第二章 标示符、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?");
    }
}
```

答案及解析：

- **错误原因**

  一个Java源文件中最多只能出现一个public修饰的类，而案例中Test1和TestAnother1两个类都是由public修饰的。

- **解决方案**

  将其中至少一个public去掉，或将两个类编写到两个不同的源代码文件中即可。

正确代码示范如下：

```
package com.briup.md01;
public class Test1 {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
class TestAnother1 {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

或者：

Test1.java

```
package com.briup.md01;

public class Test1 {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

TestAnother1.java

```
package com.briup.md01;

public class TestAnother1 {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

Test2.java

```
public class Test2 {
    public static main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

答案及解析：

- **错误原因**

  主方法的声明不正确，缺少了返回值类型void。

  在Java中，所有方法都必须声明返回值类型。即使方法无返回值，也必须声明为void。主方法返回值必须为void类型。

- **解决方案**

  在主方法声明中添加void关键字即可。

正确代码示范如下：

Test2.java

```
public class Test2 {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

Test3.java

```
public class Test3 {
    public static void main(String args) {
        System.out.println("What's wrong with this program?");
    }
}
```

答案及解析：

- **错误原因**

  主方法参数类型定义错误。

  > 在 Java 中，main 方法是程序的入口方法，是 Java 虚拟机在执行 Java 程序时第一个被执行的方法。main 方法的定义必须遵守特定的语法规则，其中一个重要的规则是 main 方法的参数必须是一个字符串数组类型 String[]。这是因为在命令行中运行 Java 程序时，用户可以在命令行中传入一些参数。这些参数将被作为一个字符串数组传递给 main 方法。因此，为了能够正确地接收和处理这些参数，main 方法的参数必须是一个字符串数组类型 String[]。如果省略了这个参数，就无法接收和处理命令行传入的参数，会导致程序出错。

- **解决方案**

  将主方法参数类型改为String[]。

正确代码示范如下：

Test3.java

```
public class Test3 {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

Test4.java

```
public class Test4 {
    public void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

答案及解析：

- **错误原因**

  主方法未使用static修饰。

  Java中要求，主方法必须是**public static void**修饰的。

> 在 Java 中，静态方法是属于类的方法，而不是属于对象的方法。静态方法不依赖于任何对象实例，可以直接通过类名调用，而不需要先创建对象。因此，main 方法必须是一个静态方法，因为它是 Java 程序的入口方法，必须在任何对象实例化之前被执行。具体来说，当我们执行 Java 程序时，JVM 首先加载并执行 main 方法，这个方法是程序的入口点，是程序开始执行的地方。因为在 main 方法被执行时还没有任何对象实例化，所以必须将 main 方法声明为静态方法，才能直接通过类名来调用它，从而启动程序的执行过程。如果将 main 方法声明为非静态方法，则必须先创建 Test4 对象实例，然后通过对象调用 main 方法。但是，这样做是没有意义的，因为在执行 main 方法时并不需要对象实例，而且在没有创建对象之前也没有办法调用对象方法。因此，**main 方法必须是一个静态方法。**

- **解决方案**

  在主方法声明处添加**static**关键字。

正确代码示范如下：

```
public class Test4 {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

或者：

```
public class Test4 {
    static public void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

2.变量声明改错

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;
}
```

答案及解析：

在上述代码中，有四个变量的声明存在问题：

```
{
    byte a = 56;
    byte b = 156; // 错误1
    int c = 2000000000 + 2000000000; // 错误2
    long d = 999999999;
    long e = 9999999999; // 错误3
    long f = 9999999999L;
}
```

错误1：

编译报错。

使用字面常量为byte类型变量赋值时，编译器会检查其值是否在byte类型可表示范围内（-128 ~ 127），如超出范围则会报错。

修正的方式是将变量类型改为更大的类型即可，例如short、int、long等。

或使用强制类型转换语法：

```
byte b = (byte) 156; // 注意，会有精度丢失
```

错误2：

编译通过，但精度丢失。

变量 c 被声明为 int 类型，可表示范围为`-2 ^ 31 ~ 2 ^ 31 -1`（大致为正负21亿），而等号右边式子的计算结果为40亿，超出int可表示范围。程序运行之后，变量c中存储的值会丢失精度。

修正的方式是使用足够大的变量类型，例如long类型：

```
long c = 2000000000 + 2000000000;
```

错误3：

编译报错。

Java中整型字面常量的默认类型是int，而int无法表示9999999999这个数值。

修正方式是在9999999999后边添加`l`或`L`，将其声明为一个long类型字面常量。

