

# 九章算法

## JAVA高级工程师专题课程介绍

主讲人-杨道

我来自你们所说的A厂,在所谓的该枪毙的年龄35+之后用一年时间入职了现在的公司.你们在大厂之路上的辛酸和痛苦我都曾亲身经历.

我做过小公司马仔,也担任过国有银行CTO,与诸位是同路人,从业十余年,从未离开过一线,从服务千万用户的金融系统到服务二十亿用户的互联网系统,软件研发之路上的各个领域都能跟大家做很好的探讨

作为大厂的认证面试官,在实际招聘中见过各型候选人,他们的失败和成功经历都可以给你们最有力的指导.

作为每年最大购物节的高可用保障一号位,我可以给你们带来最新鲜的资讯和最严谨的工程训练

1

求职路径

2

学习路径

3

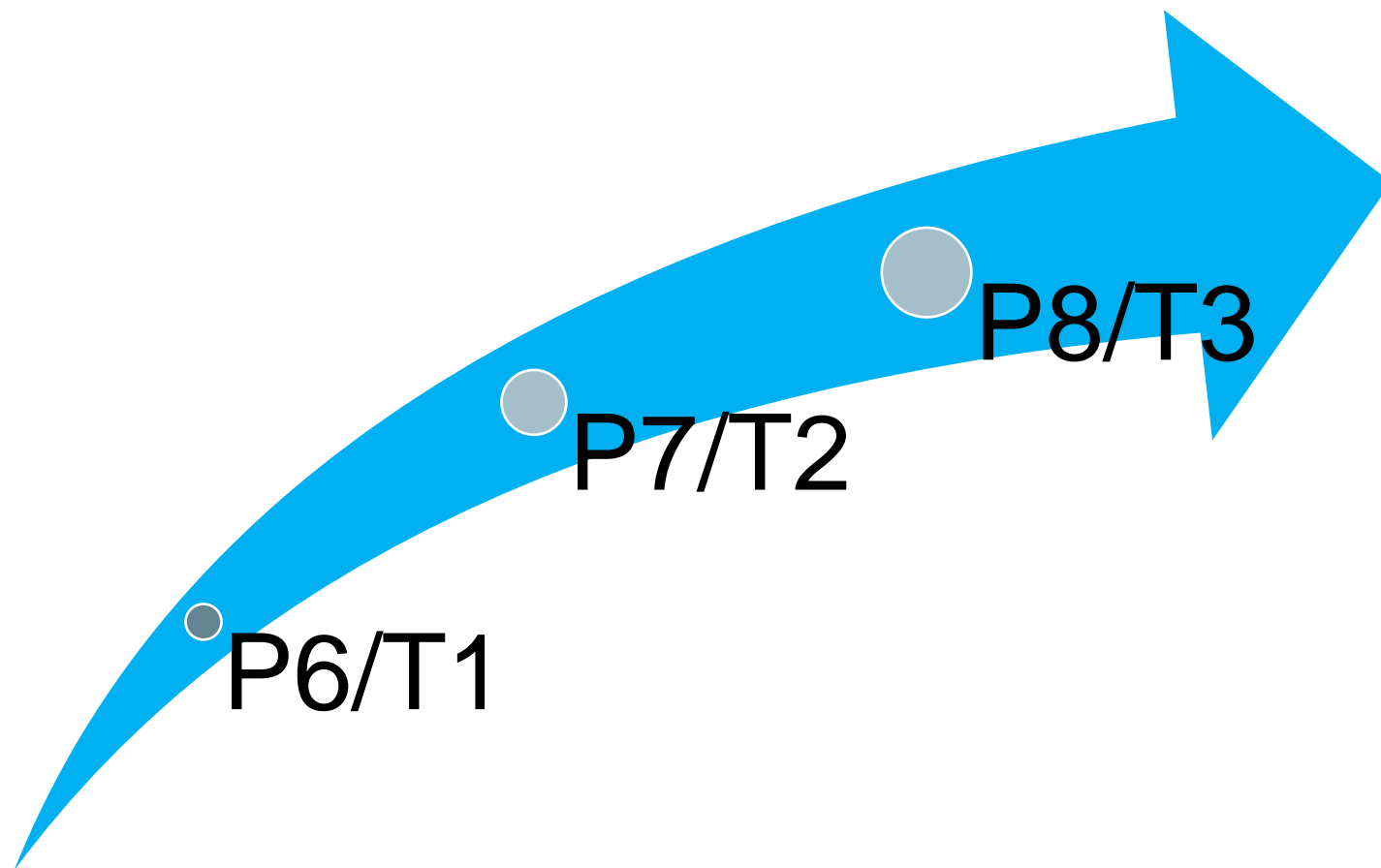
课程安排

4

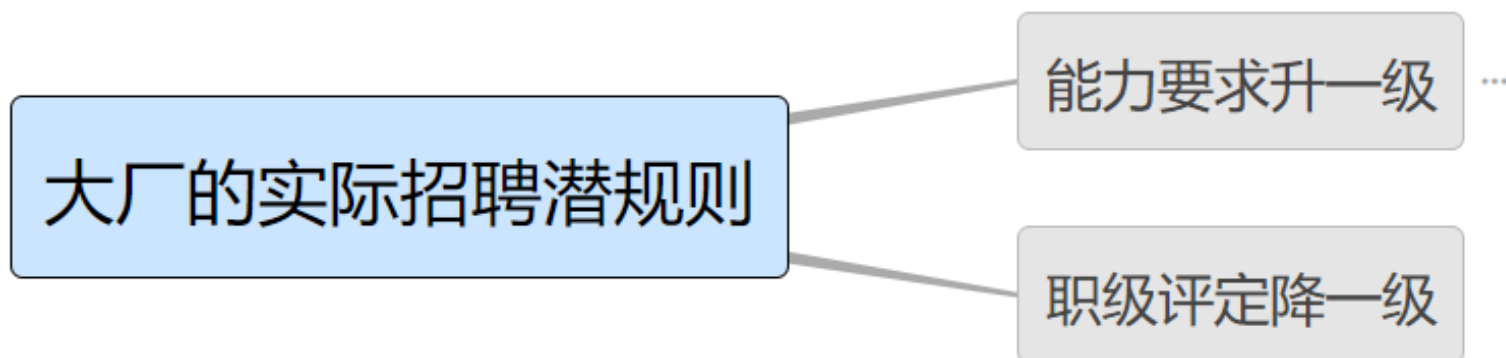
面试题串讲

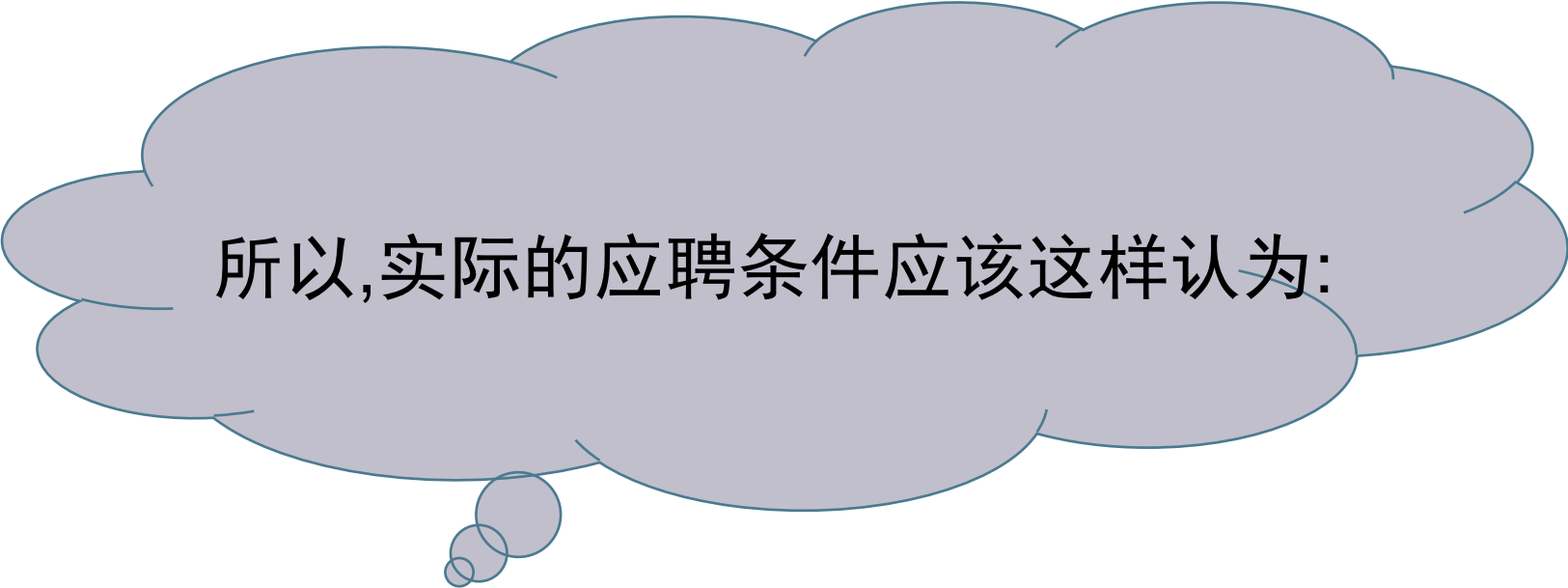
5

总结



职级	薪酬范围	履职要求	入职年龄
P5	20w~40w	开发工程师	22~24岁(校招)
P6-	40w	高级开发工程师	25~28岁
P6	40w~45w	高级开发工程师	25~28岁
P6+	45w~50w	高级开发工程师	25~28岁
P7-	50w~55w (含股票)	技术专家	26~32岁
P7	55w~60w (含股票)	技术专家	26~32岁
P7+	60w~70w (含股票)	技术专家	26~35岁
P8	70w~100w(含股票)	高级专家	30~38岁





