基础[1]

> - 介绍
> - 基础语法

目录

一、介绍

1、认识

> `Java`是一门**面向对象编程语言**，不仅吸收了**C++语言的各种优点，还摒弃了C++里难以理解的多继承、指针等概念**，因此Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程 。

`Java`语言具有**简单性、面向对象、分布式、健壮性、安全性、跨平台性、可移植性、多线程与动态性等特点**。`Java`语言可以编写**桌面应用程序、Web应用程序、分布式系统和嵌入式系统等** 。Java 快速、安全、可靠。

- Java语言是简单的
- Java语言是面向对象的
- Java语言是分布式的
- Java语言是多线程的
- Java语言是高性能的
- Java语言是跨平台的
- Java语言是可移植的
- Java语言是安全的
- Java语言是健壮的
- Java语言是动态的

2、JDK、JRE、JVM

1、JVM

> **Java Virtual Machine**是Java虚拟机，**Java程序需要运行在虚拟机上**，不同的平台有自己的虚拟机，因此**Java语言可以实现跨平台**。

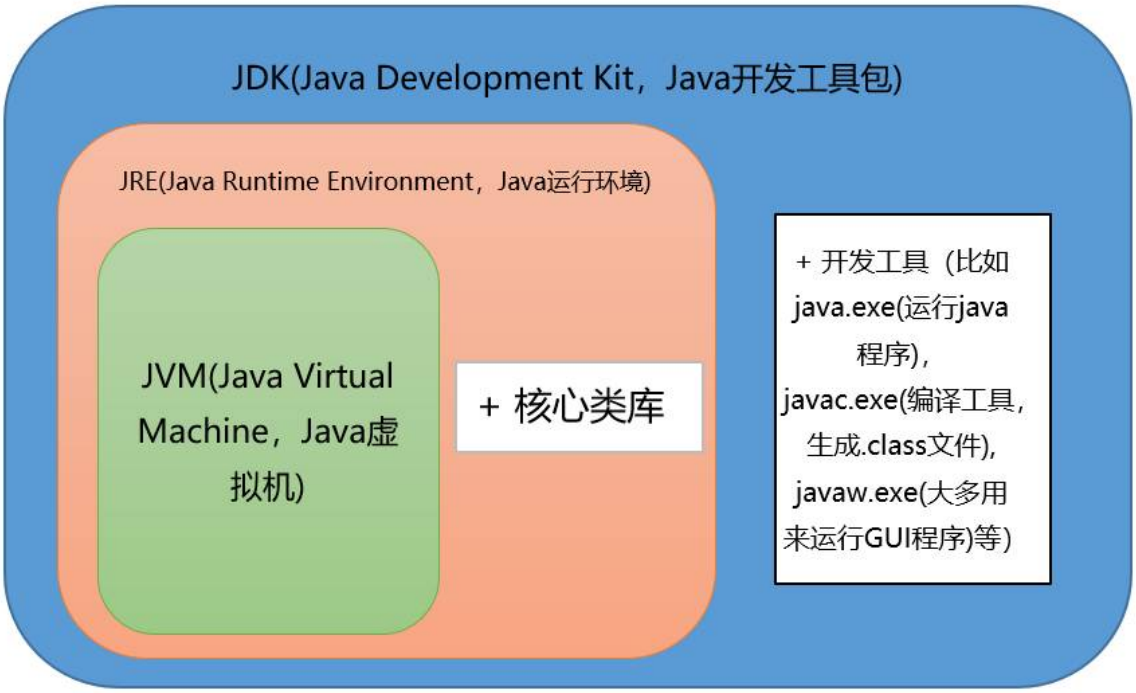
2、jre

> **Java Runtime Environment**包括**Java虚拟机和Java程序所需的核心类库等**。核心类库主要是java.lang包：包含了运行Java程序必不可少的系统类，如基本数据类型、基本数学函数、字符串处理、线程、异常处理类等，系统缺省加载这个包

3、jdk

> **Java Development Kit**是提供给Java开发人员使用的，**其中包含了Java的开发工具**，也**包括了JRE**。所以安装了JDK，就无需再单独安装JRE了。其中的开发工具：**编译工具(javac.exe)，打包工具(jar.exe)**等

