河南中医药大学智能医学工程专业《计算机程序设计》课程

第02章：了解Java程序

黄子杰

河南中医药大学信息技术学院（智能医疗行业学院）与河南方和信息科技有限公司 联合建设

河南中医药大学信息技术学院互联网技术教学团队

https://webdev.hactcm.edu.cn

2024/9/4

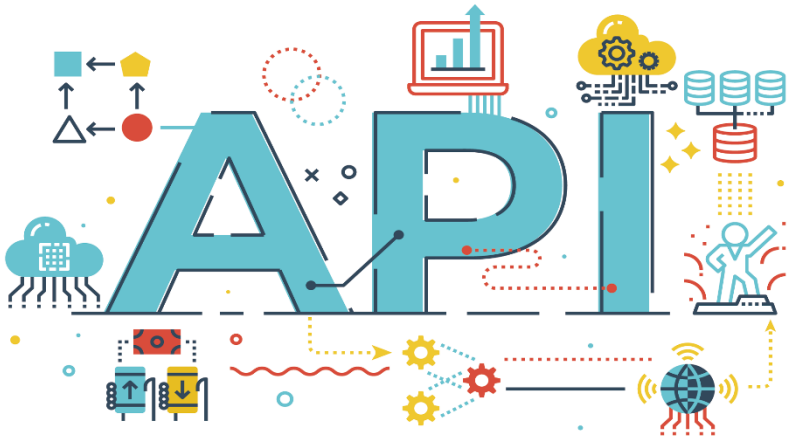
本章概要

了解程序和计算机程序

```
START:   MOV AX,5000H
         MOV DS,AX
         MOV SI,0
         MOV CX,0FFFFH
LOOP1:   MOV BYTE PTR[SI],55H
         MOV AL,[SI]
         CMP AL,55H
         JNZ LOOPERR
LOOP2:   INC SI
         LOOP LOOP1
         MOV BYTE PTR[SI],55H     ;最后一个单元
         MOV AL,[SI]
         CMP AL,55H
         JNZ LOOPERR
         MOV AL,0                 ;全对
       JMP LOOPOUT
LOOPERR:  MOV AL, 0FFH
LOOPOUT: NOP
```