所以,实际的应聘条件应该这样认为:

P5

对于技术栈有通透的理解,体系化思维完整,有强大的培养潜力

P6

对行业和技术趋势有着深刻的见解,技术知识熟知原理,能一竿子到底

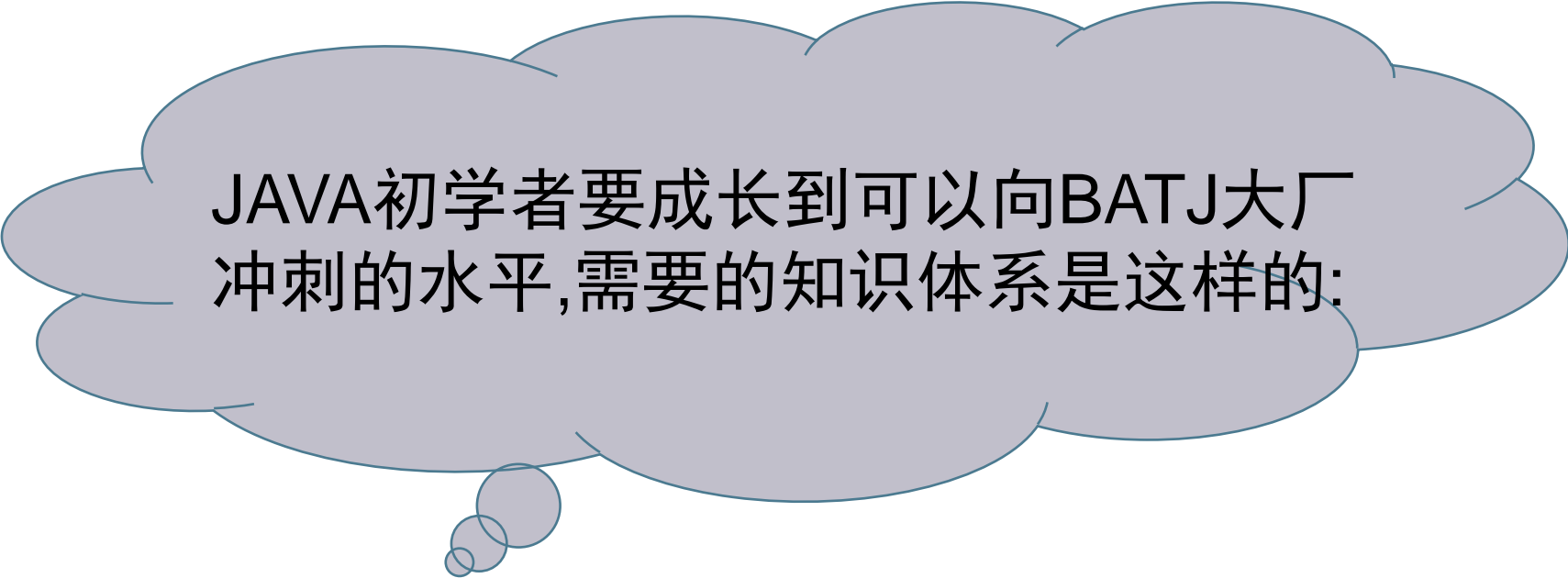
P7

有着独立完整的从业方法论,有良好的从业背景和成功的案例,有0-1,1-N裂变的成功经验分享.有带领中小型团队的成功经验.

P8

在行业内有一定的影响力,有作出决策、实践决策的成功经历,并能批判性的复盘自己的职业经历.能操盘多团队协作,有足够体量背景下的成功经验.



