

反射、泛型、容器



反射

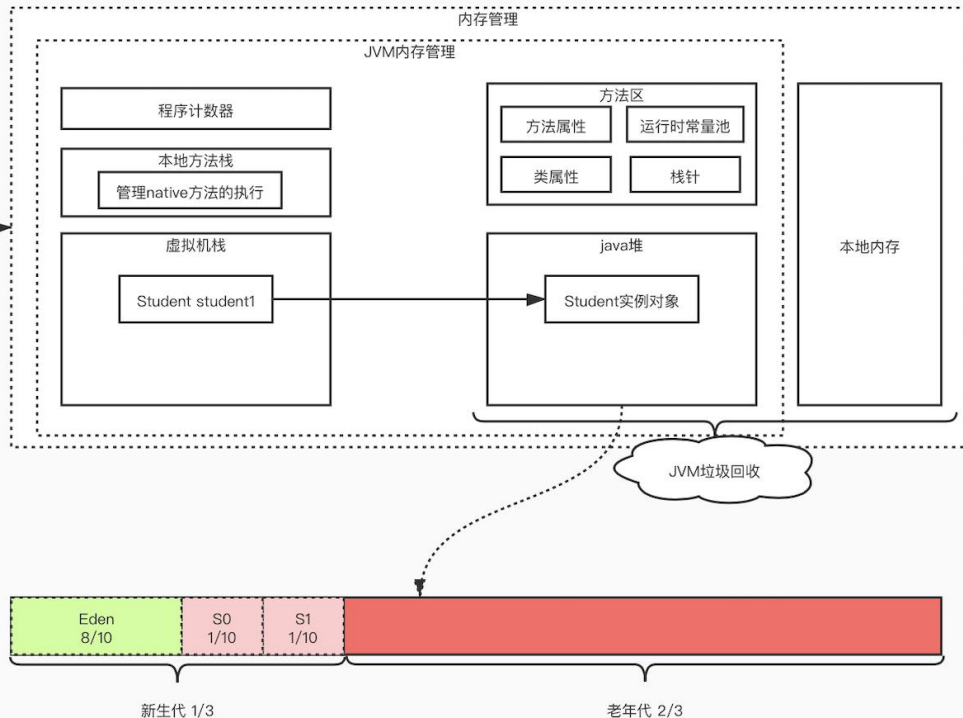
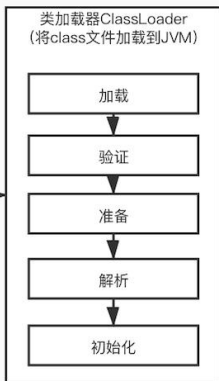
JVM类加载流程和内存结构

```
【Java源文件】 public class Student {  
    private String name;  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