将内存从50000H到5FEEFH的每个单元均写入数55H，
并再逐个单元读出比较，看写入的与读出的是否一致。
若全对，则将AL置0：只要有错，则将AL置0F

汇编语言是最接近机器语言的编程语言

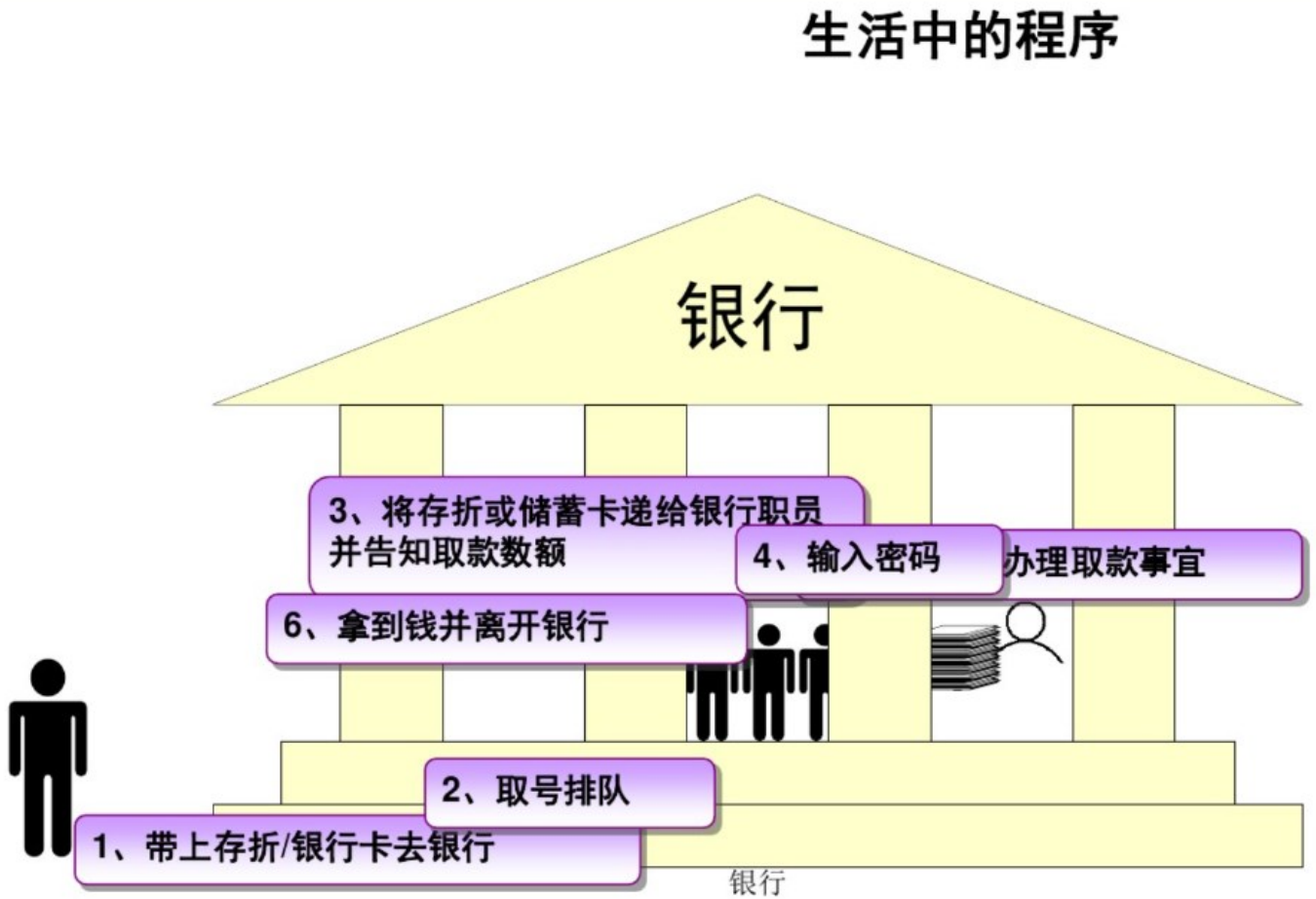


2.1 什么是程序

- 程序一词来源于生活,通常指完成某些事务的一种既定方式和过程
- 在日常生活中可以将程序看成对一系列动作的执行过程的描述

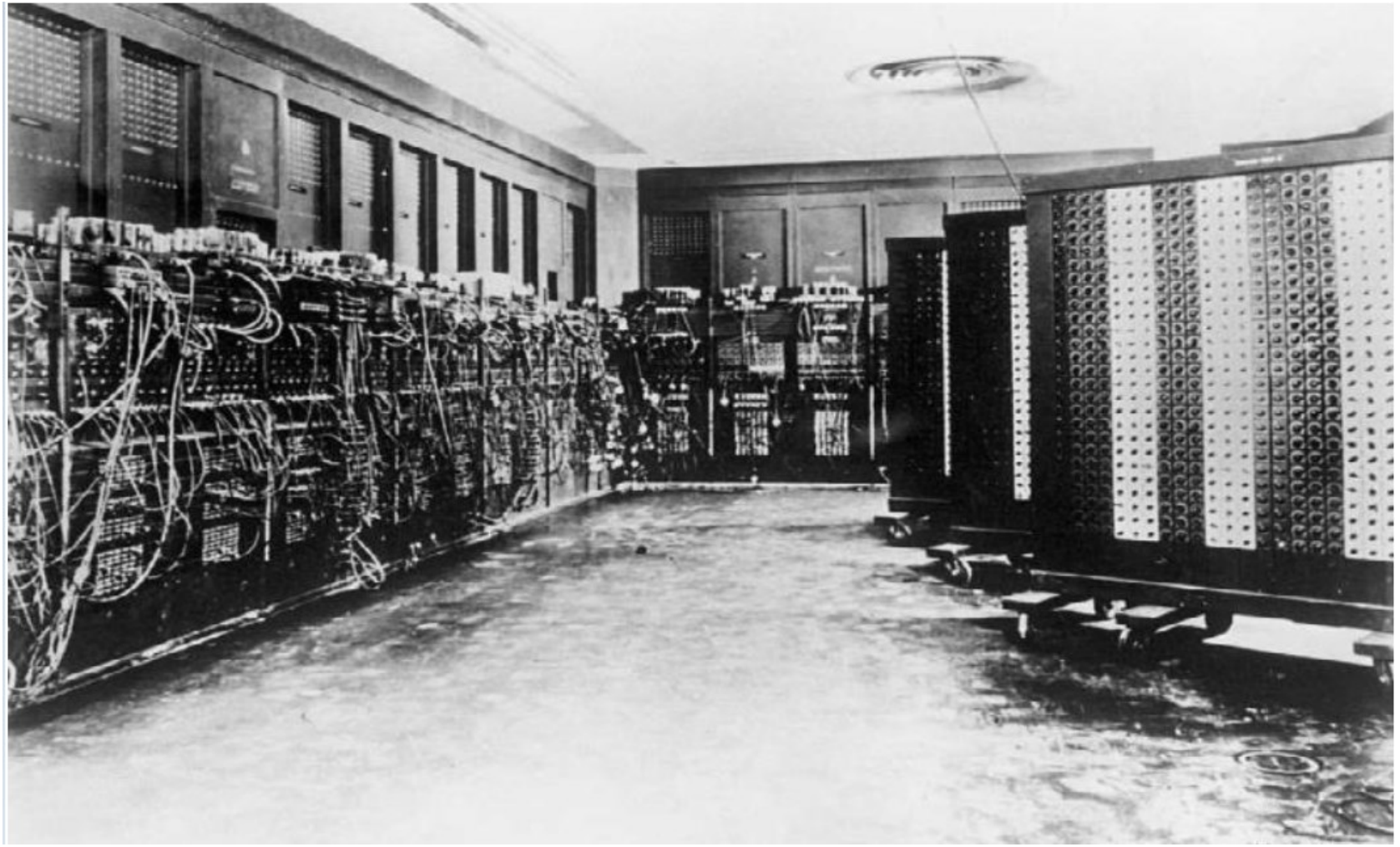
2.1 什么是程序

- 一张图来理解什么是程序



2.1 什么是程序

- 1946年第一台计算机ENIAC



2.1 什么是程序

- 第一台计算机的程序

世界上第一台计算机ENIAC是没有操作系统的，它是靠改变插线编程的，没有屏幕，输入输出全靠穿孔卡。

当时的穿孔卡只能用来输入数据还不能用来编程。

(穿孔卡大概长这样，每一列是一个字符)

从第二台计算机开始，可以在内存里存储程序了，

穿孔卡也就用来输入程序了。

最早的程序应该是用机器语言写的，那时程序以卡片做为承载媒介，在卡片上打孔打出来，再把这样一摞卡片交给机房，你就可以等结果了。结果算出来了，就再打一摞卡给你。当然，你要是写错了，纸上可能只有一句错误代码，那之前的等待就白等了，卡片也就浪费了。