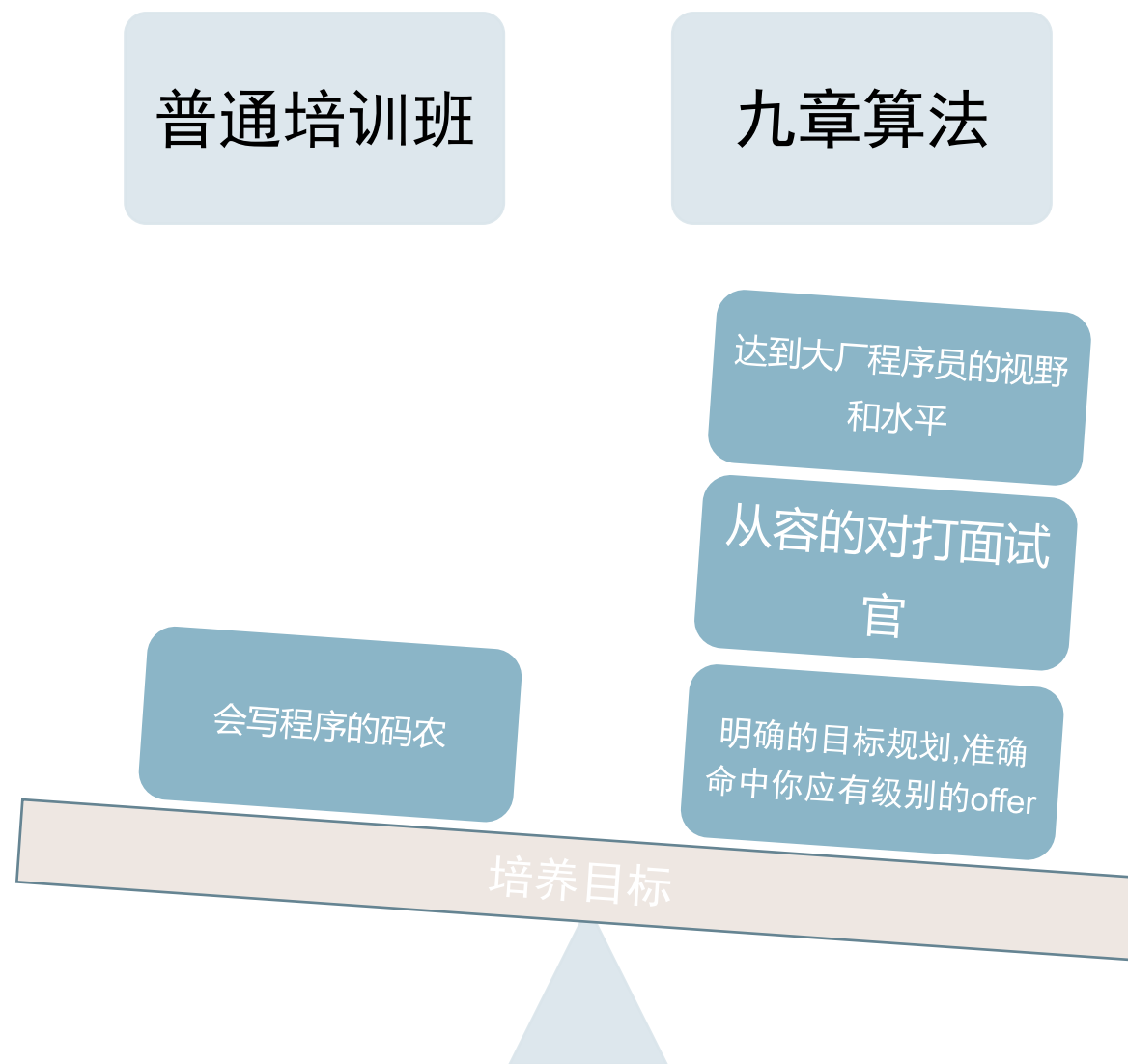


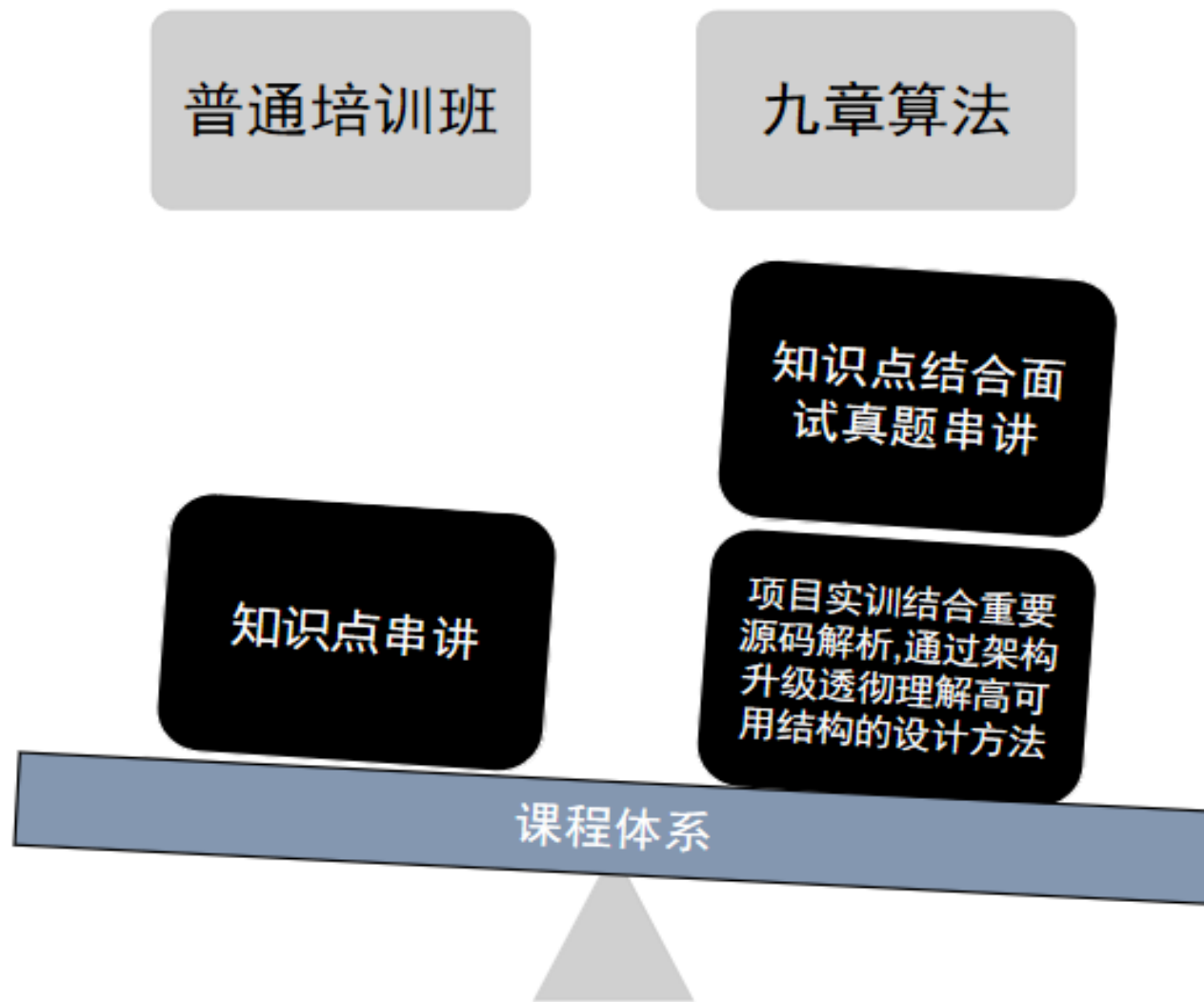
JAVA初学者要成长到可以向BATJ大厂  
冲刺的水平,需要的知识体系是这样的:



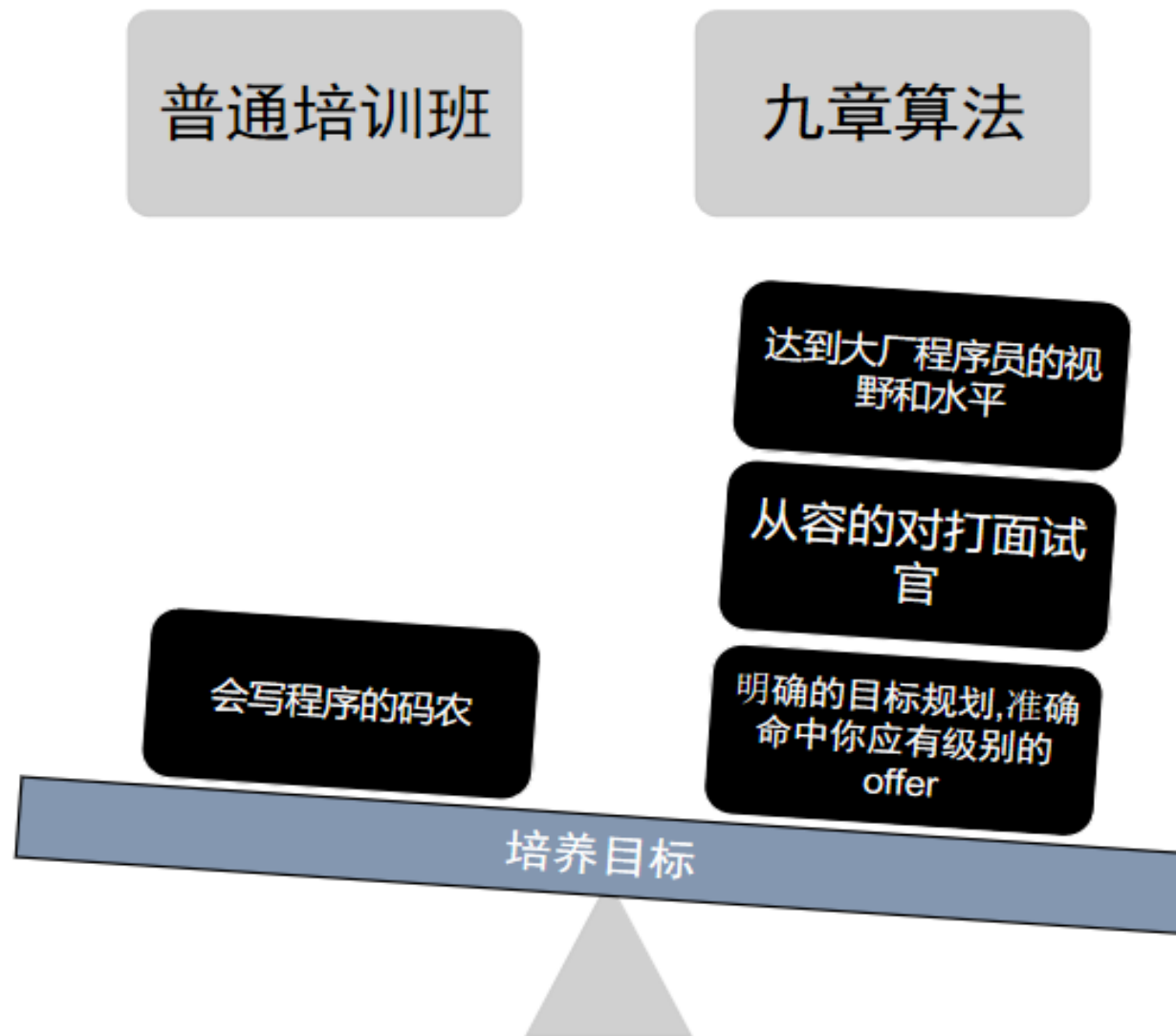


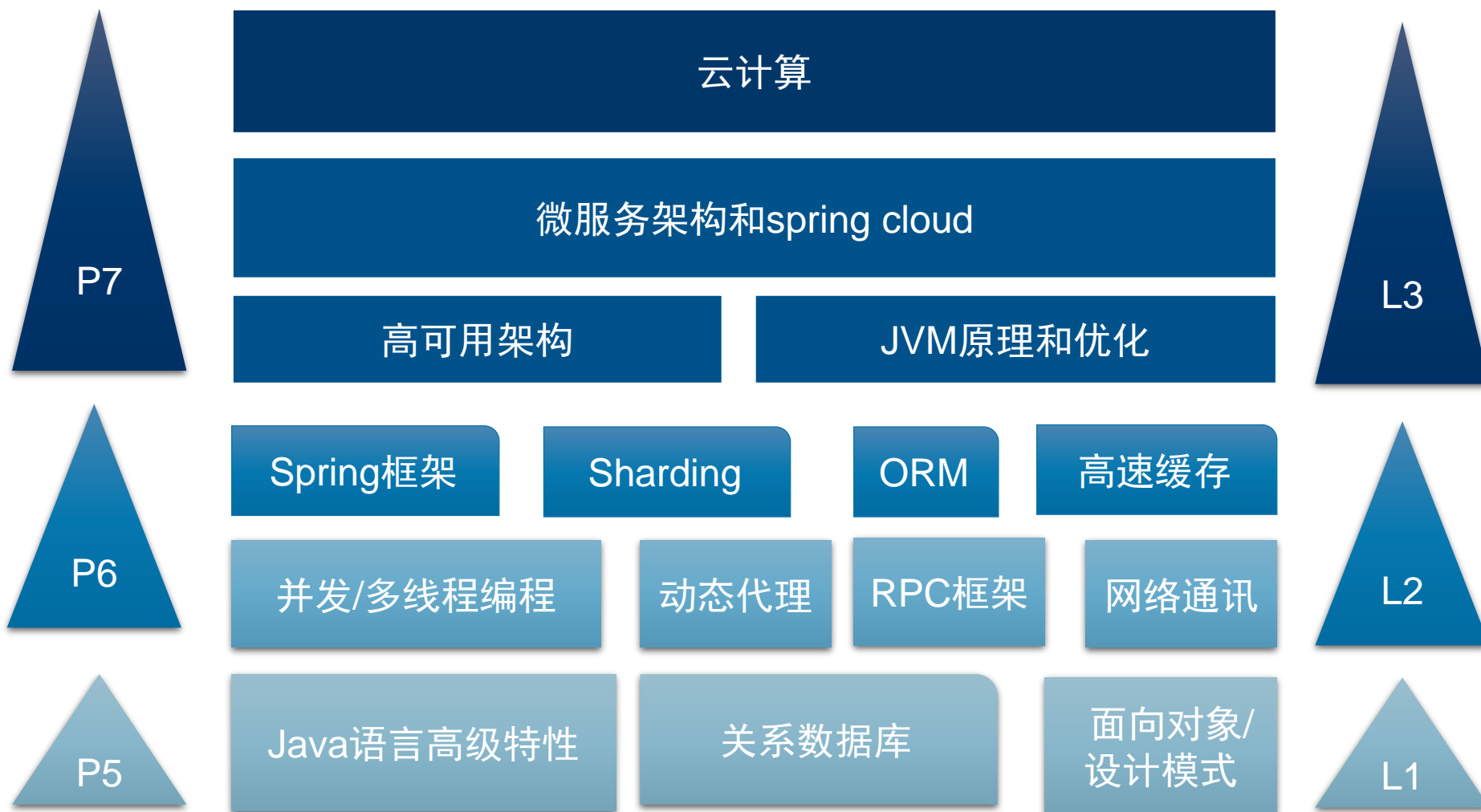






# 课程安排—总体介绍



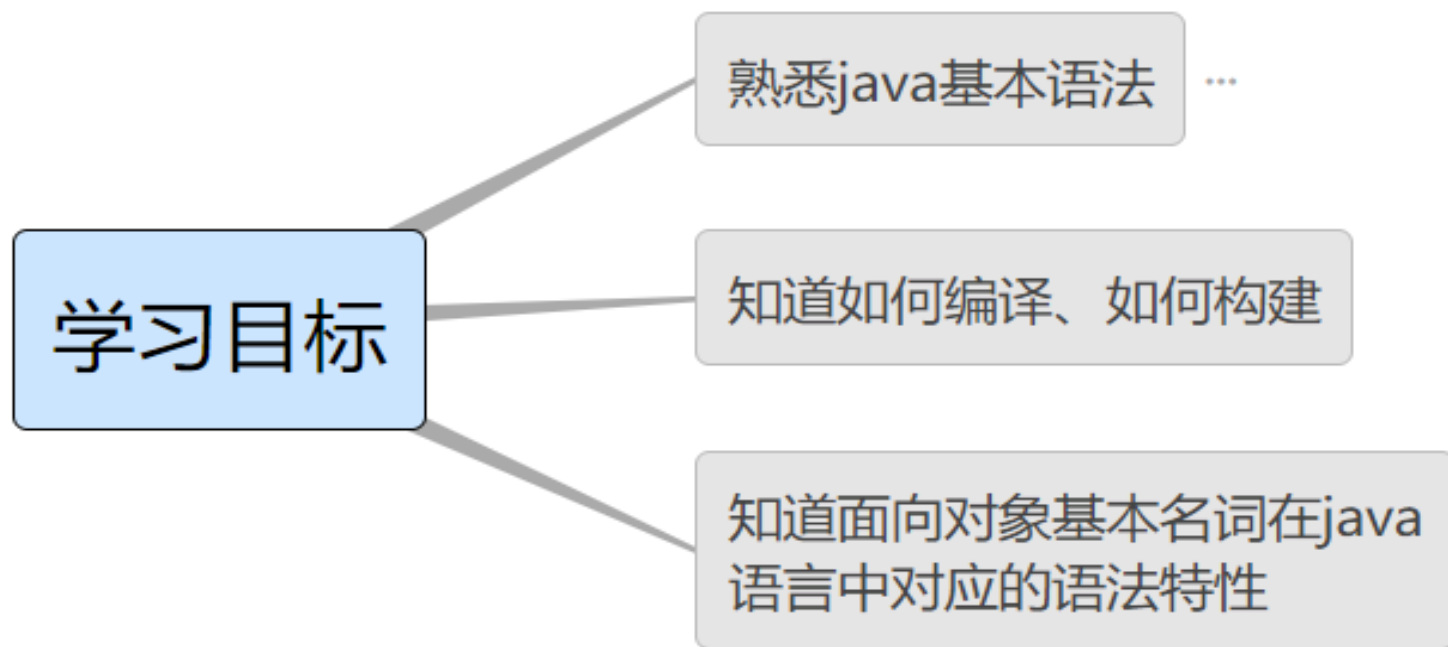


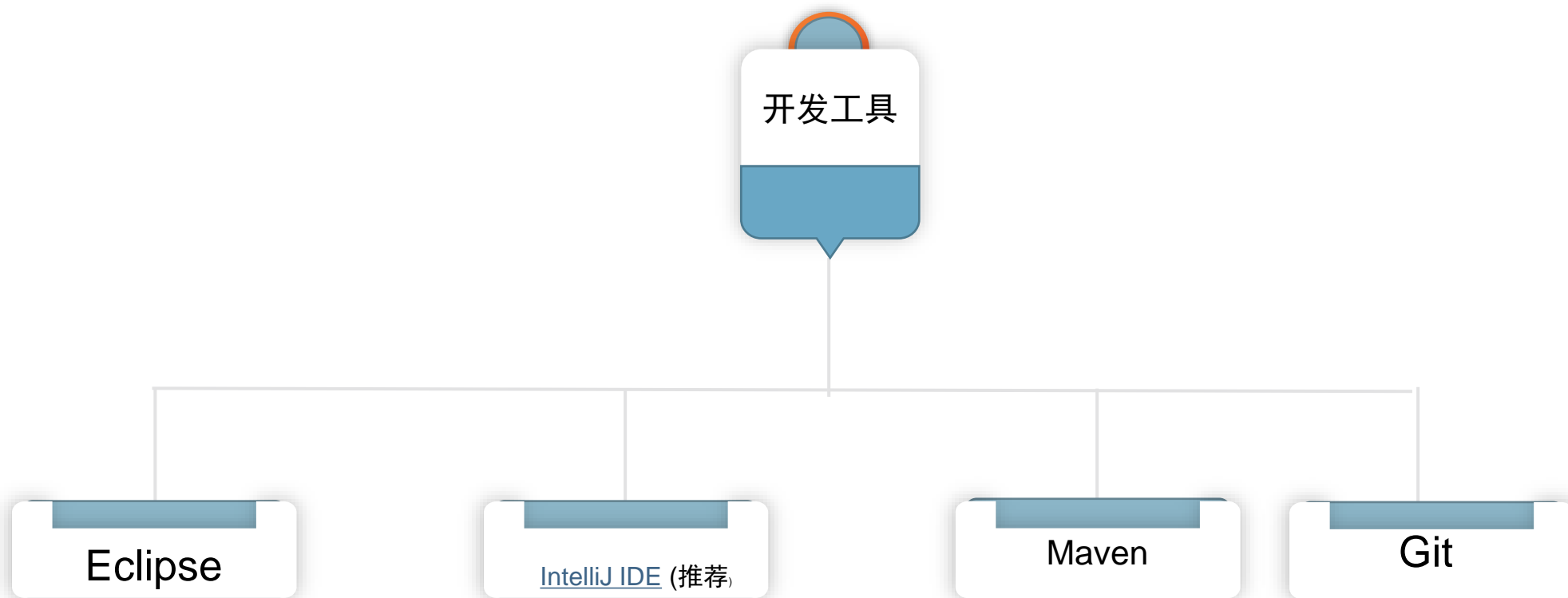




## JAVA基本语法

推荐阅读:<https://www.cnblogs.com/cangqinglang/p/8989986.html>  
java基础知识总结





推荐阅读:

**IntelliJ IDEA 配置开发环境**

~~<https://blog.csdn.net/lduzhenlin/article/details/88883830>~~

<https://www.cnblogs.com/yif0118/p/11367187.html>

## 目标:

下载试用[IntelliJ IDE](#)

知道maven, git是用来做什么的

# 课程安排—先修知识

---

操作系统(linux)

推荐阅读:《Linux常用必会60个命令实例详解》:

<https://blog.csdn.net/ww130929/article/details/69788517>



目标:

- 会使用shell

- 知道进程、线程、管道、信号量的基础知识

## 一阶课程:打造java高级开发者

## 一阶课程的目标 (P5 ~P6-)

高质量Coding能力

算法/数  
据结构  
的正确  
操作

正确高  
效使用  
关系数  
据库

正确的运  
用设计模  
式改善软  
件架构

专注于技术栈的

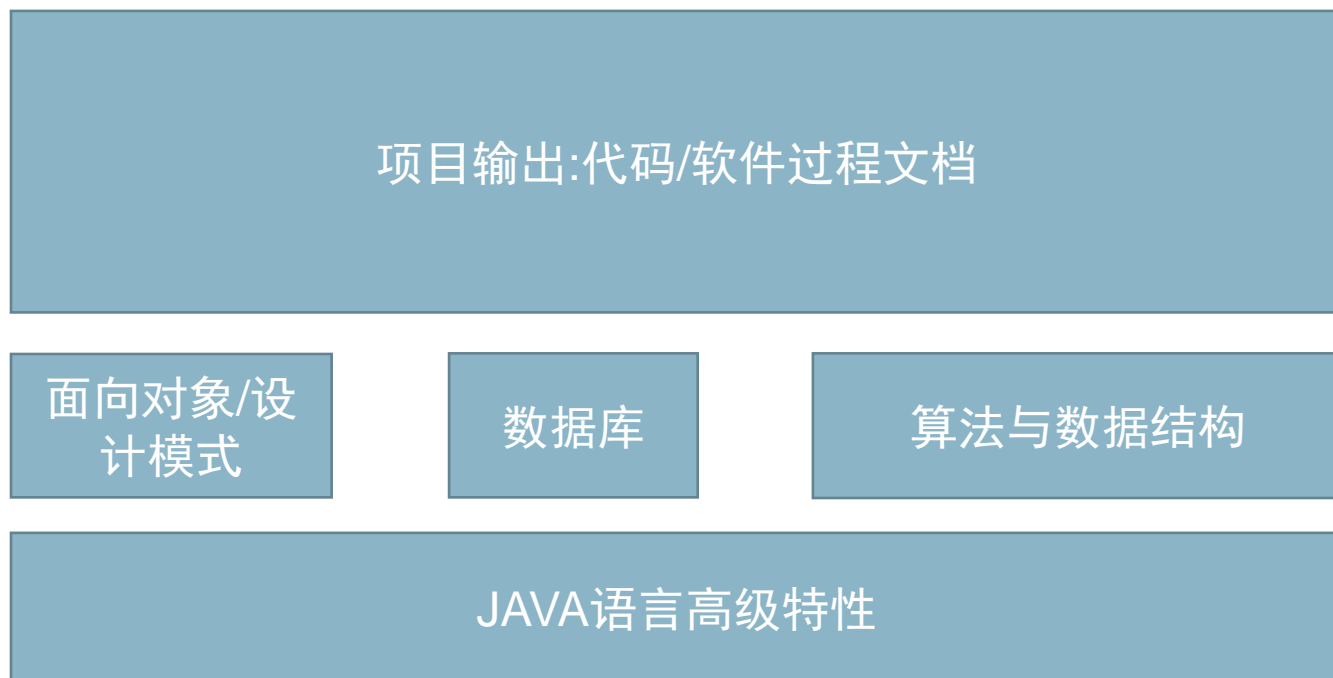


通



透

在一阶课程中,我们从知识升级的视角,帮助你成长为高级的java开发者



## 普通培训班

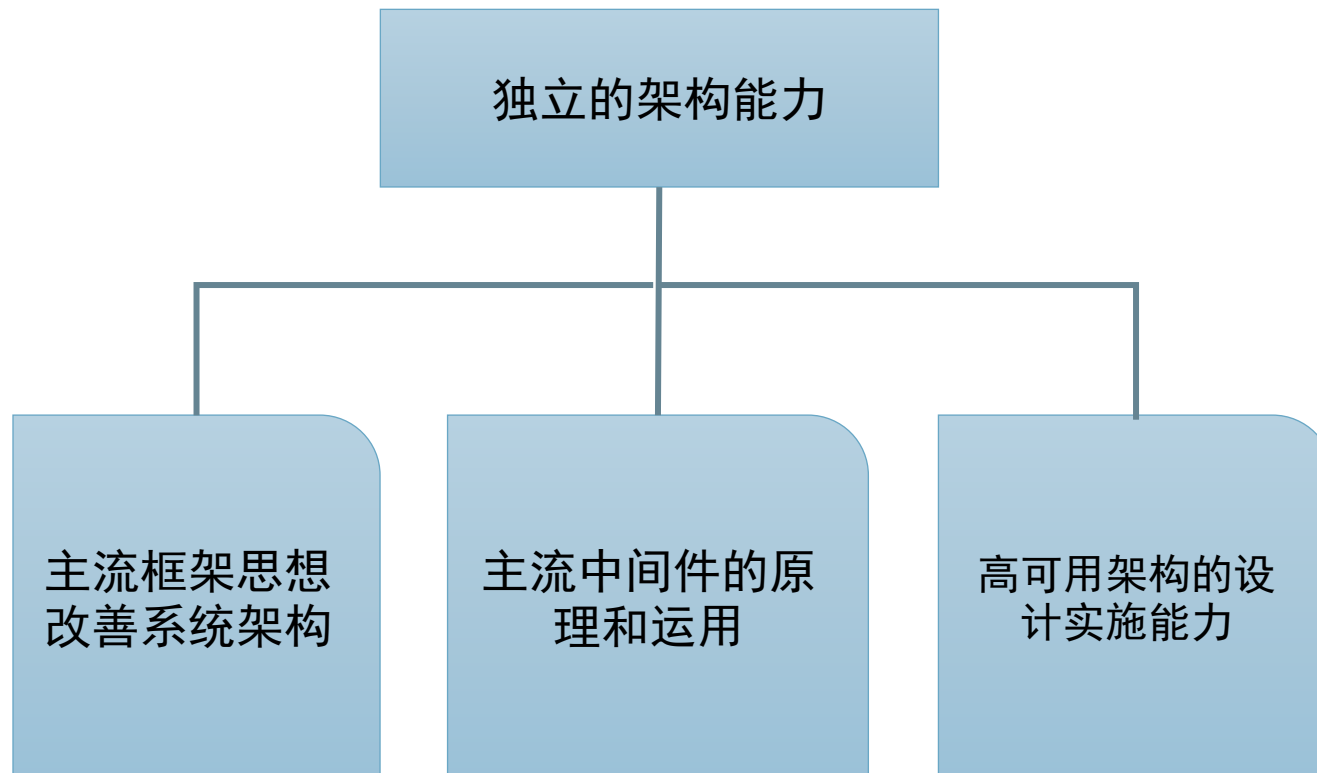
基础知识以讲授知识点为主, 进行简单的实践训练

VS

## 九章算法

对于知识点的讲授深度拉齐大厂要求, 一线大厂高级专家带领下以真实职场需求进行项目输出实践. 算法名师引领, 直接对接海外的技术高度

## 二阶课程的目标(P6- ~P6)



专注于体系化的架构思维

升级

大厂程序员的视野





## 普通培训班

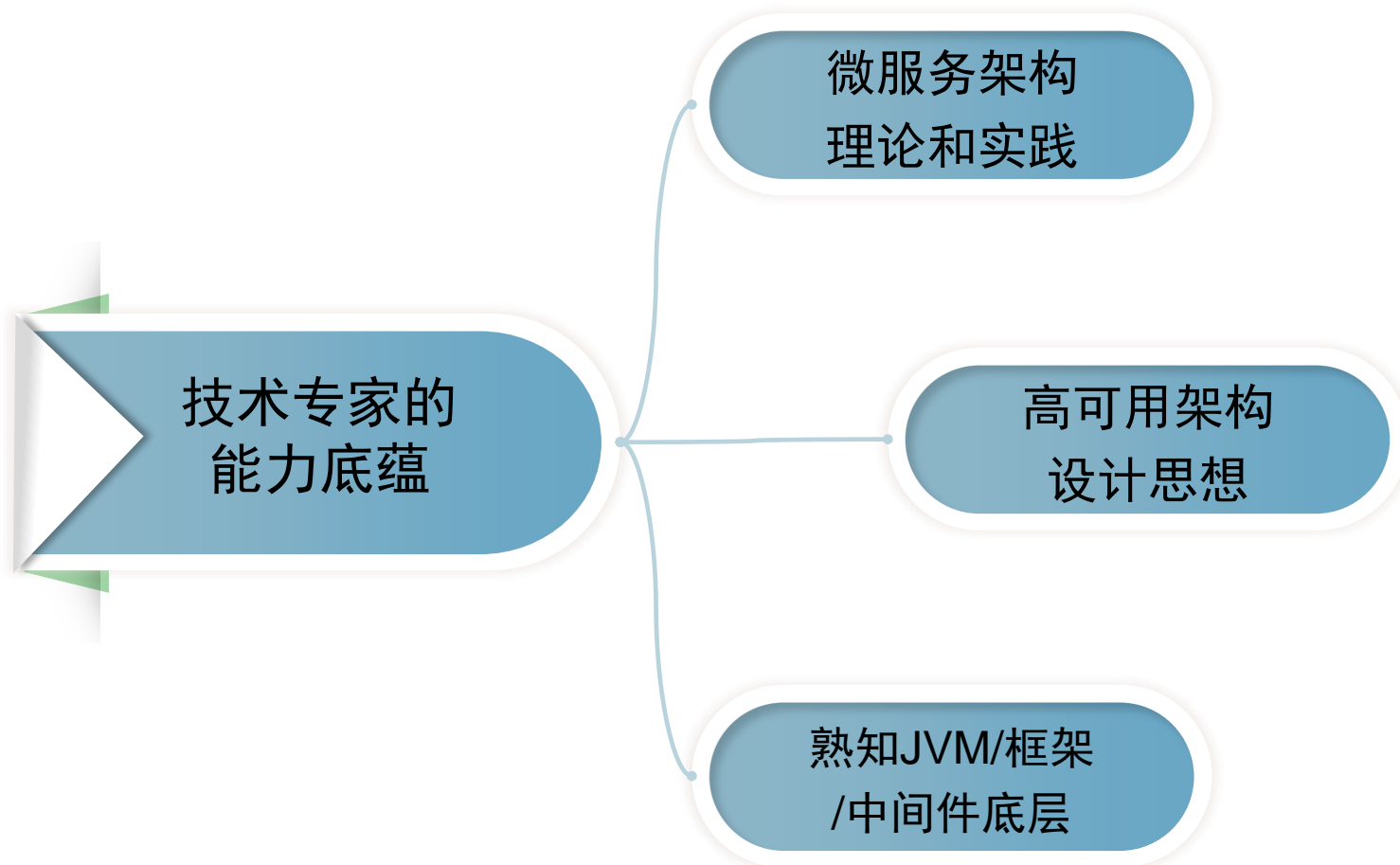
框架讲授以职业技能为主,知道怎么用.  
架构讲授参照公开资料,缺乏实战背景

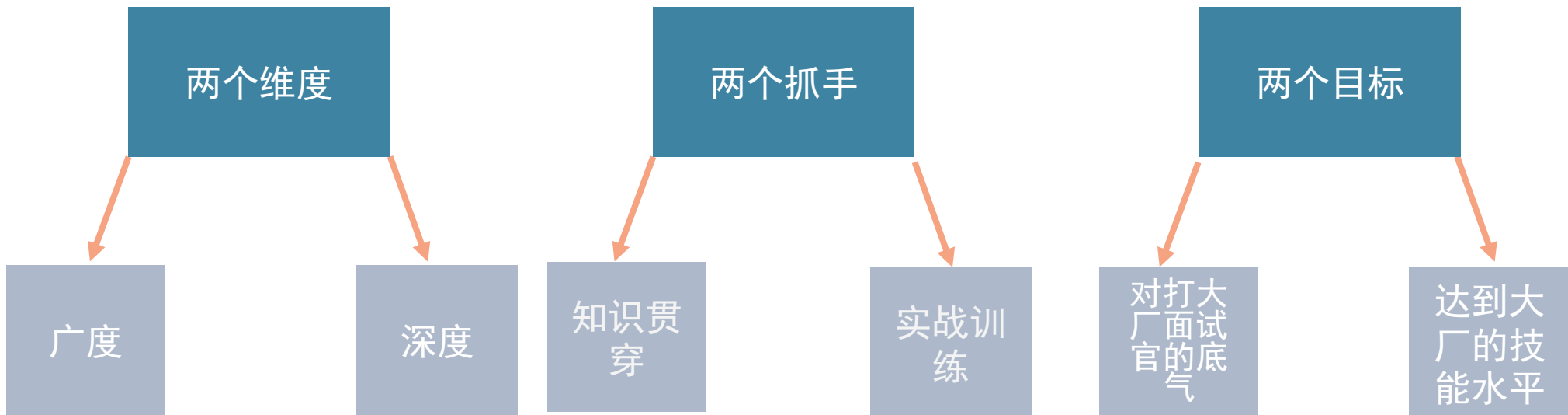
VS

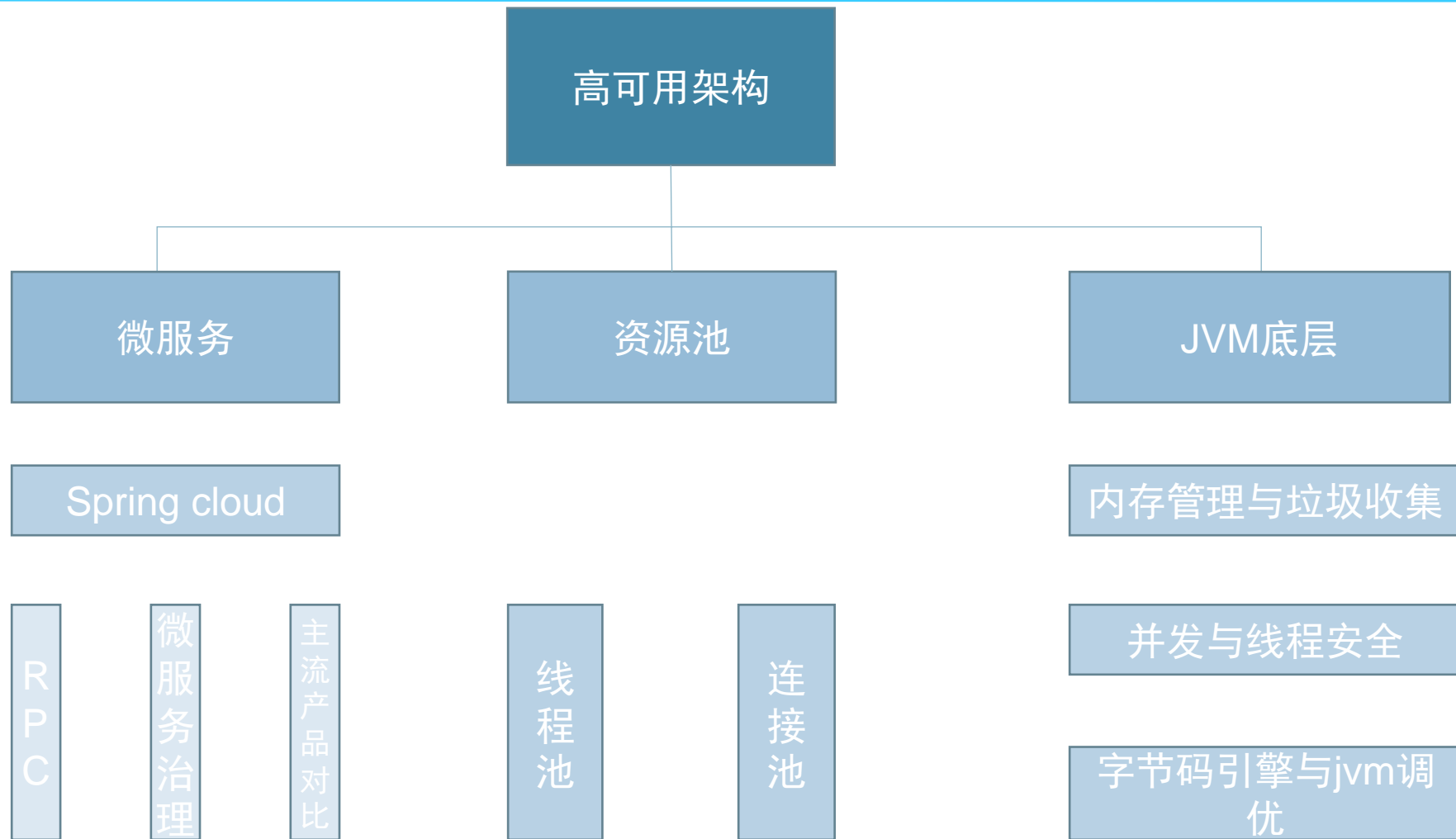
## 九章算法

源码解读、高可用架构实践和大厂面试对焦为教学三板斧,让学员不仅知,而且全方位理解,使用水平和架构思维和大厂程序员拉齐

## 三阶课程的目标(P6+ ~P7)







## 普通培训班

网上公开资料  