```
    public String getName() {  
        return this.name;  
    }  
    public static void main(String[] args) {  
        Student student1 = new Student();  
        student1.setName("muse");  
    }  
}
```

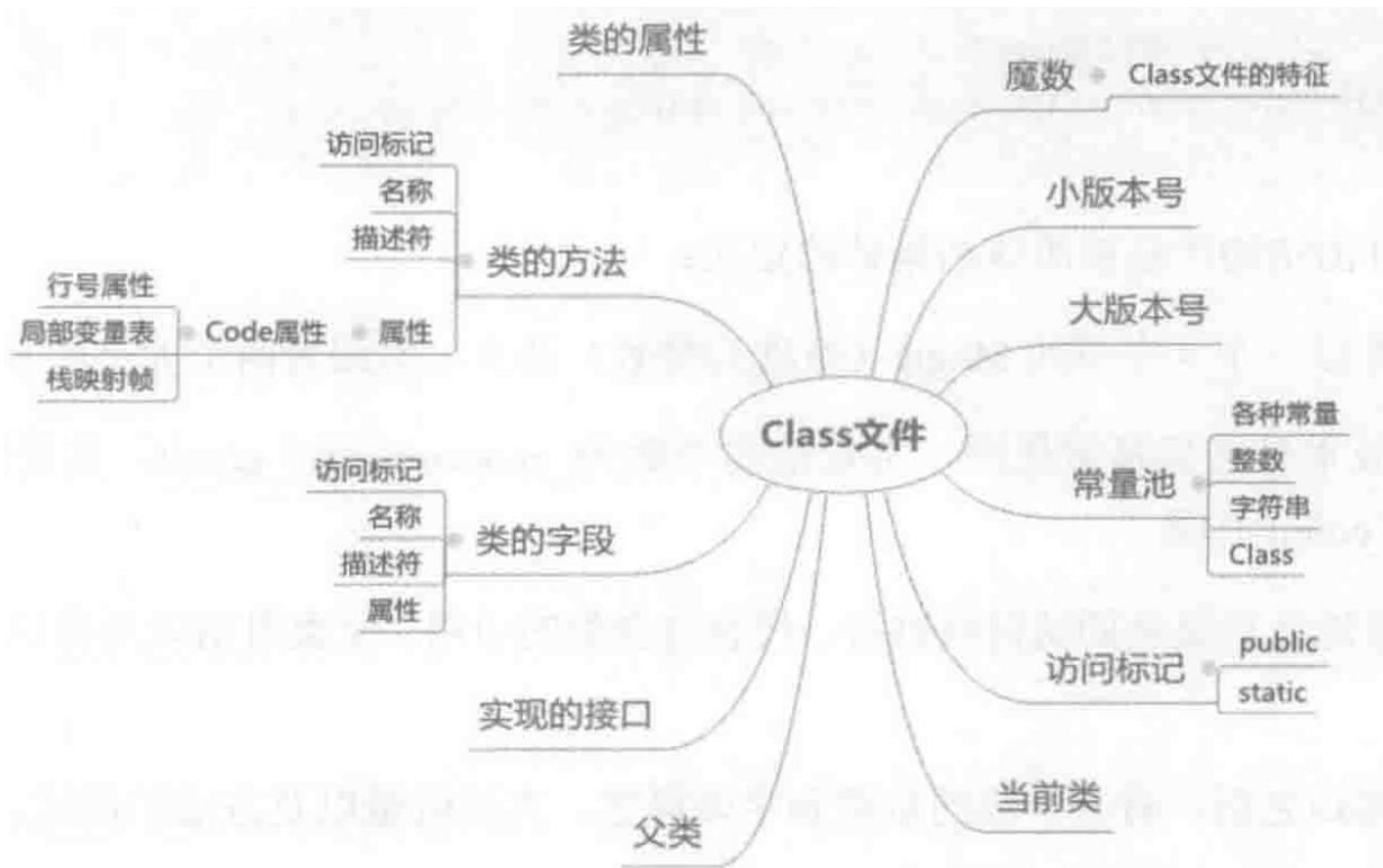
类编译器
(将java源码编译为class文件)

