.java**

package com.briup.md01;

public class Test1 {

    public static void main(String[] args) {

        System.out.println("What's wrong with this program?");

    }

}

public class TestAnother1 {

    public static void main(String[] args) {

     System.out.println("What's wrong with this program?");

     }

}

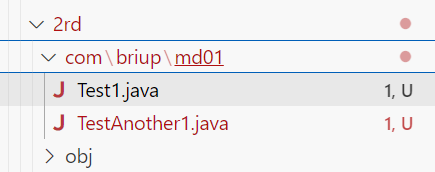
* **解：**
  + **错误1**

一个.java文件中只能有一个公共(public)类,源代码中有两个，故代码错误

* + **错误2**

且一个java程序可以没有main方法，但如果有只能有一个main方法

* + **修改后文件**

****

* + **修改后代码**

**文件Test1.java代码：**

package com.briup.md01;

public class Test1 {

    public static void main(String[] args) {

        System.out.println("What's wrong with this program?");

    }

}

**文件TestAnother1.java代码：**

package com.briup.md01;

public class TestAnother1 {

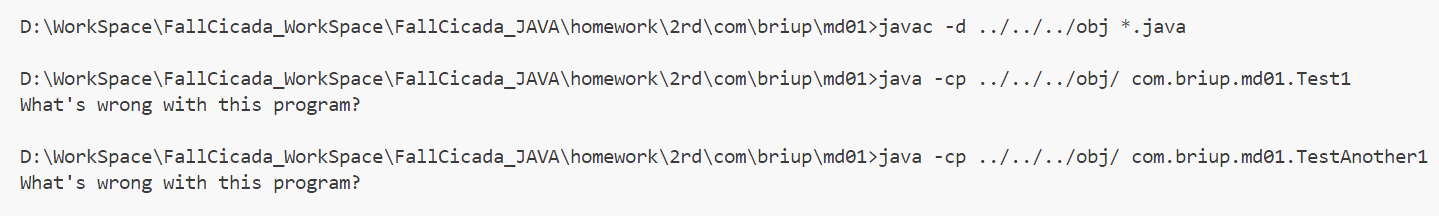
    public static void main(String[] args) {

     System.out.println("What's wrong with this program?");

     }

}

* + **运行指令**

****

* **Test2.java**

public class Test2 {

public static main(String[] args) {

 System.out.println("What's wrong with this program?");

 }

}

* **解：**
  + **错误**

Main方法中缺少关键字void

* + **修改后代码**

public class Test2 {

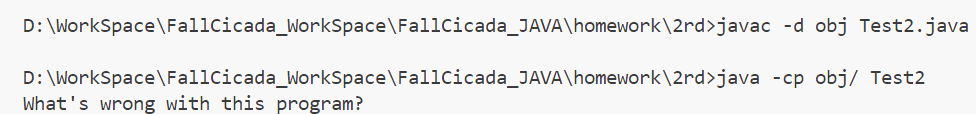
public static void main(String[] args) {

 System.out.println("What's wrong with this program?");

 }

}

* + **运行指令**

****

* **Test3.java**

public class Test3 {

public static void main(String args) {

 System.out.println("What's wrong with this program?");

 }

}

* **解：**
  + **错误**

**主函数定义中main()方法的中调用的参数应位String字符串所以应定义为String[ ] args 故原代码有错**

* + **修改后代码**

public class Test3 {

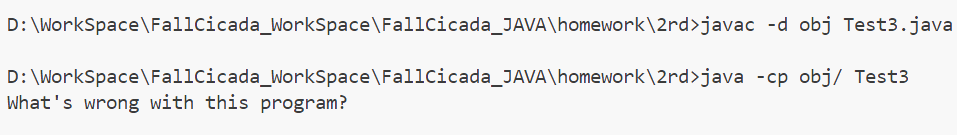
public static void main(String[] args) {

System.out.println("What's wrong with this program?");

 }

}

* **运行指令**



* **Test4.java**

public class Test4 {

public void main(String[] args) {

 System.out.println("What's wrong with this program?");