2.2 计算机中的程序

- **程序的组成**

程序由数据和指令组成。

- **什么是机器语言**

机器语言是直接用二进制代码指令表达的计算机语言，指令是用0和1组成的一串代码，它们有一定的位数，并分成若干段，各段的编码表示不同的含义。

- **运行中的程序存储在什么位置**

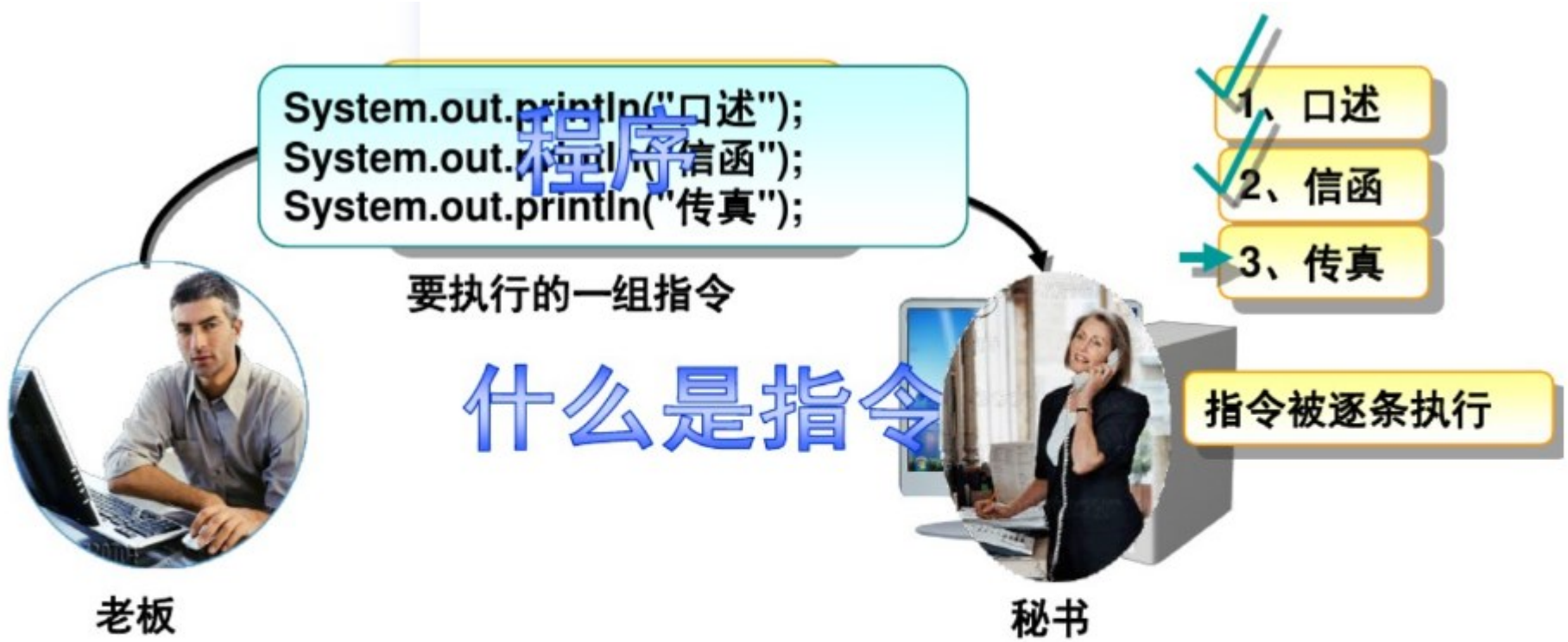
程序加载时首先到寄存器中，寄存器会将程序复制到内存中从而进行存储，当程序运行时，CPU会把主从的程序的数据和指令调用到寄存器特定的位置，从而执行。