为范本,导师的  
实践局限于小  
厂和传统应用,  
没有高并发高  
可用的经历,导  
师没体感,学员  
get不到

VS

## 九章算法

从分布式、微服  
务到云计算逐步  
升级,完全匹配大  
厂的架构升级历  
程,大量的实践案  
例结合底层源码  
和算法分析,专业  
从事高可用架构  
的导师亲自讲授,  
直接对标p7招聘  
要求

一个观点:

争取大厂java板块的职位绝不能停留在程序语言使用和编码上,大厂不需要码农,看重的是用最少的资源实现效益的最大化.

结论:

高质量代码能力是取得大厂offer的钥匙,而这个能力是对于背后的原理有着深刻理解为基础的.

下面通过分享一些基础面试题来看看我们集中学习提升的意义:



**题型:常识背后的原理**

- JDK 和 JRE 有什么区别？

## 普通解答:

具体来说 JDK 其实包含了 JRE, 同时还包含了编译 Java 源码的编译器 Javac, 还包含了很多 Java 程序调试和分析的工具。简单来说: 如果你需要运行 Java 程序, 只需安装 JRE 就可以了, 如果你需要编写 Java 程序, 需要安装 JDK。

## 追杀令:

除了javac你还了解些什么命令行工具, 它们的用途是什么?

-  
答案:

Jcmd: 综合工具

jps: 虚拟机进程状况工具

jinfo

jstat: 虚拟机统计信息监视工具

jinfo: Java配置信息工具

jmap: Java内存映像工具

jhat: 虚拟机堆转储快照分析工具

jstack: Java堆栈跟踪工具

面试官心语:有点意思了哈...