4、三者基本关系



3、基本编译

> **文件名需要和类名一致**

1、编译

> **javac**
>
> - 示例

```
public class Hello{
    public static void main(String[] args){
        System.out.println("Hello World!!!");
    }
}
```

> - 编译

```
javac Hello.java
```

2、运行

> **java**

```
java Hello
```

3、运行jar包

1、直接运行

```
java -jar xxx.jar
```

> 最简单的启动方式，但是会**占用当前终端窗口**，可以打断程序运行，或者关闭终端。

2、后台运行（&）

```
java -jar xx.jar &
```

> `&`代表在后台运行，**停止程序后还是会继续执行**

- 结果输出到终端
- 关闭命令（`Ctrl+C`），程序继续
- 关闭`SSh`，程序关闭

3、后台运行（nohup）

```
nohup java -jar xx.jar &
#默认输出到nohup.out
nohup java -jar xxx.jar>xx.log &
#输出到xx.log文件中
```

> `nohup` --->`no hang up`
>
> **关闭SSh链接，程序也还是继续运行。**

- 结果默认输出到 `nohup.out`
- 关闭命令（`Ctrl+C`），程序关闭
- 关闭`SSh`，程序继续

二、基础语法

- 注释
- 变量

1、注释

> 用于解释说明程序的文字，注释不影响程序执行

1.1、单行注释

```
// 注释文字
```

1.1、多行注释

```
/*
注释文字
注释文字
*/
```

1.3、文档注释

> 文档注释的内容是可以提取到一个程序说明文档中的

```
/**
    注释内容
    注释内容
*/
```

2、变量

> 在程序执行的过程中，**在某个范围内其值可以发生改变的量**。从本质上讲，变量其实是内存中的一小块区域。

1、声明位置区分

1.1、成员变量

> 位置：**方法外部，类内部**

1.1.1、实例变量

> (非静态变量)**独立于方法之外的变量，不过没有 static 修饰。**

- **实例变量声明在一个类中，但在方法、构造方法和语句块之外**；
- 当一个对象被实例化之后，每个实例变量的值就跟着确定；
- **实例变量在对象创建的时候创建，在对象被销毁的时候销毁**；
- 实例变量的值应该至少被一个方法、构造方法或者语句块引用，使得外部能够通过这些方式获取实例变量信息；
- 实例变量可以声明在使用前或者使用后；
- **访问修饰符可以修饰实例变量**；
- 实例变量对于类中的方法、构造方法或者语句块是可见的。一般情况下应该把实例变量设为私有。通过使用访问修饰符可以使实例变量对子类可见；
- **实例变量具有默认值。**数值型变量的默认值是0，布尔型变量的默认值是false，引用类型变量的默认值是null。变量的值可以在声明时指定，也可以在构造方法中指定；
- 实例变量可以直接通过变量名访问。但在静态方法以及其他类中，就应该使用完全限定名：ObejectReference.VariableName。

1.1.2、类变量

> (静态变量)**独立于方法之外的变量，用 `static` 修饰。**

- 类变量也称为静态变量，在类中以static关键字声明，但必须在方法构造方法和语句块之外。
- 无论一个类创建了多少个对象，类只拥有类变量的一份拷贝。
- 静态变量除了被声明为常量外很少使用。常量是指声明为public/private，final和static类型的变量。常量初始化后不可改变。
- 静态变量储存在静态存储区。经常被声明为常量，很少单独使用static声明变量。
- **静态变量在第一次被访问时创建，在程序结束时销毁。**
- 与实例变量具有相似的可见性。但为了对类的使用者可见，大多数静态变量声明为public类型。
- **默认值和实例变量相似。**数值型变量默认值是0，布尔型默认值是false，引用类型默认值是null。变量的值可以在声明的时候指定，也可以在构造方法中指定。此外，静态变量还可以在静态语句块中初始化。
- 静态变量可以通过：ClassName.VariableName的方式访问。

- 类变量被声明为public static final类型时，类变量名称一般建议使用大写字母。如果静态变量不是public和final类型，其命名方式与实例变量以及局部变量的命名方式一致。