 }

}

* **解：**
  + **错误：**

Main方法命名应该是公共、静态、空类型函数故原代码中缺失static关键字，故原程序代码错误

* + **修改后代码**

public class Test4 {

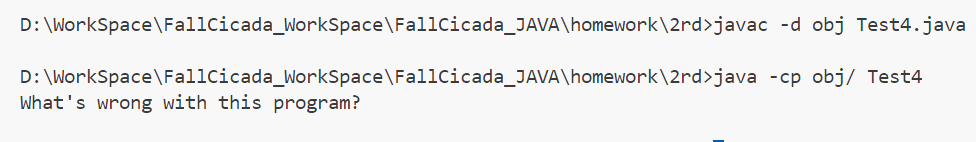
public static void main(String[] args) {

System.out.println("What's wrong with this program?");

  }

}

* **运行指令：**



* **总结**

**Java 的 main 方法之所以这样定义 (public static void main(String[] args))，是因为 Java 虚拟机 (JVM) 在启动时需要一个入口点来开始执行程序。以下是这个方法定义的各个部分的作用：**

1. **public: 这个关键字表示 main 方法是公开的，可以从程序的外部调用。这意味着 JVM 可以找到并调用这个方法，而不需要导入任何类或包。**
2. **static: main 方法必须是静态的，因为它是在类的实例化之前就被调用的。也就是说，在程序启动时，还没有创建类的任何实例，因此 main 方法不能依赖于类的实例。静态方法可以在不创建类实例的情况下直接调用。**
3. **void: 这个关键字表示 main 方法不会返回任何值。也就是说，main 方法的主要任务是执行程序，并且它不会向调用者返回任何结果。**
4. **main: 这是方法的名字。JVM 在启动时会寻找一个名为 main 的方法作为程序的入口点。**
5. **String[] args: 这个参数表示 main 方法接受一个字符串数组作为参数。这个数组允许程序从命令行接收参数。在命令行中传递给程序的每一个参数都会成为数组中的一个元素。**

* **公共性 (public): 允许 JVM 调用这个方法。**
* **静态性 (static): 不需要创建类的实例即可调用。**
* **无返回值 (void): 方法不需要返回任何值。**
* **方法名 (main): 必须叫做 main，因为这是 JVM 启动程序时查找的方法名。**
* **参数 (String[] args): 提供了一种方式让程序接收命令行参数。**

## 2.变量声明

* **题⽬要求： 请观察下列代码中的变量声明语句，指出哪⼏句会产⽣编译报错或精度丢失，并解释原因。**

{

            byte a = 56;

            byte b = 156;

            int c = 2000000000 + 2000000000;

            long d = 999999999;

            long e = 9999999999;

            long f = 9999999999L;

 }

* **解**
* **byte a = 56;**
  + **结果：** 正确
  + **原因：** 56 是一个有效的 byte 值范围内的数字（-128 到 127）。
* **byte b = 156;**
  + **结果：** 编译错误
  + **原因：** 156 超出了 byte 类型的有效范围（-128 到 127）。编译器会在编译时检测到这一点，并抛出错误。
  + **修改：**定义为long b = 156L;
* **int c = 2000000000 + 2000000000;**
  + **结果：** 正确
  + **原因：** 这个表达式计算的是两个 int 类型的值相加。虽然结果可能会超出 int 类型的正常范围（-2^31 到 2^31 - 1），但由于 int 类型的溢出行为是循环的（即当数值超过最大值时会从最小值开始计数），所以这个表达式本身不会导致编译错误。然而，如果预期结果需要保持准确，那么这种溢出可能导致精度丢失。
  + **修改：**定义为long c = 2000000000L + 2000000000L;
* **long d = 999999999;**
  + **结果：** 正确
  + **原因：** 999999999 是一个有效的 long 类型的值。默认情况下，整数常量被视为 int 类型，但是 999999999 在 int 和 long 的范围内，因此这个赋值是安全的。
  + **修改：**long d = 999999999L;
* **long e = 9999999999;**
  + **结果：** 编译错误
  + **原因：** 9999999999 默认被视为 int 类型的值，但这个值超出了 int 类型的最大值（2^31 - 1），并且没有明确指定为 long 类型（缺少 L 或 l 后缀）。因此，编译器会抛出错误。
  + **修改：**long e = 9999999999L;
* **long f = 9999999999L;**
  + **结果：** 正确
  + **原因：** 9999999999L 明确指定了 long 类型，因此这是一个有效的 long 值。后缀 L 表示这是一个 long 类型的常量。

