## 基础

### HelloWorld

1、代码

public class **HelloWorld**

{

    public static void **main**(String[] argv)

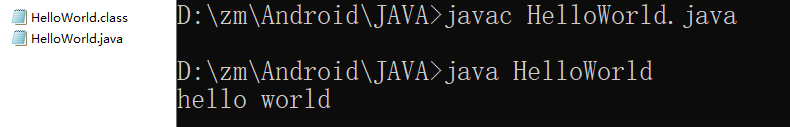
    {

        System.out.**println**("hello world");

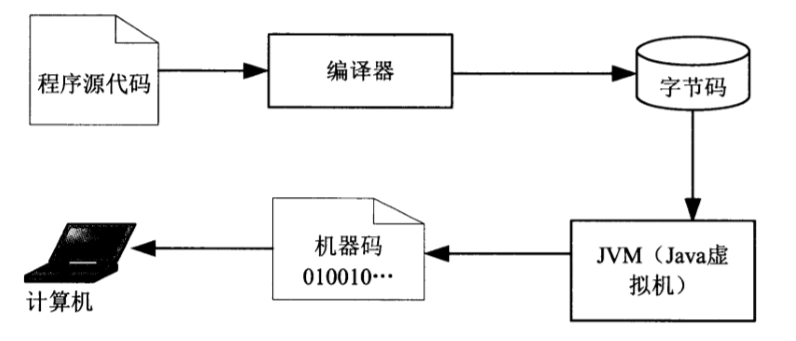
    }

}

2、执行



3、java程序运行流程



### JVM、JRE、JDK

**jvm：**java虚拟机，负责在不同平台执行java字节码文件

**jre：** java运行环境，包含了jvm

**jdk：** java开发环境，包含了jre

### 标识符

**含义：**给类、变量、方法等取名字的规范

**命名规范(强制)：**

26个英文字符，大小写都可以，0-9数字，$，\_

不能以数字开头

不能是关键字

**命名规范（建议）：**

**类名：**首字母大写，门面每个单词首字母大写

**方法、变量：**首字母小写，门面每个单词首字母大写

### 关键字

#### Import 导包：

Import 包名.类名;

#### final 不可改变:

修饰类，不能继承

修饰方法，不能被重写

修饰变量，不可以重新复制

引用类型，不可以改变指向

成员变量，可以在 构造 或者 声明 的时候复制

### 方法

**方法重载**，允许方法名称相同，但是参数不同（参数个数、类型、顺序皆可 ）

**注意：**

1. 数组作为参数时，传入的是地址
2. 数组作为返回值时，返回的是地址
3. 对象作为参数时，传入的是地址
4. 对象作为返回值时，返回的是地址

### String

### 面向对象

#### 构造方法

java中的类会默认提供一个空构造；

如果自己提供了任意构造，空构造将不存在

#### 继承

Java语言是单继承的

**语法：** public Zi extends Fu

**Super:**

1、访问父类中成员变量和成员方法 super.

2、子类会默认调用父类的无参构造 super()，也可以自己在子类中调用父类其他构造，super(xx,xx,...)，super()方法必须要放在子类的第一个语句

**方法重写：**

子类继承父类时，使用和父类一样的方法名和参数，同时返回值范围小于等于父类，访问权限大于等于父类，即可覆盖父类方法，

可以在方法上方加@Override检测是否覆盖

#### 抽象类和方法

**关键字：**abstract

1. 抽象类直接创建对象
2. 子类必须重写抽象类的所有抽象方法，除非子类也是抽象类
3. 只要有一个抽象方法就必须是抽象类

#### 接口

接口是一个特殊的抽象类

**关键字：**interface

**接口中可以有：**

1. 成员变量，只能是 punlic static void 类型的
2. 抽象方法 public abstract
3. 默认方法 default ，相当于成员方法，但是继承多个接口时，需要重写冲突的默认方法 public default java1.8
4. 静态方法，只能通过接口名调用 public static java1.8
5. 私有方法，可以有静态私有方法和普通私有方法

#### 多态

多态就是父类引用，保存子类对象

**判断变量是什么数据类型：**

变量名 instanceof 数据类型

引用名 instanceof 子类名称

变量属于该类型返回true，否则返回false

#### 内部类

**特点**

内部类可以直接访问外部类的成员，包括私有成员。

外部类要访问内部类的成员，必须要建立内部类的对象

**使用**

*/WaiBu.NeiBu w = new WaiBu().new NeiBu();*

*//w.run();*

*//创建对象*

WaiBu w1 =new WaiBu();

NeiBu n = w1.new NeiBu();

*//调用外部类成员*

System.out.println(w1.str) ;

*//调用外部类成员*

n.run();

**匿名内部类**

**匿名内部类必须继承一个父类或者实现一个父接口。**



#### 继承自object类的方法

##### toString方法

public String toString()

说明：默认打印 *对象的类型+@+内存地址值*

示例：

// 如果对象地址一样，则认为相同

if (this == o)

return true;

// 如果参数为空，或者类型信息不一样，则认为不同

if (o == null || getClass() != o.getClass())

return false;

// 转换为当前类型

Person person = (Person) o;

##### equals方法

public boolean equals(Object obj)

说明：比较两个对象是否相等，默认是比较地址值，和 == 一样