1.

jdk jre jvm

jvm 运行环境

jre 核心类

jdk 编译工具 打包工具

2:oracle jdk -open jdk

oracle 更稳定

3:修饰符

同包 子类

def 不支持子类 pro支持

4:final

类 不可继承

方法 不可override

变量 常或者引用 不可改变

5:static

静态内部类

静态成员

静态代码块

6:

封装 表示封闭

继承 表示可以重用

多态 具体类型和发出的方法调用不确定;

向上转型 重写 继承

7：

单一原则

开闭原则

里shi替换 子类可以出现在父类的地方；

依赖倒置：高层依赖抽象

接口分离：使用多个接口

8:抽象类和接口

抽象类是类，可以有成员变量，使用的时候要继承

接口可以多继承，里面的成员是static final修饰;

都不能实例化，需要具体的实现了所有抽象方法的类实例才行

9:作用域

静态：类

成员：类的实例

局部：方法

成员有默认初始，局部没有，必须赋值，

10：

如果父类没有构造 默认无参构造

否则，没有无参

子类默认会调用父类无参 此时会报错 所以

11：内部类

静态内部 成员内部 匿名内部 方法内部

12：内部类加final

为了一致性

13:

构造器不重写，只重载

14:hashcode equals ==

==只判断地址；

新创建的对象， 在符合条件时候 equal

此时 要重写hashcode 满足 equal必然hahscode相同

15:BIO AIO NIO

BIO是阻塞的 底层是linux 的阻塞io

要么一直轮询直到OS准备好了数据

NIO 有select poll epoll三种模型

select poll每次都需要把关注的fd传到内核， 内核处理完再返回来，

epoll则是注册到内核的红黑树上， fd发生了读写时间 会注册到事件列表上， 然后进程查看事件列表就可以处理，极大提高的效率;

事件驱动 对应java是 reactor响应式，支持事件处理;

chinal

16:

反射

获得类的属性和方法

在框架设计里面大量使用

比如spring的本质就是beandifination， 通过反射的方式创建类的实例

17：

字符串常量池；

string a=new String("abc")创建两个对象

常量池一个

a一个；

String可以优化hashmap 因为 会缓存下来；

18:

string:不可变

stringbuffer 线程安全

stringbuilder不安全； 效率高

节省空间， 如果可变的话就没法用缓冲池了

19:

a=new Integer(12)

b=12;

a!=b;

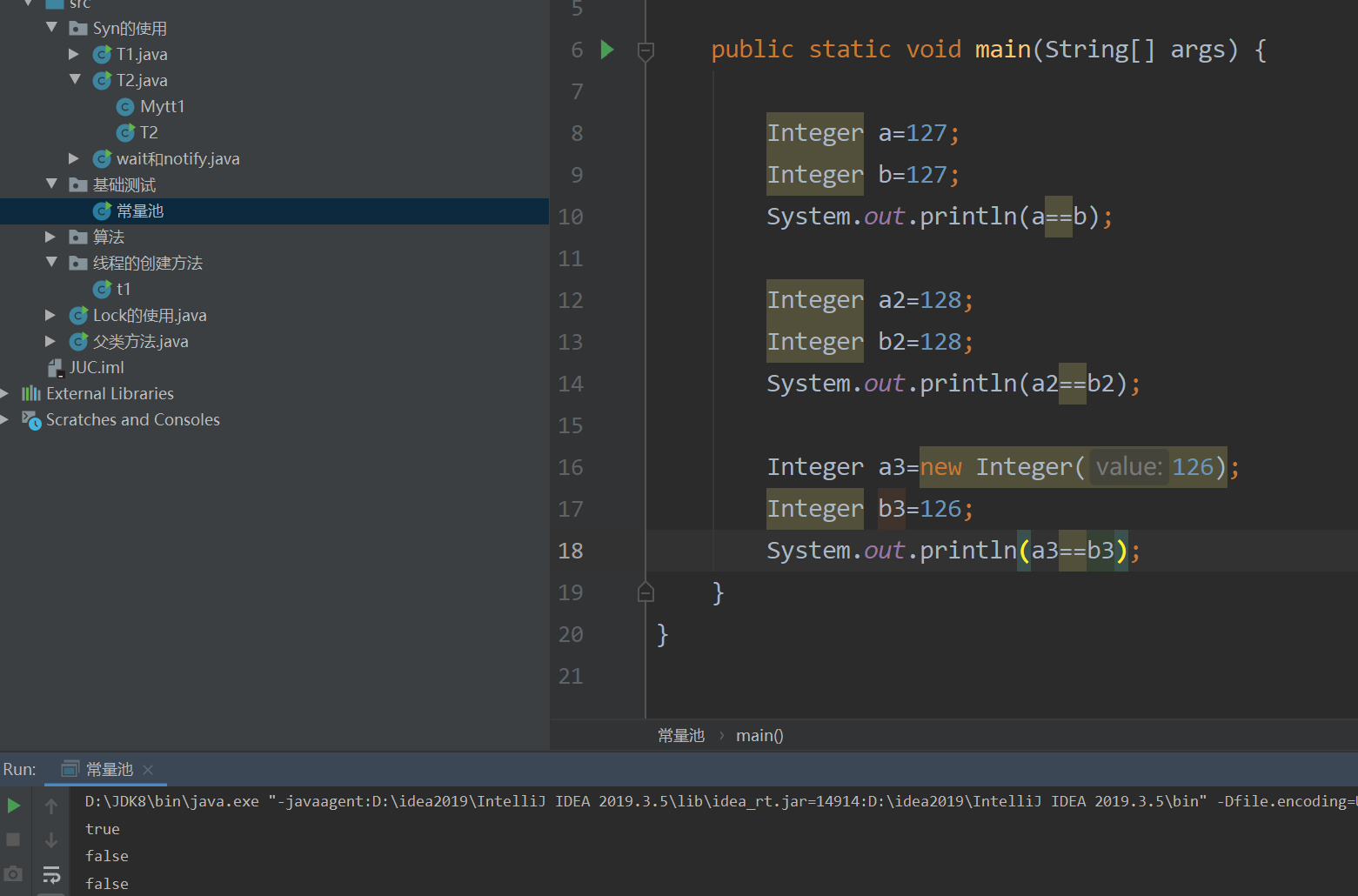
a=12;

