# Java学习笔记

## 基础

### ****Java基础****

* 面向对象的特征：继承、封装和多态
* final, finally, finalize 的区别
* Exception、Error、运行时异常与一般异常有何异同
* 请写出5种常见到的runtime exception
* int 和 Integer 有什么区别，Integer的值缓存范围
* 包装类，装箱和拆箱
* String、StringBuilder、StringBuffer
* 重载和重写的区别
* 抽象类和接口有什么区别
* 说说反射的用途及实现
* 说说自定义注解的场景及实现
* HTTP请求的GET与POST方式的区别
* Session与Cookie区别
* 列出自己常用的JDK包
* MVC设计思想
* equals与==的区别
* hashCode和equals方法的区别与联系
* 什么是Java序列化和反序列化，如何实现Java序列化？或者请解释Serializable 接口的作用
* Object类中常见的方法，为什么wait  notify会放在Object里边？

因为synchronized中的这把锁可以是任意对象，所以任意对象都可以调用wait()和notify()；所以wait和notify属于Object。

* Java的平台无关性如何体现出来的

最主要的是Java平台本身。Java平台扮演Java程序和所在的硬件与操作系统之间的缓冲角色。这样Java程序只需要与Java平台打交道，而不用管具体的操作系统。

Java语言保证了基本数据类型的值域和行为都是由语言自己定义的。而C/C++中，基本数据类是由它的占位宽度决定的，占位宽度由所在平台决定的。不同平台编译同一个C++程序会出现不同的行为。通过保证基本数据类型在所有平台的一致性，Java语言为平台无关性提供强有力的支持。

Java class文件。Java程序最终会被编译成二进制class文件。class文件可以在任何平台创建，也可以被任何平台的Java虚拟机装载运行。它的格式有着严格的定义，是平台无关的。

可伸缩性。Sun通过改变API的方式得到三个基础API集合，表现为Java平台不同的伸缩性：J2EE,J2SE,J2ME。

* JDK和JRE的区别
* Java 8有哪些新特性

Lambda 表达式 − Lambda允许把函数作为一个方法的参数（函数作为参数传递进方法中。

Java8 内置的四大核心函数式接口

|  |  |  |  |
| --- | --- | --- | --- |
| 函数式接口 | 参数类型 | 返回类型 | 用途 |
| Consumer<T>  消费型接口 | T | void | 对类型为T的对象应用操作，包含方法 :  void accept(T t) |
| Supplier<T>  供给型接口 | 无 | T | 返回类型为T的对象，包含方法 :  T get() |
| Function<T, R>  函数型接口 | T | R | 对类型为T的对象应用操作，并返回结果。结果是R类型的对象，包含方法 :  R apply(T t) |
| Predicate<T>  断定型接口 | T | boolean | 确定类型为T的对象是否满足某约束，并返回boolean 值。包含方法 :  boolean test(T t) |

方法引用 − 方法引用提供了非常有用的语法，可以直接引用已有Java类或对象（实例）的方法或构造器。与lambda联合使用，方法引用可以使语言的构造更紧凑简洁，减少冗余代码。

**四、四种方法引用类型**

方法引用的标准形式是：类名::方法名。（**注意：只需要写方法名，不需要写括号**）

有以下四种形式的方法引用：

|  |  |
| --- | --- |
| **类型** | **示例** |
| 引用静态方法 | ContainingClass::staticMethodName |
| 引用某个对象的实例方法 | containingObject::instanceMethodName |
| 引用某个类型的任意对象的实例方法 | ContainingType::methodName |
| 引用构造方法 | ClassName::new |

默认方法 − 默认方法就是一个在接口里面有了一个实现的方法。

新工具 − 新的编译工具，如：Nashorn引擎 jjs、 类依赖分析器jdeps。

Stream API −新添加的Stream API（java.util.stream） 把真正的函数式编程风格引入到Java中。

Date Time API − 加强对日期与时间的处理。

Optional 类 − Optional 类已经成为 Java 8 类库的一部分，用来解决空指针异常。

Nashorn, JavaScript 引擎 − Java 8提供了一个新的Nashorn javascript引擎，它允许我们在JVM上运行特定的javascript应用。

## 设计模式

### 设计模式设计原则

* + 1. 找出应用中可能需要变化之处，把它们独立出来，不要和那些不需要变化的代码混在一起。
    2. 针对接口编程，而不是针对实现编程
    3. 多用组合，少用继承。
    4. 为交互对象之间的松耦合设计而努力