- **内存地址**

内存地址指系统 RAM 中的特定位置，通常以十六进制的数字表示，如同计算机内部特定位置的编号。

2.2 计算机中的程序

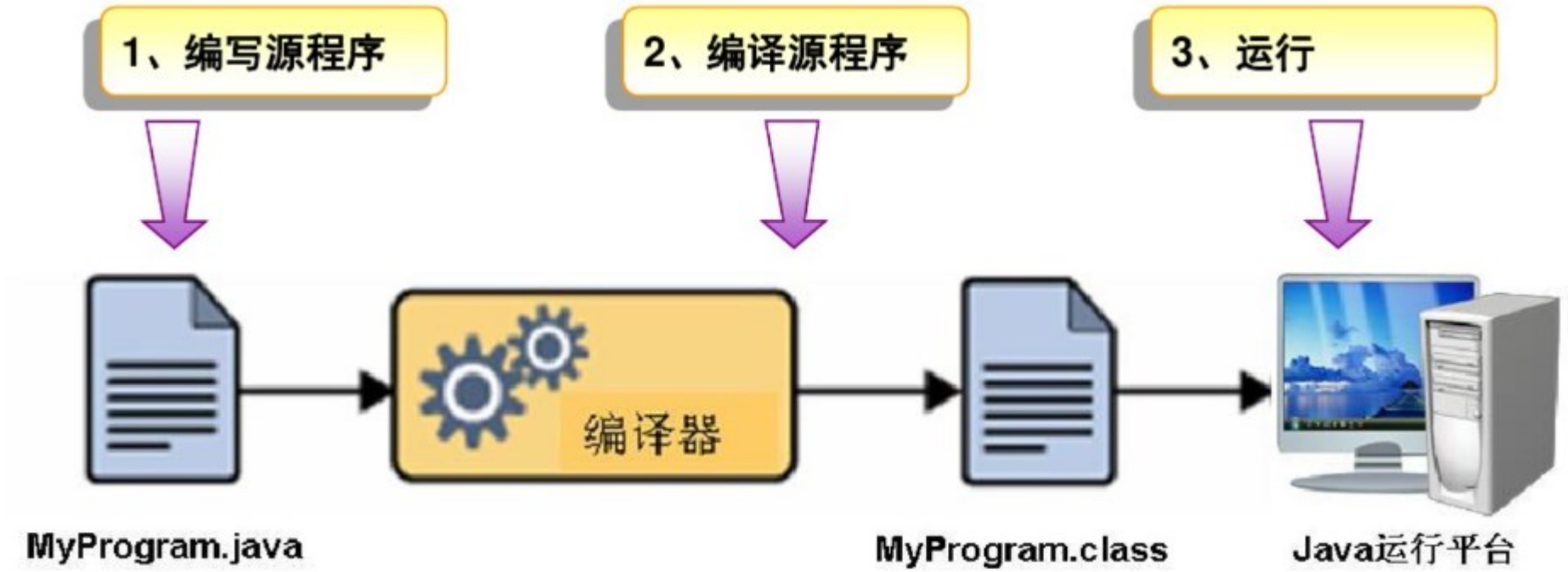
生活中理解的程序



程序：为了让计算机执行某些操作或解决某个问题而编写的一系列有序指令的集合

2.3 Java程序

三步走



2.3 Java程序

- Java程序通常包含以下结构：

包声明（Package statement）：如果程序在包内，这是必需的。

导入声明（Import statements）：导入程序中用到的其他类。

类声明（Class declaration）：程序的入口是一个类，通常带有 public 访问修饰符。

方法声明（Method declarations）：类中包含方法定义，这些方法是程序的主要执行逻辑。

主方法（Main method）：是程序的入口点，所有的Java程序都以 main 方法为起点开始执行。

```
package com.example; // 包声明
import java.util.Scanner; // 导入Scanner类
public class HelloWorld { // 类声明
    public static void main(String[] args) {            // 主方法，程序的入口
        Scanner scanner = new Scanner(System.in);        // 创建Scanner对象来从控制台接收输入
        System.out.print("请输入您的名字：");              // 提示用户输入
        String name = scanner.nextLine();                // 读取用户输入
        System.out.println("Hello, " + name + "!");      // 打印输出
        scanner.close();                                 // 关闭scanner
    }
}
```

2.3 Java程序

- **Java程序中的类型主要包括基本数据类型和引用数据类型。**

基本数据类型包括：

整数类型：byte、short、int、long。

浮点数类型：float（32位单精度浮点数）和double（64位双精度浮点数）。

字符类型：char（16位Unicode字符）。

布尔类型：boolean（只有两个值：true和false）。

引用数据类型包括：

类（Class）和接口（Interface）。

数组（Array），可以存储多个同种数据类型的值。

字符串类型（String），常用于存储和操作文本数据。

枚举类型（Enum），用于定义一组命名的常量。

第一个Java程序

外层框架

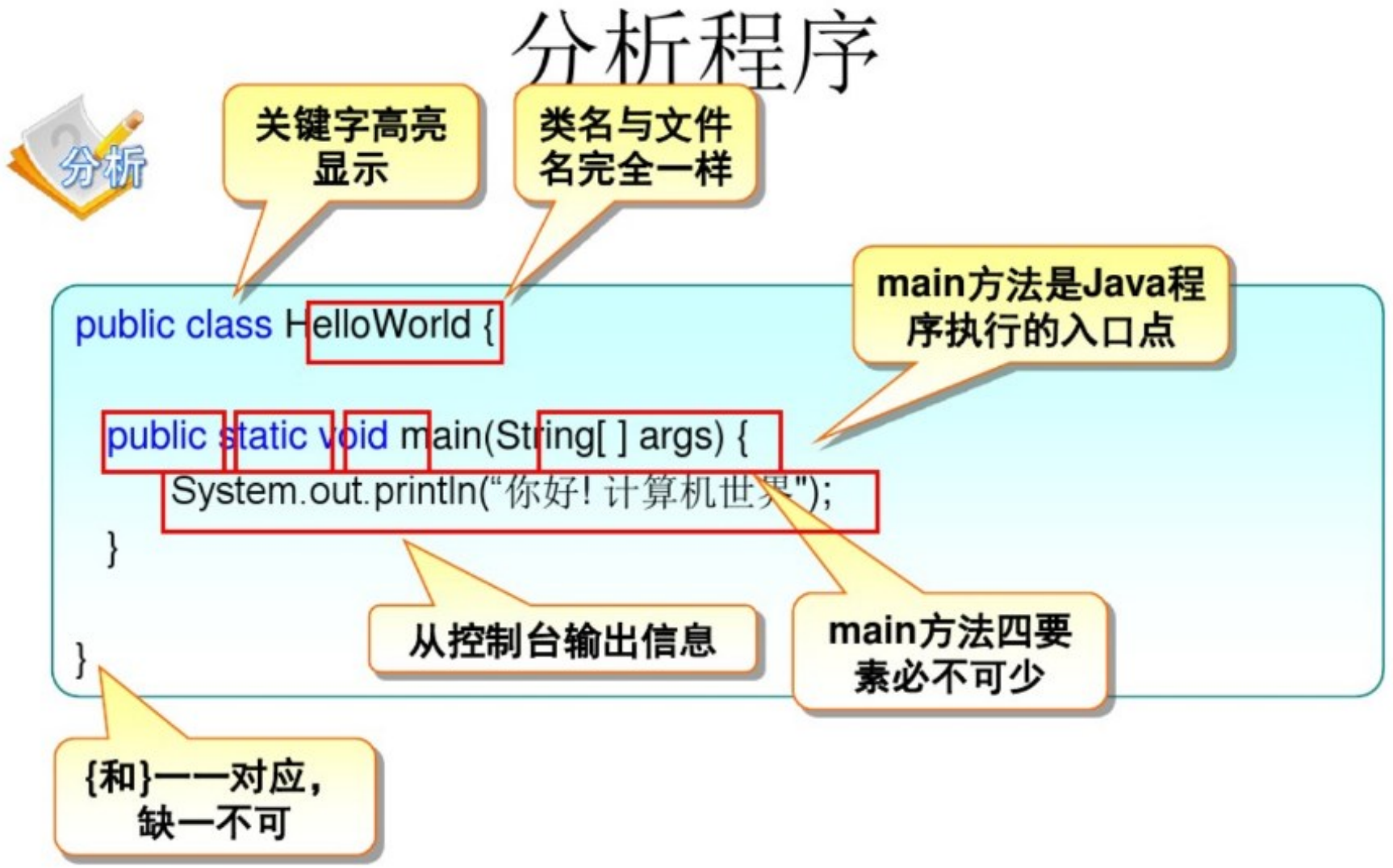
Java入口程序框架

```
public class HelloWorld{

    public static void main(String[ ] args) {
        ...这里填写代码!...
    }

}
```