1.1.3、区别

- 调用方式
  - 静态变量也称为类变量，可以直接通过类名调用。也可以通过对象名调用。这个变量属于类。
  - 成员变量也称为实例变量，只能通过对象名调用。这个变量属于对象。
- 存储位置
  - 静态变量存储在方法区长中的静态区。
  - 成员变量存储在堆内存。
- 生命周期
  - 静态变量随着类的加载而存在，随着类的消失而消失。生命周期长。
  - 成员变量随着对象的创建而存在，随着对象的消失而消失。
- 与对象的相关性
  - 静态变量是所有对象共享的数据。
  - 成员变量是每个对象所特有的数据。

1.2、局部变量

- 局部变量：类的**方法中的变量。**
- 局部变量**声明在方法、构造方法或者语句块中**；
- **局部变量在方法、构造方法、或者语句块被执行的时候创建，当它们执行完成后，变量将会被销毁**；
- 访问修饰符不能用于局部变量；
- 局部变量只在声明它的方法、构造方法或者语句块中可见；
- 局部变量是**在栈上分配的。**
- **局部变量没有默认值，所以局部变量被声明后，必须经过初始化，才可以使用。**

1.3、区别

- 作用域
  - 成员变量：针对整个类有效。
  - 局部变量：只在某个范围内有效。(一般指的就是方法,语句体内)
- 存储位置
  - 成员变量：**随着对象的创建而存在，随着对象的消失而消失，存储在堆内存中。**
  - 局部变量：在方法被调用，或者语句被执行的时候存在，**存储在栈内存中。**当方法调用完，或者语句结束后，**就自动释放。**
- 生命周期
  - 成员变量：随着对象的创建而存在，随着对象的消失而消失
  - 局部变量：当方法调用完，或者语句结束后，就自动释放。
- 初始值
  - 成员变量：**有默认初始值。**
  - 局部变量：**没有默认初始值，使用前必须赋值。**

1.4、使用原则

> **在使用变量时需要遵循的原则为：就近原则**
>
> **首先在局部范围找，有就使用；接着在成员位置找。**

2、所属数据类型区分

2.1、基本数据类型

类型	名称	默认值	占用字节	示例
整形	**byte**	0	1	byte a=10;
整形	**short**	0	2	short b=10;
整形	**int**	0	4	int c=10;
整形	**long**	0	8	**long d=100L;**
浮点型	**float**	0.0	4	**float e=10.02F;**
浮点型	**double**	0.0	8	double f=10.02;
字符型	**char**	\u0000	2	char g='A'
布尔型	**boolen**	false	1	boolen h=true;

注意

> 1. 整数默认是int类型，定义long类型的数据时，**要在数据后面加L。**
> 2. 浮点数默认是double类型，定义float类型的数据时，**要在数据后面加F。**

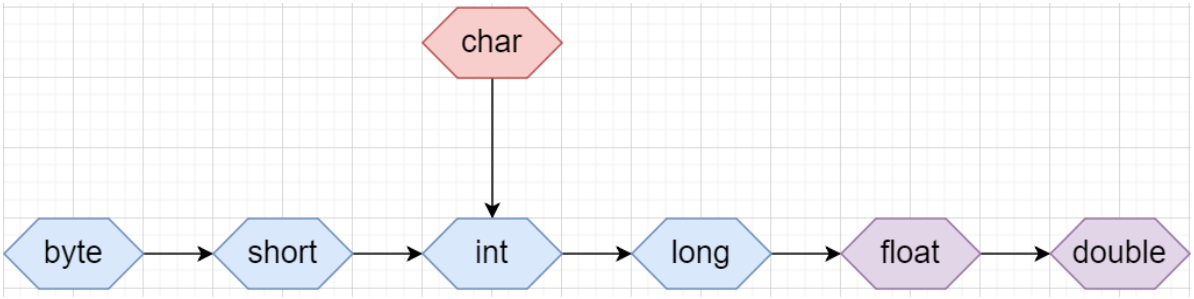
2.2、引用数据类型

- **数组**
- **类**
- **接口**

3、定义格式

> **数据类型 变量名 = 初始化值;**
>
> **注意：格式是固定的，记住格式，以不变应万变**

4、数据类型转换



1、自动类型转换

> （隐式类型转换）

- 小的类型自动转化为大的类型
- 整数类型可以自动转化为浮点类型，可能会有误差
- 字符可以自动提升为整数

2、强制类型转换

> （显示类型转换）

- 可能导致溢出或损失精度
- 浮点数强制转换为整数，是舍弃小数部分，不是四舍五入

5、注意事项

> **类外面不能有变量的声明**