```
long e = 99999999991;
```

或：

```
long e = 9999999999L;
```

> 在编程规范中，不建议使用小写字母`l`来标识long类型字面常量，因其容易与阿拉伯数字`1`混淆，影响代码可读性。尽量使用大写字母`L`。

3.进制格式考查

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。

答案及解析：

```
int binVal = 0b01010101; // 二进制，0b或0B开头均可
int octVal = 0125; // 八进制，数值前加0
int hexVal = 0x55; // 十六进制，0x或0X开头均可
```

注意，无论数字是以哪种进制表示，Java中的整数变量默认为十进制。如果需要将不同进制的数字相互转换，可以使用Java中的进制转换函数，例如：

- 将二进制字符串转换为十进制整数：Integer.parseInt("1010101", 2)；
- 将八进制字符串转换为十进制整数：Integer.parseInt("125", 8)；
- 将十六进制字符串转换为十进制整数：Integer.parseInt("55", 16)；
- 将十进制整数转换为二进制字符串：Integer.toBinaryString(85)；
- 将十进制整数转换为八进制字符串：Integer.toOctalString(85)；
- 将十进制整数转换为十六进制字符串：Integer.toHexString(85)。

4.找零问题

题目要求：

小明有2元，买东西花了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。

答案及解析：

大部分同学可能会书写如下代码：

```
//定义变量，赋初值为2元
double money = 2.0;

//定义商品价格 1.1元
double price = 1.1;

//运算得出 找零结果
double change = money - price;

// 输出找零结果
System.out.println("找零：" + change);
```

发现结果是：0.8999999999999999

因为double类型存储的是浮点数，存在精度问题，现在给出两种解决方案：

- 使用DecimalFormat对结果进行格式化，保留一位小数

```
double money = 2.0; // 给定的金额
double price = 1.1; // 购买商品的价格
double change = money - price; // 计算找零

/*
 * 因为double类型存储的是浮点数，存在精度问题，
 * 所以需要使用DecimalFormat对结果进行格式化，保留一位小数
 */
DecimalFormat df = new DecimalFormat("#0.0");
System.out.println("找零：" + df.format(change)); // 输出找零结果0.9
```

- **借助BigDecimal类型可以解决上述精度问题，Java中的`BigDecimal`类可以解决上述精度问题。`BigDecimal`可以表示任意精度的十进制数，因此可以用它来处理需要高精度计算的场景，例如金融计算。(推荐)**

```
BigDecimal money = new BigDecimal("2.0"); // 给定的金额
BigDecimal price = new BigDecimal("1.1"); // 购买商品的价格
double change = money.subtract(price).doubleValue();; // 计算找零

System.out.println("找零：" + change); // 输出找零结果0.9
```

- 易错情况：

```
BigDecimal money = new BigDecimal(2.0);
System.out.println(money.toString()); //2
BigDecimal price = new BigDecimal(1.1);
System.out.println(price.toString());//1.100000000000000088817841970012523233890533447265625
```

- **结论:在创建 BigDecimal 对象的时候推荐使用字符串类型的参数代码实现**

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！

答案及解析：

```
public class Test05_Long {
    public static void main(String[] args) {

        // 毫秒数：86400000
        long mssec = 24 * 60 * 60 * 1000;
        System.out.println("毫秒数：" + mssec);
```

```
7
8           // 毫秒数：500654080  错误写法
9           long ussec = 24 * 60 * 60 * 1000 * 1000;
10
11          // 正确写法
12          //long ussec = 24 * 60 * 60 * 1000 * 1000L;
13          System.out.println("微秒数：" + ussec);
14      }
15  }
```

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？

答案及解析：

数据类型	长度	取值范围
byte	8位	-2 ^ 7 ~ 2 ^ 7 - 1
short	16位	-2 ^ 15 ~ 2 ^ 15 - 1
int	32位	-2 ^ 31 ~ 2 ^ 31 - 1
long	64位	-2 ^ 63 ~ 2 ^ 63 - 1
float	32位	1.4E-45 ~ 3.4028235E38
double	64位	4.9E-324 ~ 1.7976931348623157E308
boolean	1位	true 或 false
char	16位	'\u0000' 到 '\uffff'

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？

答案及解析：

Java语言中引用类型主要包括以下几种：

1. 类类型（Class Type）：定义了一类对象的属性和方法，是Java语言中最常用的引用类型。例如，可以使用类类型来表示人、汽车、电视等各种具体对象。
2. 接口类型（Interface Type）：定义了一组方法的集合，不同于类类型，接口类型没有实例变量，只有方法和

常量的定义。接口类型用于定义规范和约定，便于不同的类实现相同的功能接口。
3. 数组类型（Array Type）：由相同类型的变量组成的有序集合，可以存储多个同类型的元素。数组类型在Java中非常常用，可以用来表示列表、矩阵等数据结构。
4. 枚举类型（Enumeration Type）：表示一组有限的值，这些值是事先定义好的。枚举类型可以用来表示一些固定的状态、类型等。
5. 注解类型（Annotation Type）：是Java语言中的一种元数据，用来描述程序中的代码元素（如类、方法、变量等）的属性和规范。注解类型可以被用来实现一些特定的编译时处理任务。

目前我们先知道前三种引用类型即可，这三种也是日常开发中比较常用的。

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。

答案及解析：

当我们在Java程序中声明变量时，我们需要告诉编译器这个变量的类型，根据类型的不同，变量可以分为基本数据类型和引用类型。

1.基本数据类型变量

Java的基本数据类型是内置的，它们具有固定的值范围和默认值，因此在声明基本数据类型变量时，我们只需要指定它们的类型就可以了。例如：

