Java基本语法

```
运算符：增加了无符号右移， instanceof 类型比较操作符
& | ^ 均可以对 boolean 类型操作 ~ 不行
算数运算符： + - * / % ++ -- -
关系运算符： == ≠ > ≥ < ≤
逻辑运算符： && || !
按位运算符： & | ~（按位非）^（按位异或）
    PS：整型、字符型 不同数据长度运算前要对齐
移位运算符： <<（符号被挤掉，右侧空位补0）>>（左侧空位由符号位不上）
        >>>（左侧空位用0补上）
赋值运算符：= += -= *= /=
```

```
类型转化
    自动：
    短变长 int → float
    子变父 Tree t = (Tree)oak;
    强制
    长变短 byte b = (byte) 10;
    父变子 Oak oak = (Oak)tree;
```

```
true 和 false
    Java 中只能使用 true 和 false 来决定路径;
switch
    只能允许整数值(int / char)
    非整数值用if-else;
for
    没有goto
    标签的使用：
        多重循环嵌套
        程序员想从多重循环跳出;

```

OOP

oop 的三大技术：封装、继承、多态

```
对象引用：
    对象引用未初始化的时候，初值为null
    存放的是对象的句柄
    对象作为函数参数时，传递的是对象引用，因此函数中改变会影响到函数外
    基本类型的变量作为函数参数时，传递的是值
```

```
对象实例化
    实例化的过程实际上是为该对象分配内存
    当一个对象实例不被任何变量引用时，Java会自动回收内存空间;
main 方法
    是程序运行的起点
    public static void main(String args[]);
finalize 方法
    实行垃圾收集之前，Java会自动调用该方法
    protected void finalize() throws Throwable
```

```
类方法
    构造方法：
        与类名相同，无返回值
        在创建对象实例时，由 new 运算符自动调用
        可有多个具有不同参数列表的构造方法
        没有返回值
    作用：
        在对象诞生时，给对象一个初始值
        系统赋的初值不总是合适的
```

```
缺省值
    boolean false
    char \u0000 (null)
    byte (byte)0
    short (short)0
    int 0
    long 0L
    float 0.0f
    double 0.0d
    引用 null
仅在作为类成员时有意义
```

	private	no modifier	protected	public
同一类	yes	yes	yes	yes
同一文件	no	yes	yes	yes
同一包中的子类	no	yes	yes	yes
不同包中的子类	no	no	yes	yes
其它类	no	no	no	yes

```
static 修饰符
    既可以修饰成员数据，也可以修饰成员函数
    静态成员被该类的所有对象共享
    即使还没有建立对象，静态成员函数就已经存在
```

```
extends
    子类继承了父类的所有数据和方法，并加以扩充
    子类对父类中的成员的存取权限由父类成员的存取权限修饰符决定
this 和 super
    用在类成员函数的定义中
    静态成员函数中不能使用 this 引用
可以将子类的实例赋给父类的引用
```

```
方法重载
    同一类中有多个名字相同的函数时候，相互之间依靠不同的参数列表区分
    不能依靠函数的返回值区分重载函数
```

```
方法隐藏
    子类中的成员函数将隐藏父类中的同名函数
    子类中的成员变量也将隐藏父类中的同名变量
```

```
abstract
    用abstract修饰的方法只需给出原型说明而无需具体实现
    带有abstract的类都必须声明为抽象类
    抽象类不能创建对象实例
    抽象类的子类需要完成父类中的所有抽象方法，或者自己也定义成抽象类

```

包和接口

```
Array
    数组是对象
    Java中没有静态分配的数组定义，数组的内存都是动态分配的
    自动检查下标是否越界
    Java没有多维数组，只有数组的数组
    数组是类，只有一个成员变量length
```

```
字符串
    Java中的字符串是对象
    String是不变字符串，StringBuffer是可变字符串
    String和StringBuffer类都是final的，都不能被继承
    equals() 和 == 的区别：前者是比较内容是否相等，后者是比较引用的是否是一个对象
```

```
包
    包是类的容器，用于分隔命名空间
    位于java源程序的第一句
    未指定packge属于无名的缺省包，缺省包中的类不能被其他包的类引用
    包可以包含任意个类，可以包含包，包名要全部小写
import
    制定编译器去哪儿找包
    不加载包中的类
    必须出现在所有类定义之前
```

```
多重继承
    不同类可以实现同一个接口
    一个类可以实现多个接口，用逗号分割
    instanceof用来识别一个类是否实现了一个接口
    不能创建接口的实例，但是可以创建对接口的引用
```

```
多态
    1.父类对象可以强制转化为子类对象，但是前提是父类对象为子类对象实例化的结果
```