b=12;

则a==b;

a=128

b=128

a!=b;

原因就是 创建包装类的时候 如果数值小 直接读缓存；

Integer.parseOf也是一样的道理;

20:

集合Collection

List Queue Set

Map

自己单独玩儿哈

21:哪些是线程安全的

concurenthashmap

hashtable

22:failfast

在iterator的时候维持一个modcount每次都检测；

failsafe

就是copyonwritelist

每一次put都会复制一份， iterator 使用的就是开始的元素；

iterator只是复制了一个快照；

23：

arraylist

加transient 只序列化存放的数据 ，减小体积；

24：

hashmap的实现流程：

hash

key.hashcode 和hashcode>>>16位异或，高16位0

&（length-1）

调用key的hashCode方法计算哈希值，并据此计算出数组下标index

如果发现当前的桶数组为null，则调用resize()方法进行初始化

如果没有发生哈希碰撞，则直接放到对应的桶中

如果发生哈希碰撞，且节点已经存在，就替换掉相应的value

如果发生哈希碰撞，且桶中存放的是树状结构，则挂载到树上

如果碰撞后为链表，添加到链表尾，如果链表长度超过TREEIFY\_THRESHOLD默认是8，则将链表转换为树结构

数据put完成后，如果HashMap的总数超过threshold就要resize

1：首先计算hashkey高16位>>>16 与低16做一个异或；

相当于二次hash 目的是 增强hash的分布；

2: 获得桶下标&2^n-1;

3：如果为空 resize

4:如果没有节点 直接插入

5：如果有节点， 不管是链表还是树 先找一遍是否存在，存在就替换 不存在就插入

6：对于链表 插入了以后 判断够不够8个 扩容

7：判断此时 是否达到了扩容因子，的数目 达到扩容， 需要改变的要变化；

相比于hashtable hashmap支持null

1.7使用头插法 会出现死链的问题

7 8都会出现数据错乱的情况

25: concurentHashmap

1.7使用segment，

1.8使用 node【数组】，cas sysn如果存在元素 就在队首加一个锁, 超过的话还是会树化的；

26: ZeroCOpy

mmap 做了映射

同时网卡可以收到缓冲区数据的变化，直接发送，没有拷贝事件

外来数据 到了网卡 dma到kernel 然后拷贝到内存

内存拷贝到kernel 在dma到网卡；

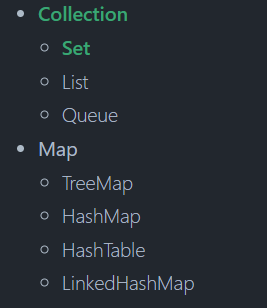
0拷贝mmap做了一个映射， 网卡

27：元注解：

Target 注解作用的类型？

Retention 生命周期编译 test 运行

28:常用的数据结构



Collection:

set: hashset

treeset 可以范围查找 查找慢一点

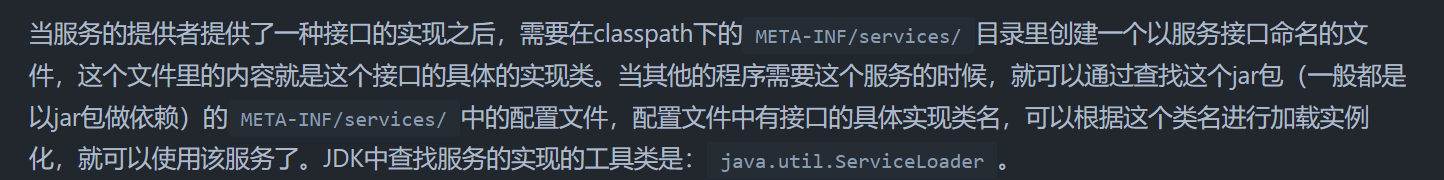
List: array 和linked

Queue 使用LinkedList

Map

treeMap hashMap

29:SPI原理：



30:

日志的级别：

DEBUG < INFO < WARN < ERROR