填写代码


分析程序
分析
关键字高亮显示
类名与文件名完全一样
main方法是Java程序执行的入口点
public class HelloWorld {
public static void main(String[ ] args) {
System.out.println("你好! 计算机世界");
}
}
从控制台输出信息
main方法四要素必不可少
{和}一一对应，缺一不可

小结

1、从控制台打印输出你的姓名和年龄

```
System.out.println("张三");
System.out.println("18");
```

println：输出信息并执行换行

```
System.out.print("张三\n");
System.out.print("18");
```

print：输出信息，但不执行换行
\n：换行符

```
System.out.println("张三\t18");
```

\t：制表位

2、从控制台打印输出“张三 18

Java程序的注释


```
/*
 *  HelloWorld.java
 *  2024年9月
 *
 *    第一个Java程序
 */
public class HelloWorld {
    public static void main(String[ ] args) {
        System.out.println("你好！ 计算机世界");
    }
}
```

文件的名称

日期

功能说明

符号 /* */ 指示中间的语句是该程序中的注释
多行注释以 /* 开始，以 */ 结束

Java程序的注释


```
public class HelloWorld{
    public static void main(String[ ] args) {
        //输出消息到控制台
        System.out.println("你好！ 计算机世界");
    }
}
```

单行注释以 // 开始，以行末结束

Java编码规范


```
class HelloWorld {
    public static void main(String[ ] args) {
        //输出消息到控制台
        System.out.println("你好！计算机世界");
    }
}
```

去掉public，程序可以运行，但不规范；
规范要求类名必须使用public修饰!

常见错误

代码错误

```
public class helloWorld {
    public static void main(String[ ] args) {
        //输出消息到控制台
        System.out.println("你好！ 计算机世界");
    }
}
```

public修饰的类的名称必须与Java文件同名!

常见错误


```
public class HelloWorld {
    public static main(String[ ] args) {
        //输出消息到控制台
        System.out.println("你好！ 计算机世界");
    }
}
```

void

main方法作为程序入口，
void必不可少！

常见错误


```
public class HelloWorld {
    public static void main(String[ ] args) {
        //输出消息到控制台
        system.out.println("你好！ 计算机世界");
    }
}
```

代码错误

编译出错，无法解析system!
Java对大小写敏感!

常见错误


```
public class HelloWorld {
    public static void main(String[ ] args) {
        //输出消息到控制台
        System.out.println("你好！计算机世界")
    }
}
```

;

代码错误

每一条Java语句必须以分号结束!

常见错误


```
public class HelloWorld {
    public static void main(String[ ] args) {
        //输出消息到控制台
        System.out.println("你好！计算机世界);
    }
}
```

"

注意：不要漏写引号!

小结

- 程序运行出现了问题，怎么办？
  - 如何定位错误代码的位置？
  - 如何知道错误的原因？

2.4 数据的分类

- Java的数据类型分为两大类：基本数据类型和引用数据类型。

- 基本数据类型包括6种数字类型（byte、short、int、long、float、double）、1种字符类型（char）、1种布尔类型（boolean）。

- 引用数据类型包括类、接口、数组等。

2.4 数据的分类

- byte

byte 数据类型是8位、有符号的，以二进制补码表示的整数；

最小值是 -128（-2^7）；

最大值是 127（2^7-1）；

默认值是 0；

byte 类型用在大型数组中节约空间，主要代替整数，因为 byte 变量占用的空间只有 int 类型的四分之一；

例子：byte a = 100，byte b = -50。

2.4 数据的分类

- short

short 数据类型是 16 位、有符号的以二进制补码表示的整数

最小值是 -32768（-2^15）；

最大值是 32767（2^15 - 1）；

short 数据类型也可以像 byte 那样节省空间。一个short变量是int型变量所占空间的二分之一；

默认值是 0;

例子：short s = 1000，short r = -20000。

2.4 数据的分类

- int

int 数据类型是32位、有符号的以二进制补码表示的整数；

最小值是 -2,147,483,648（-2^31）；

最大值是 2,147,483,647（2^31 - 1）；

一般地整型变量默认为 int 类型；

默认值是 0 ；

例子：int a = 100000, int b = -200000。

2.4 数据的分类

- long

long 数据类型是 64 位、有符号的以二进制补码表示的整数；

最小值是 -9,223,372,036,854,775,808（-2^63）；

最大值是 9,223,372,036,854,775,807（2^63 -1）；

这种类型主要使用在需要比较大整数的系统上；

默认值是 0L；

例子： long a = 100000L，long b = -200000L。
"L"理论上不分大小写，但是若写成"l"容易与数字"1"混淆，不容易分辩。所以最好大写。

