public class IdWorker{

private long workerId; //工作id

private long datacenterId; //数据id

private long sequence; //12位的序列号

public IdWorker(long workerId, long datacenterId, long sequence){

this.workerId = workerId;

this.datacenterId = datacenterId;

this.sequence = sequence;

}

private long twepoch = 1288834974657L;//初始时间戳 private long lastTimestamp = -1L; //上次时间戳，初始值为负数

private long workerIdBits = 5L; private long datacenterIdBits = 5L;//长度为5位

private long maxWorkerId = -1L ^ (-1L << workerIdBits); //最大值

private long maxDatacenterId = -1L ^ (-1L << datacenterIdBits);

private long sequenceBits = 12L; //序列号id长度

private long sequenceMask = -1L ^ (-1L << sequenceBits); //序列号最大值

private long workerIdShift = sequenceBits; //工作id需要左移的位数，12位

private long datacenterIdShift = sequenceBits + workerIdBits;//数据id需要左移位数 12+5=17位

private long timestampLeftShift = sequenceBits + workerIdBits + datacenterIdBits;

public synchronized long nextId() {//下一个ID生成算法

long timestamp = timeGen();

if (timestamp < lastTimestamp) {//获取当前时间戳如果小于上次时间戳，则表示时间戳获取出现异常

throw new RuntimeException(String.format("Clock moved backwards. Refusing to generate id for %d milliseconds", lastTimestamp - timestamp));

}

//获取当前时间戳如果等于上次时间戳（同一毫秒内），则在序列号加一；否则序列号赋值为0，从0开始。

if (lastTimestamp == timestamp) {

sequence = (sequence + 1) & sequenceMask;

if (sequence == 0) { timestamp = tilNextMillis(lastTimestamp); }

} else {

sequence = 0;

}

lastTimestamp = timestamp; //将上次时间戳值刷新

return ((timestamp - twepoch) << timestampLeftShift) | (datacenterId << datacenterIdShift) | (workerId << workerIdShift) |sequence;

}

private long tilNextMillis(long lastTimestamp) { //获取时间戳，并与上次时间戳比较

long timestamp = timeGen();

while (timestamp <= lastTimestamp) {

timestamp = timeGen();

}

return timestamp;

}

private long timeGen(){return System.currentTimeMillis();//获取系统时间戳}

}