**用过jstat吗?你了解哪些参数?**

## 细节决定成败

选 项	作 用
-class	监视类装载、卸载数量、总空间以及类装载所耗费的时间
-gc	监视 Java 堆状况，包括 Eden 区、两个 survivor 区、老年代、永久代等的容量、已用空间、GC 时间合计等信息
-gccapacity	监视内容与 -gc 基本相同，但输出主要关注 Java 堆各个区域使用到的最大、最小空间
-gcutil	监视内容与 -gc 基本相同，但输出主要关注已使用空间占总空间的百分比
-gccause	与 -gcutil 功能一样，但是会额外输出导致上一次 GC 产生的原因
-gcnew	监视新生代 GC 状况
-gcnewcapacity	监视内容与 -gcnew 基本相同，输出主要关注使用到的最大、最小空间
-gcold	监视老年代 GC 状况
-gcoldcapacity	监视内容与 -gcold 基本相同，输出主要关注使用到的最大、最小空间
-gcpermcapacity	输出永久代使用到的最大、最小空间
-compiler	输出 JIT 编译器编译过的方法、耗时等信息
-printcompilation	输出已经被 JIT 编译的方法

Class文件结构  
类加载的时机  
类加载过程

选项	作用
-class	监视类装载、卸载数量、总空间以及类装载所耗费的时间
-gc	监视 Java 堆状况，包括 Eden 区、两个 survivor 区、老年代、永久代等的容量、已用空间、GC 时间合计等信息
-gccapacity	监视内容与 -gc 基本相同，但输出主要关注 Java 堆各个区域使用到的最大、最小空间
-gcutil	监视内容与 -gc 基本相同，但输出主要关注已使用空间占总空间的百分比
-gccause	与 -gcutil 功能一样，但是会额外输出导致上一次 GC 产生的原因
-gcnew	监视新生代 GC 状况
-gcnewcapacity	监视内容与 -gcnew 基本相同，输出主要关注使用到的最大、最小空间
-gcold	监视老年代 GC 状况
-gcoldcapacity	监视内容与 -gcold 基本相同，输出主要关注使用到的最大、最小空间
-gcpermcapacity	输出永久代使用到的最大、最小空间
-compiler	输出 JIT 编译器编译过的方法、耗时等信息
-printcompilation	输出已经被 JIT 编译的方法

问题:Class的文件结构是什么:



中文描述	数据类型	名称	数量
魔数	u4	magic	1
次版本号	u2	minor_version	1
主版本号	u2	major_version	1
常量池计数器	u2	constant_pool_count	1
常量池	cp_info	constant_pool	constant_pool_count-1
访问标志	u2	access_flags	1
类索引	u2	this_class	1
父类索引	u2	super_class	1
接口计数器	u2	interfaces_count	1
接口索引集合	u2	interfaces	interfaces_count
字段计数器	u2	fields_count	1
字段表集合	field_info	fields	fields_count
方法计数器	u2	methods_count	1
方法表集合	method_info	methods	methods_count
属性计数器	u2	attributes_count	1
属性表集合	attribute_info	attributes	attributes_count

详细解释参考: [https://blog.csdn.net/A\\_zhenzhen/article/details/77977345?depth\\_1-utm\\_source=distribute.pc\\_relevant.none-task&utm\\_source=distribute.pc\\_relevant.none-task](https://blog.csdn.net/A_zhenzhen/article/details/77977345?depth_1-utm_source=distribute.pc_relevant.none-task&utm_source=distribute.pc_relevant.none-task)

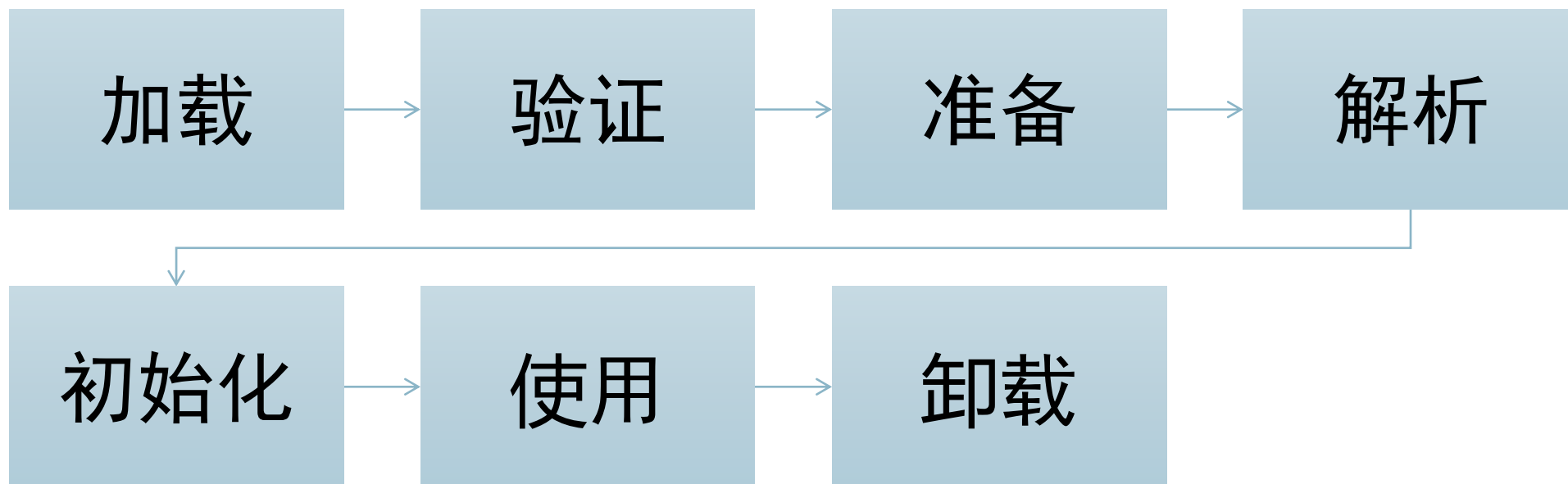
追问一:

Class加载的时机?

## 要点:

- 启动类加载器，根装载器，用户不可访问，Bootstrap ClassLoader，加载JAVA\_HOME\lib，或者被-Xbootclasspath参数限定的类
- 扩展类加载器，Extension ClassLoader，加载\lib\ext，或者被java.ext.dirs系统变量指定的类
- 应用程序类加载器，Application ClassLoader，加载ClassPath中的类库
- 自定义类加载器，通过继承ClassLoader实现，一般是加载我们的自定义类

追问二:Class加载的过程?