## 3.进制格式考察

* **题⽬要求：定义3个int类型变量，分别使⽤⼆进制、⼋进制、⼗六进制三种字⾯常量语法为其赋值，使其值均等于⼗进制值 85。**
* **解**

int binaryValueEighty\_five = 0b1010101;

int octalValueEighty\_five = 0125;

int hexValueEighty\_five = 0x55;

## 4.找零问题

* **题⽬要求：⼩明有2元，买东⻄花了1.1元，问找零多少，请⽤Java代码描述上述过程！ 注意：观察程序运算输出是否能得到0.9，如果不能，请解决该精度问题。**
* **解**

import java.math.BigDecimal;

import java.math.RoundingMode;

public class Math {

    public static void main(String[] args) {

        BigDecimal money = new BigDecimal("2.0");

        BigDecimal cost = new BigDecimal("1.1");

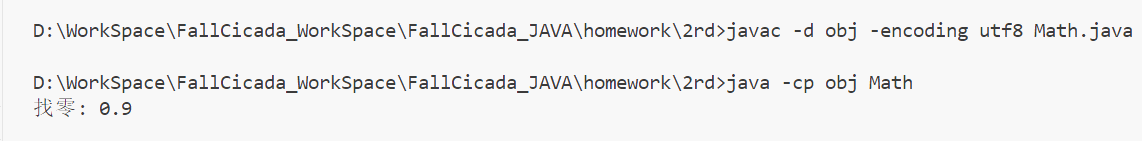
        BigDecimal change = money.subtract(cost).setScale(1, RoundingMode.HALF\_UP);

        System.out.println("找零: " + change);

    }

}

**指令：**



## 5．字⾯值常量考查

* **题⽬要求： 请定义2个变量，分别⽤每天对应的毫秒数以及微秒数（表达式描述）给其赋值。输出变量值，观察是否正确。如 果输出有问题，请解决！**
* **解：**

public class Time {

    public static void main(String[] args) {

        long millisecondsInADay = 24 \* 60 \* 60 \* 1000L;

        long microsecondsInADay = 24 \* 60 \* 60 \* 1000L \* 1000L;

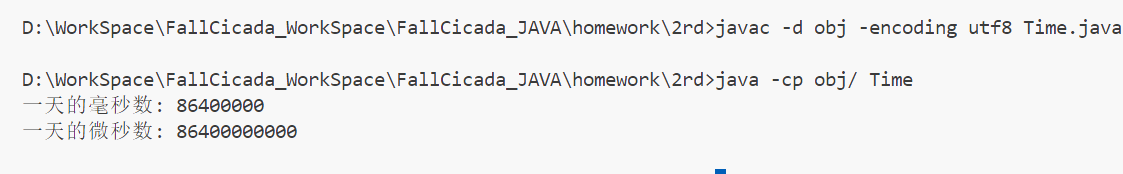
        System.out.println("一天的毫秒数: " + millisecondsInADay);

        System.out.println("一天的微秒数: " + microsecondsInADay);