2.4 数据的分类

- float

float 数据类型是单精度、32位、符合IEEE 754标准的浮点数；

float 在储存大型浮点数组的时候可节省内存空间；

默认值是 0.0f；

浮点数不能用来表示精确的值，如货币；

例子：float f1 = 234.5f。

2.4 数据的分类

- double

double 数据类型是双精度、64 位、符合 IEEE 754 标准的浮点数；

浮点数的默认类型为 double 类型；

double类型同样不能表示精确的值，如货币；

默认值是 0.0d；

例子：

```
double   d1  = 7D ;
double   d2  = 7.;
double   d3  =  8.0;
double   d4  =  8.D;
double   d5  =  12.9867;
```

2.4 数据的分类

- boolean

boolean数据类型是布尔型，数据类型表示一位的信息；

只有两个取值：true 和 false；

这种类型只作为一种标志来记录 true/false 情况；

默认值是 false；

例子：boolean one = true。

2.4 数据的分类

- **char**

char 类型是一个单一的 16 位 Unicode 字符；

最小值是 \u0000（十进制等效值为 0）；

最大值是 \uffff（即为 65535）；

char 数据类型可以储存任何字符；

例子：char letter = 'A';。

2.4 数据的分类

2.4.1 Java中的八种基本数据类型

实例

```java
public class PrimitiveTypeTest {
    public static void main(String[] args) {
        // byte
        System.out.println("基本类型：byte 二进制位数：" + Byte.SIZE);
        System.out.println("包装类：java.lang.Byte");
        System.out.println("最小值：Byte.MIN_VALUE=" + Byte.MIN_VALUE);
        System.out.println("最大值：Byte.MAX_VALUE=" + Byte.MAX_VALUE);
        System.out.println();

        // short
        System.out.println("基本类型：short 二进制位数：" + Short.SIZE);
        System.out.println("包装类：java.lang.Short");
        System.out.println("最小值：Short.MIN_VALUE=" + Short.MIN_VALUE);
        System.out.println("最大值：Short.MAX_VALUE=" + Short.MAX_VALUE);
        System.out.println();

        // int
        System.out.println("基本类型：int 二进制位数：" + Integer.SIZE);
        System.out.println("包装类：java.lang.Integer");
        System.out.println("最小值：Integer.MIN_VALUE=" + Integer.MIN_VALUE);
        System.out.println("最大值：Integer.MAX_VALUE=" + Integer.MAX_VALUE);
        System.out.println();

        // long
        System.out.println("基本类型：long 二进制位数：" + Long.SIZE);
        System.out.println("包装类：java.lang.Long");
        System.out.println("最小值：Long.MIN_VALUE=" + Long.MIN_VALUE);
        System.out.println("最大值：Long.MAX_VALUE=" + Long.MAX_VALUE);
        System.out.println();

        // float
        System.out.println("基本类型：float 二进制位数：" + Float.SIZE);
        System.out.println("包装类：java.lang.Float");
        System.out.println("最小值：Float.MIN_VALUE=" + Float.MIN_VALUE);
        System.out.println("最大值：Float.MAX_VALUE=" + Float.MAX_VALUE);
        System.out.println();

        // double
        System.out.println("基本类型：double 二进制位数：" + Double.SIZE);
        System.out.println("包装类：java.lang.Double");
        System.out.println("最小值：Double.MIN_VALUE=" + Double.MIN_VALUE);
        System.out.println("最大值：Double.MAX_VALUE=" + Double.MAX_VALUE);
        System.out.println();

        // char
        System.out.println("基本类型：char 二进制位数：" + Character.SIZE);
        System.out.println("包装类：java.lang.Character");
        // 以数值形式而不是字符形式将Character.MIN_VALUE输出到控制台
        System.out.println("最小值：Character.MIN_VALUE="
                + (int) Character.MIN_VALUE);
        // 以数值形式而不是字符形式将Character.MAX_VALUE输出到控制台
        System.out.println("最大值：Character.MAX_VALUE="
                + (int) Character.MAX_VALUE);
    }
}
```

```
基本类型：byte 二进制位数：8
包装类：java.lang.Byte
最小值：Byte.MIN_VALUE=-128
最大值：Byte.MAX_VALUE=127

基本类型：short 二进制位数：16
包装类：java.lang.Short
最小值：Short.MIN_VALUE=-32768
最大值：Short.MAX_VALUE=32767

基本类型：int 二进制位数：32
包装类：java.lang.Integer
最小值：Integer.MIN_VALUE=-2147483648
最大值：Integer.MAX_VALUE=2147483647

基本类型：long 二进制位数：64
包装类：java.lang.Long
最小值：Long.MIN_VALUE=-9223372036854775808
最大值：Long.MAX_VALUE=9223372036854775807

基本类型：float 二进制位数：32
包装类：java.lang.Float
最小值：Float.MIN_VALUE=1.4E-45
最大值：Float.MAX_VALUE=3.4028235E38

基本类型：double 二进制位数：64
包装类：java.lang.Double
最小值：Double.MIN_VALUE=4.9E-324
最大值：Double.MAX_VALUE=1.7976931348623157E308

基本类型：char 二进制位数：16
包装类：java.lang.Character
最小值：Character.MIN_VALUE=0
最大值：Character.MAX_VALUE=65535
```

2.4 数据的分类

- 小结

数据类型	默认值
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'u0000'
String (or any object)	null
boolean	false