```
【Class文件】 cafe babe 0000 0031 0022 0a00  
0700 1c090003 001d 0700 1e0a 0003 001c  
0800 1f0a0003 0020 0700 2101 0004 6e61  
6d65 0100124c 6a61 7661 2f6c 616e 672f 5374  
72696e67 3b01 0006 3c69 6e69 743e 0100  
03282956 0100 0443 6f64 6501 000f 4c69  
6e654e75 6d62 6572 5461 626c 6501 0012  
4c6f... ...
```

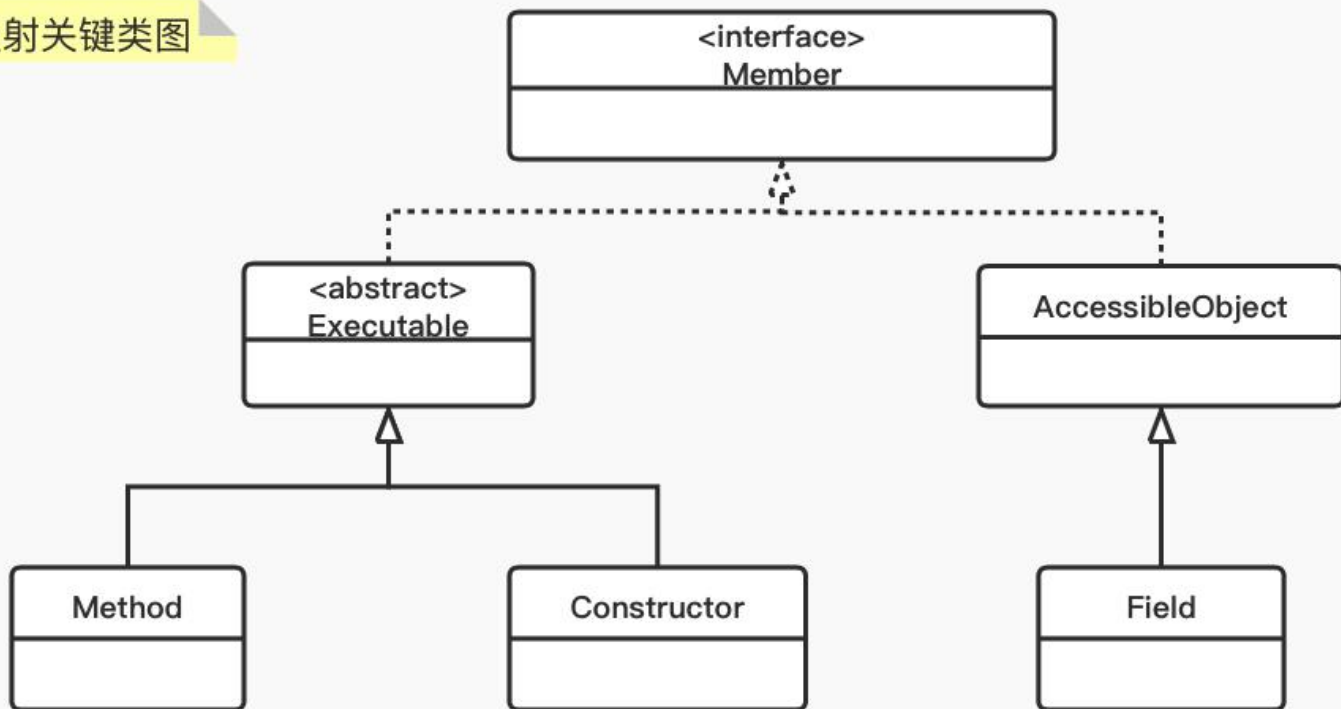


堆内存 = Eden内存 + S0/S1内存 + 老年代内存

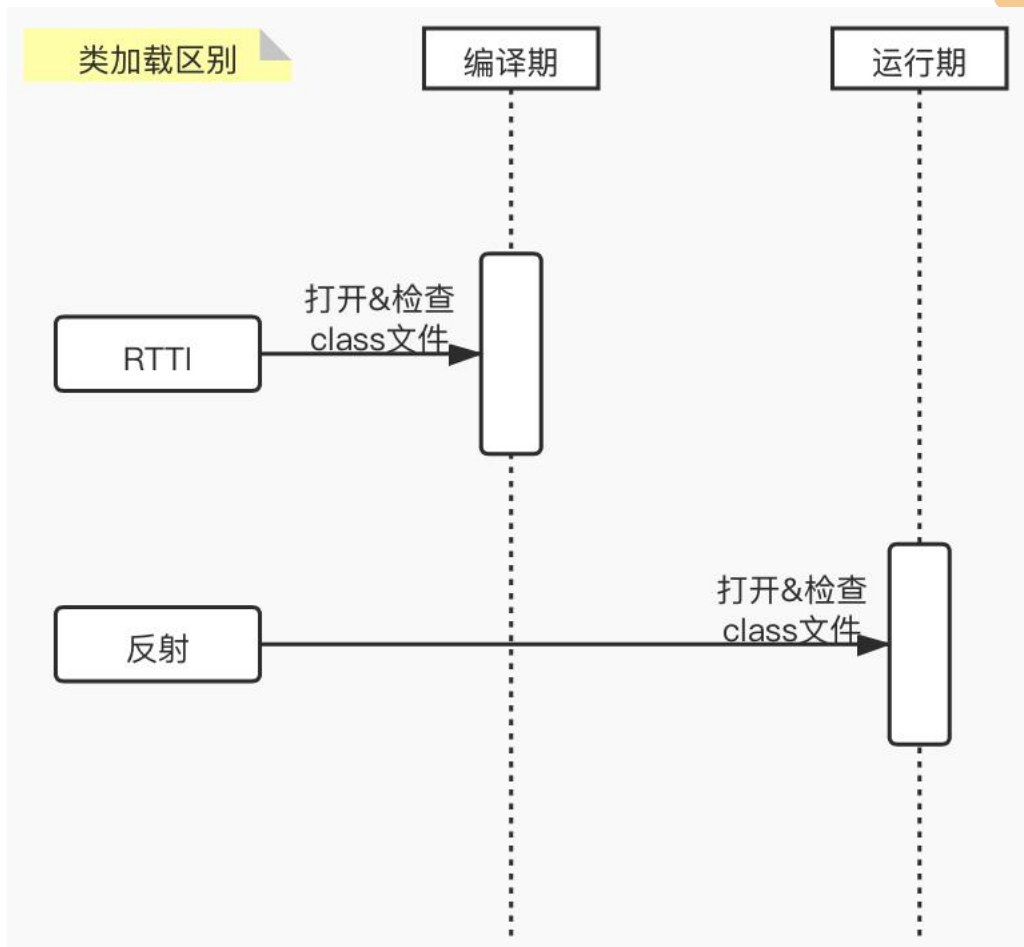
Class文件包含的内容



反射关键类图



类加载区别



生成对象的步骤

生成对象步骤