```
int number = 10;
double pi = 3.1415926;
boolean flag = true;
char ch = 'A';
```

上述代码中，我们声明了四个基本数据类型变量，它们分别是整型变量number，双精度浮点型变量pi，布尔型变量flag和字符型变量ch。这些变量都是直接存储在栈内存中，变量名表示该变量存储的值。

基本数据类型变量的特点：

- 存储方式：基本数据类型变量的值直接存储在栈空间中，这些值的大小是固定的。
- 复制方式：基本数据类型的值在赋值给另一个变量时，会将该值复制一份，分配到另一个变量的栈空间中。（数值传递）
- 默认值：在Java语言中，基本数据类型有默认值，如果没有给变量赋初值，系统会自动给它赋上一个默认值。
- 内存回收：基本数据类型的变量在超出作用域后，会自动被系统回收。

2.引用类型变量

与基本数据类型不同，引用类型的变量存储的是指向堆内存中对象的地址，而对象的实际数据存储在堆中。在声明引用类型变量时，我们需要使用new关键字来创建对象，并使用变量名来引用对象。例如：

```
String str = new String("hello");
```

上述代码中，我们声明了一个String类型的引用类型变量str，用new关键字创建了一个字符串对象并将该对象的地址赋给str变量。

引用类型变量的特点：

- 存储方式：引用类型变量存储的是该对象在堆空间中的地址。
- 复制方式：在将一个引用类型的变量赋值给另一个变量时，只会复制该引用的地址，指向同一个对象，它们共享同一个对象实例。（地址传递）
- 默认值：在Java语言中，引用类型变量默认值为null。
- 内存回收：引用类型的变量指向的对象，在没有任何引用指向它时，就变成了垃圾对象，Java的垃圾收集器会自动回收这些对象占用的内存。

3.总结：

在Java语言中，基本数据类型和引用类型是两种不同的变量类型。它们之间的主要区别在于：

- 存储方式

基本数据类型的变量直接存储在栈（Stack）中，而引用类型的变量存储在堆（Heap）中。基本数据类型的值直接存储在栈空间中，这些值的大小是固定的，而引用类型的变量存储在堆空间中，存储的是该对象在堆空间中的地址。

- 复制方式

在将一个基本数据类型的变量赋值给另一个变量时，会将该值复制一份，分配到另一个变量的栈空间中（数值传递）。而在将一个引用类型的变量赋值给另一个变量时，只会复制该引用的地址，指向同一个对象，它们共享同一个对象实例（地址传递）。

- 默认值

在Java语言中，基本数据类型有默认值，而引用类型没有。当定义一个基本数据类型的变量时，如果没有给它赋初值，系统会自动给它赋上一个默认值。比如，int类型的默认值为0，boolean类型的默认值为false。而引用类型变量默认值为null。

- 内存回收

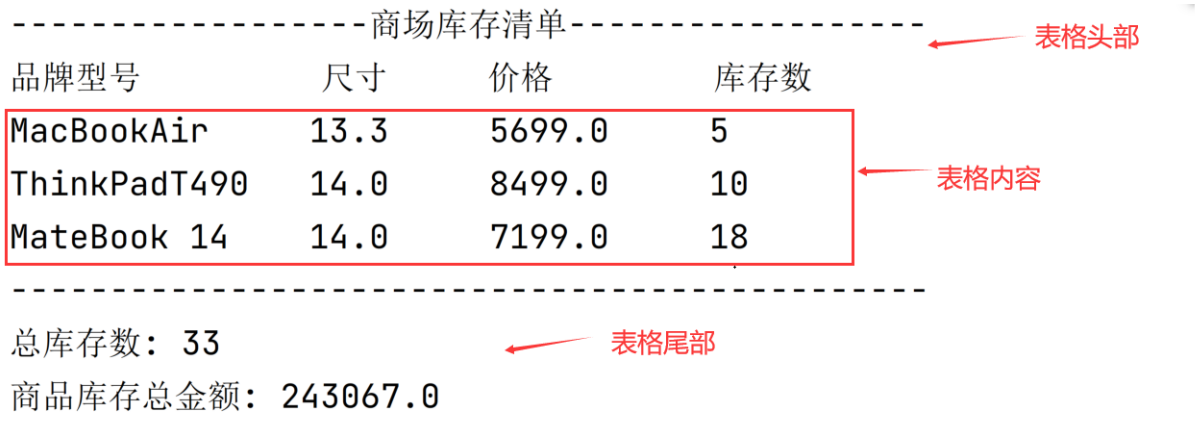
在Java语言中，基本数据类型的变量在超出作用域后，会自动被系统回收。而引用类型的变量指向的对象，在没有任何引用指向它时，就变成了垃圾对象，Java的垃圾收集器会自动回收这些对象占用的内存。

综上所述，基本数据类型和引用类型在存储方式、复制方式、默认值和内存回收方面存在明显的区别。需要注意的是，Java语言中的基本数据类型包括byte、short、int、long、float、double、char和boolean，而其他的所有类型都是引用类型。

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

答案及解析：