2.4 数据的分类

2.4.1 Java中的八种基本数据类型

- 关键字

关键字	含义
abstract	抽象类或方法
assert	用来查找内部程序错误
break	跳出一个switch或循环
byte	8位整数类型
case	switch的一个分支
catch	捕获异常的try块子句
class	定义一个类类型
continue	在循环末尾继续
default	switch的缺省语句
do	do/while循环最前面的语句
double	双精度浮点数类型
else	if语句的else子句
enum	枚举类型
extends	定义一个类的父类
final	一个常量，或不能覆盖的一个类或方法
finally	try块中总会执行的部分
float	单精度浮点数类型
for	一个循环类型
if	一个条件语句
implements	定义一个类实现的接口
import	导入一个包
instanceof	测试一个对象是否是某个类的实例
int	32位整型数
interface	接口，一种抽象类型，仅有方法和常量的定义
long	64位长整数型
native	由宿主系统实现的一个方法
new	分配新的类实例
null	一个空引用
package	包含类的一个包
private	表示私有字段，或者方法等，只能从类内部访问
protected	表示保护类型字段
public	表示共有属性或者方法
return	从一个方法中返回

2.4 数据的分类

2.4.1 Java中的八种基本数据类型

- 关键字

short	16位整数类型
static	这个特性是这个类特有的，而不属于这个类的对象
strictfp	对浮点数计算使用严格的规则
super	超类对象或构造函数
switch	选择语句
synchronized	对线程而言是原子的方法或代码块
this	当前类的一个方法或构造函数的隐含参数
throw	抛出一个异常
throws	一个方法可能抛出的异常
transient	标志非永久性的数据
try	捕获异常的代码块
void	标记方法不返回任何值
volatile	标记字段可能会被多个线程同时访问，而不做同步
while	一种循环

2.4 数据的分类

- 自动类型转换
- 强制类型转换
- 隐含强制类型转换

下面我们将举例说明:

2.4 数据的分类

- **自动类型转换**

整型、实型（常量）、字符型数据可以混合运算。运算中，不同类型的数据先转化为同一类型，然后进行运算。

转换从低级到高级。

```
低  ------------------------------------>  高

byte,short,char-> int -> long-> float -> double
```

- **数据类型转换必须满足的规则**

1. 不能对boolean类型进行类型转换。
2. 不能把对象类型转换成不相关类的对象。
3. 在把容量大的类型转换为容量小的类型时必须使用强制类型转换。
4. 转换过程中可能导致溢出或损失精度，因为 byte 类型是 8 位，最大值为127，所以当 int 强制转换为 byte 类型时，值 128 时候就会导致溢出。例如：

```
int i =128;
byte b = (byte)i;
```

5. 浮点数到整数的转换是通过舍弃小数得到，而不是四舍五入，例如：

```
(int)23.7 == 23;
(int)-45.89f == -45
```

• 自动类型转换实例

必须满足转换前的数据类型的位数要低于转换后的数据类型，例如: short数据类型的位数为16位，就可以自动转换位数为32的int类型，同样float数据类型的位数为32，可以自动转换为64位的double类型。转换从低级到高级。

实例

```
public class ZiDongLeiZhuan{
        public static void main(String[] args){
            char c1='a';//定义一个char类型
            int i1 = c1;//char自动类型转换为int
            System.out.println("char自动类型转换为int后的值等于"+i1);
            char c2 = 'A';//定义一个char类型
            int i2 = c2+1;//char 类型和 int 类型计算
            System.out.println("char类型和int计算后的值等于"+i2);
        }
}
```

运行结果为:

```
char自动类型转换为int后的值等于97
char类型和int计算后的值等于66
```

解析: c1 的值为字符 **a** ,查 ASCII 码表可知对应的 int 类型值为 97， A 对应值为 65，所以 `i2=65+1=66`。

● 强制类型转换

1. 条件是转换的数据类型必须是兼容的。

2. 格式：(type)value type是要强制类型转换后的数据类型 实例：

实例

```
public class ForceTransform {
    public static void main(String[] args){
        int i1 = 123;
        byte b = (byte)i1;//强制类型转换为byte
        System.out.println("int强制类型转换为byte后的值等于"+b);
    }
}
```

运行结果：

```
int强制类型转换为byte后的值等于123
```

2.4 数据的分类

- **隐含强制类型转换**

1. 整数的默认类型是 int。
2. 小数默认是 double 类型浮点型，在定义 float 类型时必须在数字后面跟上 F 或者 f。

下面将举例说明:

2.4 数据的分类

- 在Java中，隐式强制类型转换是指在不需要显式使用类型转换操作的情况下，数据类型可以自动从一种类型转换到另一种类型。这通常发生在将子类对象赋值给父类类型的变量时，或者在方法调用中传递参数时，子类的对象会隐式转换为父类类型。

代码示例:

eg2.4.3.txt

在上述代码中，Dog 类型的对象被隐式转换为 Animal 类型，并赋值给了 myAnimal 变量。尽管 myAnimal 的声明类型是 Animal，但当调用 eat 方法时，实际执行的是 Dog 类的 eat 方法，这是因为 Dog 对象被视为 Animal 类型，但保留了 Dog 的所有特性。

需要注意的是，隐式转换并不总是安全的。如果尝试调用仅在子类中定义，而不在父类中定义的方法，会导致编译错误。这是因为编译器只知道对象是父类类型，不知道它实际上是子类类型，从而无法保证该方法在运行时可用。

2.4 数据的分类

- 引用类型

包含: 类类型,接口类型,数组类型

在Java中，引用类型的变量非常类似于C/C++的指针。引用类型指向一个对象，指向对象的变量是引用变量。这些变量在声明时被指定为一个特定的类型，比如 Employee、Puppy 等。变量一旦声明后，类型就不能被改变了。