容器类

```
Collection：一个元素序列
Map：一组成对的键值对对象，Map允许通过一个对象查找另一个对象
ArrayList
    是大小可变数组，保存的是Object，任何java类对象都可以存放，类型必须是Object的子类
    当指定了元素类型后，该类型的子类也可以通过向上转型加入到容器
    Array.asList返回List，底层表示的是数组，不能调整尺寸，使用add()或delete()将报错

```

方法摘要	
void	clear () 移除此列表中的所有元素。
boolean	contains (Object o) 如果此列表中包含指定的元素，则返回 true。
E	get (int index) 返回此列表中指定位置上的元素。
int	indexOf (Object o) 返回此列表中首次出现的指定元素的索引，或如果此列表不包含元素，则返回 -1。
boolean	isEmpty () 如果此列表中没有元素，则返回 true
int	lastIndexOf (Object o) 返回此列表中最后一次出现的指定元素的索引，或如果此列表不包含索引，则返回 -1。
E	remove (int index) 移除此列表中指定位置上的元素。
E	set (int index, E element) 用指定的元素替代此列表中指定位置上的元素。
int	size () 返回此列表中的元素数。
Object []	toArray () 按适当顺序（从第一个到最后一个元素）返回包含此列表中所有元素的数组。
T[]	toArray (T[] a) 按适当顺序（从第一个到最后一个元素）返回包含此列表中所有元素的数组；返回数组的运行时类型是指定数组的运行时类型。

List

	随机访问	插入删除	接口数量
ArrayList	快	慢	少
LinkedList	慢	快	多

迭代器

- 使用iterator()函数获取容器的迭代器对象
- 使用next()获得下一个元素
- 使用hasNext()检查序列中是否还有元素
- 使用remove()将迭代器新近返回的元素删除

Set

- HashSet是无序的，采用hash算法，所以查询速度快。
- TreeSet是有序的（按字典序）

异常处理

```
Error：JVM系统内部错误、资源耗尽等严重情况
Exception：其它因编程错误或偶然的外在因素导致的一般性问题
    空指针访问
    试图读取不存在的文件
    网络连接中断
```

```
RuntimeException
    错误的类型转换
    数组下标越界
    空指针访问
IOException
    从一个不存在的文件中读取数据
    越过文件结尾继续读取
    连接一个不存在的URL
运行时异常即使没有使用try catch，Java也能自己捕获，并且编译通过
如果抛出的是IOException，则必须捕获，否则编译错误

```

分类	字节输入流	字节输出流	字符输入流	字符输出流
抽象基类	InputStream	OutputStream	Reader	Writer
访问文件	FileInputStream	FileOutputStream	FileReader	FileWriter
访问数组	ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArrayWriter
访问管道	PipedInputStream	PipedOutputStream	PipedReader	PipedWriter
访问字符串			StringReader	StringWriter
缓冲流	BufferedInputStream	BufferedOuputStream	BufferedReader	BufferedWriter
转换流			InputStreamReader	OutputStreamWriter
对象流	ObjectInputStream	ObjectOutputStream		
抽象基类	FilterInputStream	FilterOutputStream	FilterReader	FilterWriter
打印流		PrintStream		PrintWriter
推回输入流	PushbackInputStream		PushbackReader	
特殊流	DataInputStream	DataOutputStream		

```java
// 利用字符输入输出流，完成hello.txt文件复制，复制为hello3.txt
    static void testCopyWithReaderAndWriter() throws IOException {
        // 1.创建hello.txt的文件输入流
        Reader in = new FileReader("hello.txt");
        // 2.创建hello2.txt的文件输出流
        Writer out = new FileWriter("hello3.txt");
        // 3.创建一个byte数组，用于读写文件
        char[] buffer = new char[1024 * 10];
        int len = 0;
        // 4.读写文件：注意，写文件用write(char[]buf,int offset,int len).
        // 而不能直接使用write(char[]buf)
        while ((len = in.read(buffer)) != -1) {
            out.write(buffer, 0, len);
        }
        // 5.关闭流
        out.close();
```

```
17            in.close();
18        }
```

线程

- 新建状态
    - 当创建了一个Thread对象时候，该对象就处于新建状态
    - 没有启动，因此无法运行
- 可执行状态
    - 当其他线程调用了处于新建状态线程的start方法，该线程对象将转换到可执行状态
    - 拥有获得CPU控制权的机会，处于等待调度阶段
- 运行状态
    - 可以调用yield方法，将主动让出CPU控制权
- 阻塞状态
    - 调用sleep方法
    - 调用join方法
    - 执行IO操作
- 死亡状态
    - 一旦从run方法返回，无论是正常退出还是抛出异常，都会进入死亡状态
    - 可以使用Thread类的isAlive方法判断线程是否活着

synchronized关键字