```
public class Test09_Table {
    public static void main(String[] args) {
        //输出表头固定数据
        System.out.println("------------------商场库存清单------------------");
        System.out.println("品牌型号          尺寸      价格          库存数");

        //定义表格中的数据变量
        //品牌型号,String, 尺寸,价格 double 库存int
        String macBrand = "MacBookAir";
        double macSize = 13.3;
        double macPrice = 5699.00;
        int macCount = 5;

        String thinkBrand = "ThinkPadT490";
        double thinkSize = 14;
        double thinkPrice = 8499.00;
        int thinkCount = 10;

        String mateBrand = "MateBook 14";
        double mateSize = 14;
        double matePrice = 7199.00;
        int mateCount = 18;

        //商品信息变量进行打印,变量之间加入一定的字符串空格
        System.out.println(macBrand+"      "+macSize+"      "+macPrice+"
"+macCount);
        System.out.println(thinkBrand+"    "+thinkSize+"      "+thinkPrice+"
"+thinkCount);
```

```
        System.out.println(mateBrand+"    "+mateSize+"     "+matePrice+"
"+mateCount);

        //计算库存总数,所有商品数量库存求和
        int totalCount = macCount+thinkCount+mateCount;

        //计算所有商品库存的总金额,每个商品价格*库存数
        double totalMoney = macCount*macPrice + thinkCount*thinkPrice +
mateCount*matePrice;

        //输出表格底部
        System.out.println("-----------------------------------------------");
        System.out.println("总库存数: "+totalCount);
        System.out.println("商品库存总金额: "+totalMoney);
    }
}
```

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;
        s = s + 2;
        short x = 1;
        x += 1;
    }
}
```

答案及解析:

事实上，将值3赋给`short`类型的变量`s`是完全合法的，因为它可以表示为一个16位的二进制补码，而该值在`short`类型的取值范围内。所以对于下面的代码:

```
short s = 3;
```

不会有编译或运行时错误。但是，在上面的代码示例中，错误是由`s = s + 2;`这行代码引起的，因为`+`运算符返回一个`int`类型的结果，而将一个`int`类型的值赋给一个`short`类型的变量需要进行显式的类型转换，否则会导致编译时错误。

因此，将表达式`s + 2`的结果转换为`short`类型的最简单方法是将它括在圆括号内，并使用类型转换运算符将其转换为`short`类型。

对于`s = s + 2`这行代码，它在编译时会报错，因为`+`运算符将导致表达式的结果被自动提升为`int`类型，而将`int`类型的值直接赋给`short`类型的变量是不安全的，因为可能会导致数据丢失。为了将`int`类型的值转换为`short`类型，需要使用强制类型转换运算符`()`，将表达式`s+2`的结果转换为`short`类型。因此，正确的代码应该是：

```
short s = 1;
s = (short) (s + 2);
```

这样就可以将`int`类型的结果强制转换为`short`类型，以便将其赋值给`s`变量，避免编译和运行时错误。

对于第6行的表达式`x += 1;`，它等价于`x = (short) (x + 1);`，不会抛出错误，因为复合赋值运算符`+=`会自动将表达式的结果转换为被赋值变量的类型。