对象、数组都是引用数据类型。

所有引用类型的默认值都是null。

一个引用变量可以用来引用任何与之兼容的类型。

例子：Site site = new Site("Runoob")。

2.4 数据的分类

- **类类型的引用**

在这个例子中，obj是一个MyClass类型的对象，ref是一个引用，它指向obj对象。

Java

```java
public class MyClass {
    int data;
}

public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass();
        MyClass ref = obj;
    }
}
```

2.4 数据的分类

- **接口类型的引用**

在这个例子中，obj是一个实现了MyInterface接口的类的对象，ref是一个引用，它指向obj对象。

Java

```java
interface MyInterface {
    void display();
}

public class Main implements MyInterface {
    public void display() {
        System.out.println("Hello, World!");
    }

    public static void main(String[] args) {
        Main obj = new Main();
        MyInterface ref = obj;
    }
}
```

2.4 数据的分类

- 接口类型的引用

代码示例 (详情说明)

在Java中，接口类型的引用可以指向实现了该接口的类的实例。接口定义了一组方法规范，实现类必须实现这些方法。通过使用接口类型的引用，可以调用接口中定义的方法，而不需要关心具体实现类是哪一个。

在这个例子中，Animal 是一个接口，Dog 和 Cat 是实现了 Animal 接口的两个类。在 main 方法中，我们创建了 Animal 类型的引用 myPet，并首先将其指向 Dog 的实例，然后又将其指向 Cat 的实例。通过这个引用，我们调用了 makeSound 方法，分别输出了狗和猫的叫声。这演示了接口类型的引用如何指向不同的实现类实例，并能够调用具体实现类的方法。

eg2.4.4.txt

2.4 数据的分类

- **数组类型的引用**

在这个例子中，arr是一个包含5个整数的数组，ref是一个引用，它指向arr数组。

注意：在Java中，引用类型的变量在32位系统中大小是一样的，都是4字节，而在64位系统中大小是一样的，都是8字节。这个大小是指引用本身所占用的内存大小，不是指引用所指向的对象的大小。

Java

```java
public class Main {
    public static void main(String[] args) {
        int[] arr = new int[5];
        int[] ref = arr;
    }
}
```

2.4 数据的分类

- **数组类型的引用**

再用5个例子说明一下

在Java中，数组是一个对象，并且可以被赋予一个引用变量。例如，你可以创建一个数组，然后将它赋给一个变量。这个变量就是数组的引用。

解决方案1：声明并初始化一个数组

```
int[] myArray = new int[5];
```

在这个例子中，myArray就是一个引用变量，它指向一个包含5个整数的数组。

2.4 数据的分类

- **数组类型的引用**

解决方案2：将一个数组赋值给另一个数组

```
int[] myArray1 = new int[5];
int[] myArray2 = myArray1;
```

在这个例子中，myArray2也是一个引用变量，它现在指向myArray1所指向的数组。所以，myArray1和myArray2实际上是指向同一个数组的引用。

2.4 数据的分类

- **数组类型的引用**

解决方案3：更改数组中的元素

```
int[] myArray = new int[5];
myArray[0] = 10;
```

在这个例子中，我们更改了数组中的第一个元素，即索引为0的元素。

2.4 数据的分类

- **数组类型的引用**

解决方案4：通过方法改变数组的引用

```
public class Test {
    static void changeArray(int[] arr) {
        arr[0] = 100;
    }

    public static void main(String[] args) {
        int[] myArray = new int[5];
        changeArray(myArray);
        System.out.println(myArray[0]);  // 输出100
    }
}
```

在这个例子中，我们创建了一个方法changeArray，它接受一个int[]类型的参数。我们传入myArray作为参数，并在方法内部更改了数组中的元素。这样，myArray所引用的数组的元素值已经被改变了。

2.4 数据的分类

- **数组类型的引用**

解决方案5：通过方法改变数组的引用

```
public class Test {
    static int[] changeArray() {
        int[] newArray = {1, 2, 3, 4, 5};
        return newArray;
    }

    public static void main(String[] args) {
        int[] myArray = changeArray();
        for (int i = 0; i < myArray.length; i++) {
            System.out.print(myArray[i] + " ");
        }
    }
}
```

在这个例子中，我们创建了一个方法changeArray，它返回一个int[]类型的值。我们在main方法中调用这个方法，并将返回的值赋给myArray。这样，myArray就引用了新的数组。

2.4 数据的分类

- 关于输入输出

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("ScannerTest, Please Enter Name:");
    String name = sc.nextLine();  //读取字符串型输入
    System.out.println("ScannerTest, Please Enter Age:");
    int age = sc.nextInt();    //读取整型输入
    System.out.println("ScannerTest, Please Enter score:");
    float score = sc.nextFloat(); //读取float型输入
    System.out.println("Your Information is as below:");
    System.out.println("Name:" + name +"\n" + "Age:"+age + "\n"+"score:"+score);
    sc.close();//用完一定要关闭
}
```

本章重点

- 掌握基本数据类型的特性

  byte、short、int、long、float、double、char、boolean

- 掌握自动类型转换优先级

本章作业

- 编写并运行本章实例代码,观察返回结果
- 编写一个输入输出的实例,要求输入并返回患病姓名,年龄,病症,药品,价格等信息

本章作业

- 实现一个四则运算器

要求:它能够处理加、减、乘、除四种基本运算，并且支持用户连续输入多个表达式进行计算，直到用户选择退出。加入错误处理，以确保在除数为0等特殊情况时能够给出合理且友好提示。

举例:

用户在控制台输入: “1+1”然后回车,程序计算结果并输出“2”

“2*5”然后回车,程序计算结果并输出“10”

.....

计算机程序设计课程学习平台

面向河南中医药大学智能医学工程专业使用

河南中医药大学信息技术学院（智能医疗行业学院）与河南方和信息科技有限公司 联合建设

河南中医药大学信息技术学院互联网技术教学团队