    }

}

**运行指令**



## 6. 数据类型问答

* **题⽬要求： 基本类型有哪些，取值范围分别是什么？**



**| 数据类型 | 中文名称 | 所占内存大小 |**

**|----------|----------|--------------|**

**| `byte`   | 字节     | 1字节        |**

**| `short`  | 短整型   | 2字节        |**

**| `int`    | 整型     | 4字节        |**

**| `long`   | 长整型   | 8字节        |**

**| `float`  | 浮点型   | 4字节        |**

**| `double` | 双精度浮点型 | 8字节        |**

**| `char`   | 字符     | 2字节        |**

**| `boolean`| 布尔     | 通常认为是1字节（实际实现可能不同） |**

## 7.数据类型问答

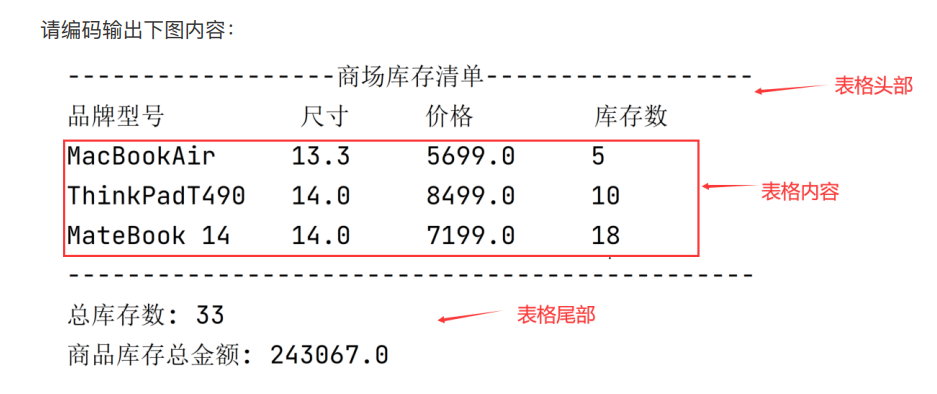
* **题⽬要求： 请回答引⽤数据类型有哪些，分别是什么？**
  + **类 (class)**: 这是最常见的引用数据类型，用于定义对象。类可以用来创建各种复杂的对象，如用户定义的对象、数组等。
  + **接口 (interface)**: 接口是一种特殊的类，它定义了一组方法签名，而不包含方法的实现。接口用于实现多态性和抽象编程。
  + **数组 (array)**: 数组是一种特殊类型的引用数据类型，用于存储固定数量的同类型元素。数组可以是一维的、多维的或不规则的。
  + **枚举 (enum)**: 枚举类型用于定义一组固定的常量值。每个枚举类型都定义了一个新的类，该类继承自 java.lang.Enum。
  + **泛型 (generic)**: 泛型是一种在类或接口中使用类型参数的方式，可以用来创建具有通用功能的类或接口。泛型提供了一种编写可重用代码的方式。
  + **注解 (annotation)**: 注解是一种特殊的引用类型，用于为代码添加元数据。注解本身也是类，通常用于编译时或运行时的元信息处理。

## 8. 数据类型问答

* **题⽬要求： 请简述基本类型和引⽤类型变量的区别。**
* **解：**
  + **基本类型（Primitive Types）** 
    - * + **存储位置：基本类型的数据直接存储在栈（Stack）中。**
        + **内存分配：为基本类型的变量分配固定的内存空间。**
        + **赋值操作：当把一个基本类型的值赋给另一个变量时，实际上是复制了一个值。这意味着两个变量各自拥有独立的内存空间，改变其中一个不会影响到另一个。**
        + **生命周期：基本类型的变量在其声明的作用域内存在，一旦超出作用域就会被销毁。**
  + **引用类型（Reference Types）**
    - * + **存储位置：引用类型的变量实际上存储的是指向实际数据的引用（地址），而实际数据则存储在堆（Heap）中。**
        + **内存分配：为引用类型的变量分配的内存较小，只存放指向数据的指针；而实际数据的内存大小取决于数据本身的大小。**
        + **赋值操作：当把一个引用类型的值赋给另一个变量时，实际上是复制了一个引用。这意味着两个变量指向同一个内存地址，因此改变其中一个会影响到另一个。**
        + **生命周期：引用类型的变量可能需要通过垃圾回收机制来释放内存，而非自动销毁。**