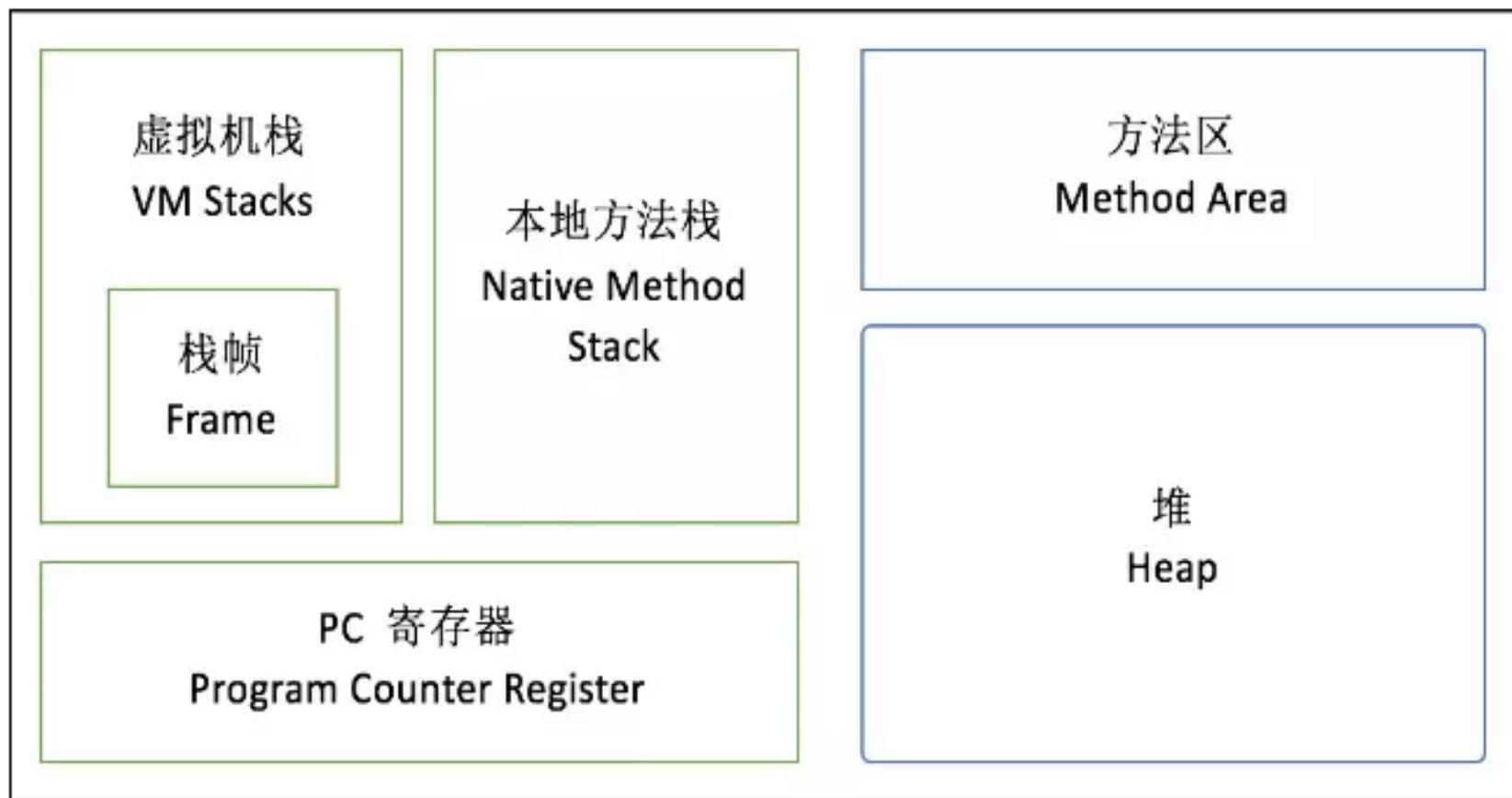
梳理背后庞大的知识栈

Java虚拟机栈  
垃圾收集算法

选 项	作 用
-class	监视类装载、卸载数量、总空间以及类装载所耗费的时间
-gc	监视 Java 堆状况，包括 Eden 区、两个 survivor 区、老年代、永久代等的容量、已用空间、GC 时间合计等信息
-gccapacity	监视内容与 -gc 基本相同，但输出主要关注 Java 堆各个区域使用到的最大、最小空间
-gcutil	监视内容与 -gc 基本相同，但输出主要关注已使用空间占总空间的百分比
-gccause	与 -gcutil 功能一样，但是会额外输出导致上一次 GC 产生的原因
-gcnew	监视新生代 GC 状况
-gcnewcapacity	监视内容与 -gcnew 基本相同，输出主要关注使用到的最大、最小空间
-gcold	监视老年代 GC 状况
-gcoldcapacity	监视内容与 -gcold 基本相同，输出主要关注使用到的最大、最小空间
-gcpermcapacity	输出永久代使用到的最大、最小空间
-compiler	输出 JIT 编译器编译过的方法、耗时等信息
-printcompilation	输出已经被 JIT 编译的方法

追问一:谈谈你对Java虚拟机栈的了解





追问二:Jvm的垃圾回收算法有哪些?

## 标记-清除、复制、标记-整理和分代收集

JVM根据对象在内存中存活时间的长短，把堆内存分为新生代（包括一个Eden区、两个Survivor区）和老年代（Tenured或Old）。Perm代（永久代，Java 8开始被“元空间”取代）属于方法区了，而且仅在Full GC时被回收。

这就是大厂面试java程序员的基本战略:

从常识开始,逐渐诱导,考察原理理解的完整度和深度

## 考察思路:

- 1、简单的话题引入,麻痹候选人
- 2、找一个角度/工具让候选人去匹配场景
- 3、深挖背后的原理

**题型:熟悉的陌生人**

== 和 equals 的区别是什么？

答案要点:

两个方法都是比较等价性,它们的区别要从“基本类型”和“引用类型”来看,作为基本类型来说,它们比较的都是值,作为引用类型——对象来说,它们比较的是双方是否是同一个对象.



考察点:

基本类型&引用类型

基本类型：比较的是值是否相同

引用类型：比较的是引用是否相同

equals()的原生实现可以直接的定义它们之间的关系:

```
public boolean equals(Object obj) {  
    return (this == obj);  
}
```

加分Tips:

当然,我们可以通过重载Object中的equals方法来实现自定义的等价计算方法,比如,在String类型中,由于源码中重新实现了该方法,所以会产生下面的结果:

```
String.java x
965 * object.
966 *
967 * @param anObject
968 *     The object to compare this {@code String} against
969 *
970 * @return {@code true} if the given object represents a {@code String}
971 *         equivalent to this string, {@code false} otherwise
972 *
973 * @see #compareTo(String)
974 * @see #equalsIgnoreCase(String)
975 */
976 @ public boolean equals(Object anObject) {
977     if (this == anObject) {
978         return true;
979     }
980     if (anObject instanceof String) {
981         String anotherString = (String)anObject;
982         int n = value.length;
983         if (n == anotherString.value.length) {
984             char v1[] = value;
985             char v2[] = anotherString.value;
986             int i = 0;
987             while (n-- != 0) {
988                 if (v1[i] != v2[i])
989                     return false;
990                 i++;
991             }
992             return true;
993         }
994     }
995     return false;
996 }
```

Tips:用字符串常量实例化的String的内容其实是保留在常量区

```
String x = "string";  
String y = "string";
```

`x==y` ——> true

引用的都是常量区的同一个数据项,所以有上面的结果

考察点:

equals()的概念

实际要求:

平时对源码的深挖意识即技术钻研和批判性思维

上面这些题目有这样的特点:

- 1、都是教科书级的题目
- 2、都是工程实践中常用的
- 3、都是习以为常,不容易去主动深究的

考察目的:

- 1、基础知识的扎实程度
- 2、候选人对技术的热情

下面在看一些例子

Java 中操作字符串都有哪些类？它们之间有什么区别？



答案:String, StringBuffer, StringBuilder

区别:String 声明的是不可变的对象,每次操作必然产生一个新的对象  
StringBuffer和StringBuilder都继承自抽象类AbstractStringBuilder  
StringBuffer具备线程安全性  
在使用场景上,并发必选 StringBuffer,迭代必选 StringBuilder  
普通场景选String,避免中途不必要的类型转换开销

答案组织策略:

知道有什么,知道为什么,知道怎么用

加分Tips:

StringBuffer和StringBuilder都继承自抽象类AbstractStringBuilder

```
/*  
 * @author      Michael McCloskey  
 * @see         java.lang.StringBuffer  
 * @see         java.lang.String  
 * @since       1.5  
 */  
public final class StringBuilder  
    extends AbstractStringBuilder  
    implements java.io.Serializable, CharSequence  
{
```

```
 * It performs no synchronization.  
 */  
 * @author      Arthur van Hoff  
 * @see         java.lang.StringBuilder  
 * @see         java.lang.String  
 * @since       JDK1.0  
 */  
public final class StringBuffer  
    extends AbstractStringBuilder  
    implements java.io.Serializable, CharSequence  
{
```

String的源码解读也可以作为加分点:

```
public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence {
    /** The value is used for character storage. */
    private final char value[];
```

谈资: final 修饰—>不可变——>每次操作都会生成新的 String 对象  
对比StringBuffer和StringBuilder ——>值可变、拼接字符串开销

## 拷问线程安全性

查源码,找synchronized、线程锁  
得到结论:StringBuffer具备线程安全性

下面看一段StringBuffer的源码:

```
@Override
public synchronized int length() { return count; }

@Override
public synchronized int capacity() { return value.length; }

@Override
public synchronized void ensureCapacity(int minimumCapacity) { super.ensureCapacity(minimumCapacity); }

/**
 * @since 1.5
 */
@Override
public synchronized void trimToSize() { super.trimToSize(); }

/**
 * @throws IndexOutOfBoundsException {@inheritDoc}
 * @see #length()
 */
@Override
public synchronized void setLength(int newLength) {
    toStringCache = null;
    super.setLength(newLength);
}
```

批判性思考:

StringBuffer具备线程安全性,迭代开销又小,那我们在工程实践中首选它不就行了吗?



举一反三:

HashMap 、 Hashtable 、 TreeMap有什么区别?

Hashtable、HashMap、TreeMap都是最常见的一些Map实现，是以键值对的形式存储和操作数据的容器类型。

Hashtable是Java类库提供的一个哈希实现，本身是同步的，不支持null键和null值，由于同步导致性能开销，所以已经很少被推荐使用。

HashMap是应用更加广泛的哈希表实现，行为上大致与HashTable一致，主要区别在于HashMap不是同步的，支持null键和null值等。通常情况下HashMap进行get和put操作可以达到常数时间的性能，所以它是绝大部分利用键值对存取场景的首选。

TreeMap则是基于红黑树的一种提供顺序访问的Map，它的get、put、remove之类的操作都是 $O(\log n)$ 的时间复杂度，具体顺序可以由指定的Comparator来决定，或者根据键的自然顺序来判断。

若在单线程中，我们往往会选择HashMap；

而在多线程中，则会选择Hashtable。

若不能插入null元素，则选择Hashtable；

面对BATJ大厂越来越苛刻的招聘要求,我们的打法是:

- 技术栈的通透讲授
- 源码的分析导读
- 贴近真实的工程实践
- 大厂思维的架构升级

我们对教学成果的要求:

- 不仅把学员送进大厂
- 还要让学员在大厂中站稳脚跟

- 我们培养的目标是懂原理、知源码、会刷题、能搭建高可用架构的高级工程师/架构师
- 我们的学员对标薪资是年薪 30W~100W

24小时内报名,即可获得团购优惠

加班主任微信获取优惠码





# 感谢聆听 Thanks