## String.format

描述:格式化；

举例: **%[argument\_index$][flags][****width][.precision]****conversion**

**argument\_index:参数位置；**

**flags：- 左对齐；**

**width：字符宽度；**

**precision：保留字符数**

**conversion:类型，如果为d ，则flags可以用0 填充，默认都是空格**

String s= String.format("my name is %s,age is %s","zs","30");

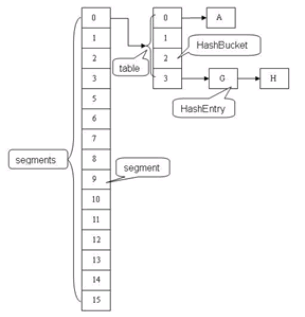
String e= String.format("my name is %1$.3s,age is %2$s","zsddd","30")-🡪

my name is zsd,age is 30

## ConcurrentHashMap

描述:

ConcurrentHashMap是由Segment数组结构和HashEntry数组结构组成。Segment是一种可重入锁ReentrantLock，在ConcurrentHashMap里扮演锁的角色，HashEntry则用于存储键值对数据。一个ConcurrentHashMap里包含一个Segment数组，Segment的结构和HashMap类似，是一种数组和链表结构， 一个Segment里包含一个HashEntry数组，每个HashEntry是一个链表结构的元素， 每个Segment守护者一个HashEntry数组里的元素,当对HashEntry数组的数据进行修改时，必须首先获得它对应的Segment锁.



putIfAbsent:如果已存在key不会添加；不存在key会添加数据。

## AtomicReference

描述: 对"对象"进行原子操作;

compareAndSet(K,V) 返回 Boolean;如果实际值与期望值K相同，则更新为V；

get()获取值.

## Idea 快捷键

选中代码整体右移 tab；选中代码整体左移: shift+tab

## ReentrantReadWriteLock

描述：锁，多线程使用；对同一个对象加锁。

示例:

ReentrantReadWriteLock readWriteLock = new ReentrantReadWriteLock();

ReentrantReadWriteLock.ReadLock readLock = readWriteLock.readLock();

ReentrantReadWriteLock.WriteLock writeLock = readWriteLock.writeLock();

ReentrantReadWriteLock是Lock的另一种实现方式，我们已经知道了ReentrantLock是一个排他锁，同一时间只允许一个线程访问，而ReentrantReadWriteLock允许多个读线程同时访问，但不允许写线程和读线程、写线程和写线程同时访问。相对于排他锁，提高了并发性。在实际应用中，大部分情况下对共享数据（如缓存）的访问都是读操作远多于写操作，这时ReentrantReadWriteLock能够提供比排他锁更好的并发性和吞吐量.

支持公平和非公平的获取锁的方式;