## 9.编程题

* **题⽬要求： 请编码输出下图内容：**

****

* **案例分析：**

**输出内容包含3部分，表头、表格、表尾，具体要求如下：**

* + **表头部分：可以采⽤多条System.out.println语句输出，其是固定数据， 直接输出即可**
  + **表格中间：请定义变量描述商品型号、尺⼨、价格、库存，然后按格式进⾏输出（注意变量数据类型 ）**
  + **表格尾巴：先对商品数据进⾏数学计算，得出结果后输出内容**

**解：**

public class Table {

    public static void main(String[] args) {

        double a1=5;

        double a2=10;

        double a3=18;

        double b1=5699.0;

        double b2=8499.0;

        double b3=7199.0;

        double A=a1+a2+a3;

        double B=b1\*a1+b2\*a2+b3\*a3;

        float size1 = 13.3f;

        float size2 = 14.0f;

        float size3 = 14.0f;

        System.out.println("----------------商场库存名单----------------");

        System.out.println("品牌型号"+"\t"+"尺寸"+"\t"+"价格"+"\t"+"库存数"+"\n" + "MacBOOKair"+ "\t"+ size1 +"\t"+ b1 +"\t" + a1 + "\n" + "ThinkPadT490"+ "\t"+size2+"\t"+ b2+"\t" + a2 + "\n"+"MateBook 14"+ "\t"+ size3 + "\t"+ b3 +"\t "+a3);

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

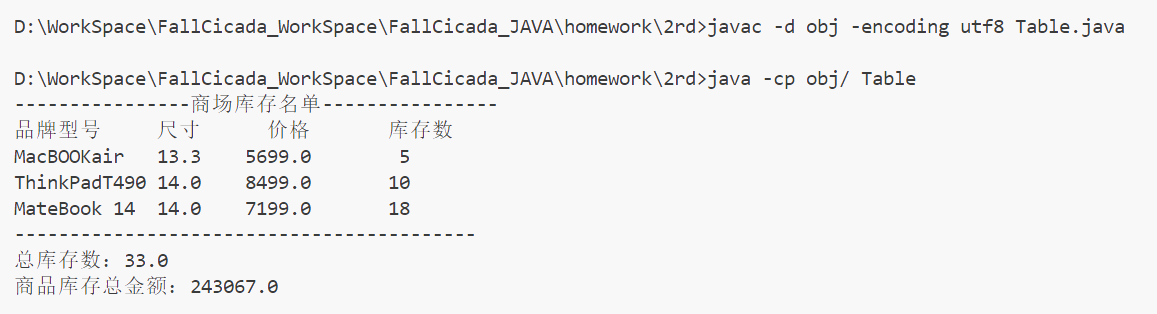
        System.out.println("总库存数："+A);

        System.out.println("商品库存总金额："+B);

    }

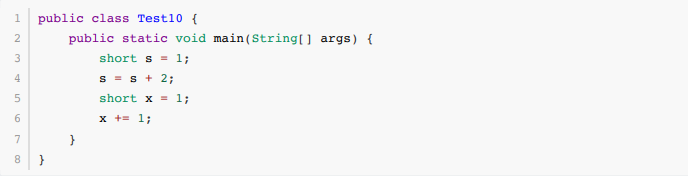
}

* **运行指令**



## 10. 程序分析题

**题⽬要求： 分析以下代码的运⾏结果。**

****

**public class Test10 {**

**public static void main(String[] args) {**

**short s = 1; // 定义了一个short类型的变量s，并初始化为1**

**s = s + 2; // 将s的值与2相加，并将结果赋值给s**

**short x = 1; // 定义了一个short类型的变量x，并初始化为1**

**x += 1; // 将x的值增加1**

**}**

**}**

* **判断：**
  + **short s = 1;**: 定义了一个 short 类型的变量 s 并将其初始化为 1。
  + **s = s + 2;**: 这一行将 s 的值与 2 相加，并将结果重新赋值给 s。由于 s 是 short 类型，而 2 是 int 类型，默认情况下 s + 2 的结果将被提升为 int 类型。然而，Java 允许将 int 类型的值自动转换回 short 类型，只要结果不超过 short 类型的范围（-32,768 至 32,767）。因此，这一行不会导致编译错误。最终 s 的值变为 3。