```
Person person = new Person();
```

编译期加
载Person.class文
件

查找构造函数
Person()

通过构造函
数Person()创建对
象

```
Class personClazz = Class.forName("com.muse.Person");
```

运行期加
载Person.class文
件

```
Constructor constructor = personClazz.getConstructor();
```

创建构造函数
Person()

```
Person person = (Person) constructor.newInstance();
```

通过构造函
数Person()创建对
象

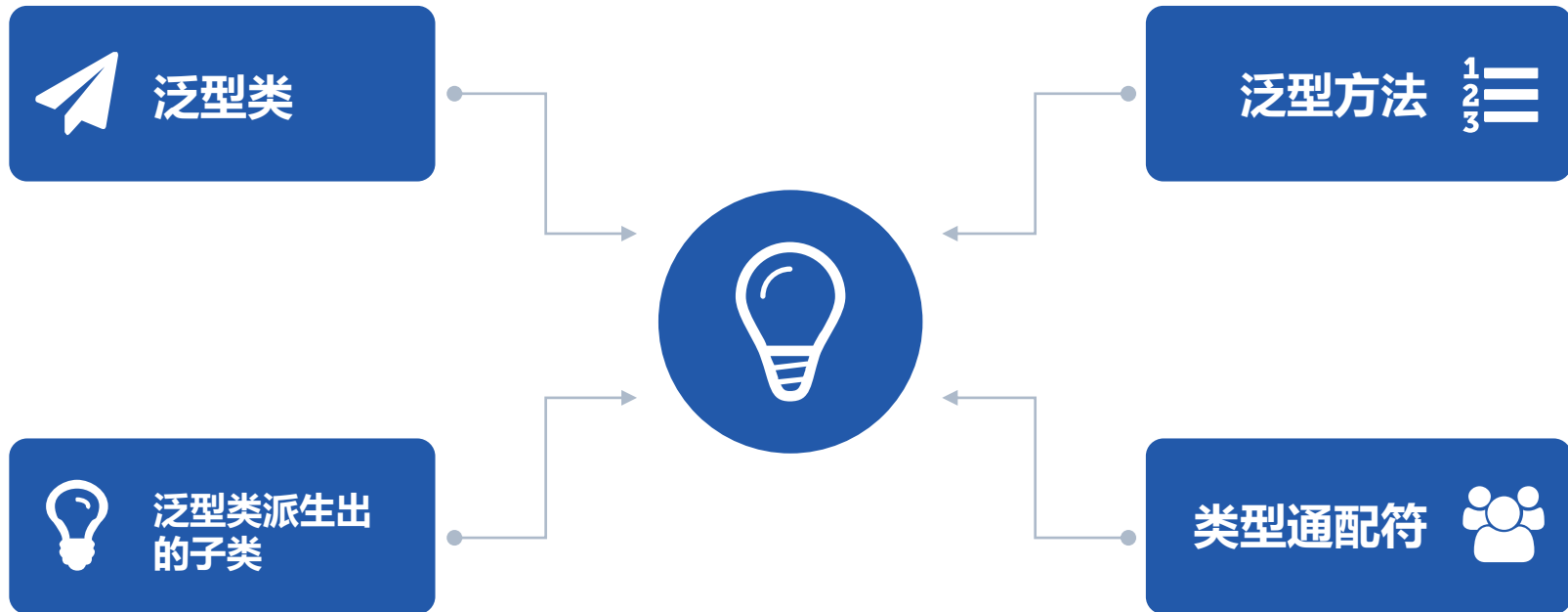
我们采用反射机制来实现一个工具BeanUtils，
可以将一个对象属性相同的值赋值给另一个对象。

泛型

泛型本质是指**类型参数化**。

允许在定义**类**、**接口**、**方法**时使用类型形参，当使用时指定具体类型。

所有使用该泛型参数的地方都被统一化，保证类型一致。如果未指定具体类型，默认是Object类型。集合体系中的所有类都增加了泛型，**泛型也主要用在集合**。



泛型的上限：

- 格式： 类型名称 <? extends 类 > 对象名称
- 意义： 只能接收该类型及其子类

泛型的下限：

- 格式： 类型名称 <? super 类 > 对象名称
- 意义： 只能接收该类型及其父类型

泛型是提供给javac编译器使用的，它用于限定集合的输入类型，让编译器在源代码级别上，即挡住向集合中插入非法数据。但编译器编译完带有泛形的java程序后，**生成的class文件中将不再带有泛型信息**，以此使程序运行效率不受到影响，这个过程称之为“擦除”。

由于类型被擦除了，**为了维持多态性**，所以编译器就自动生成了桥接方法。



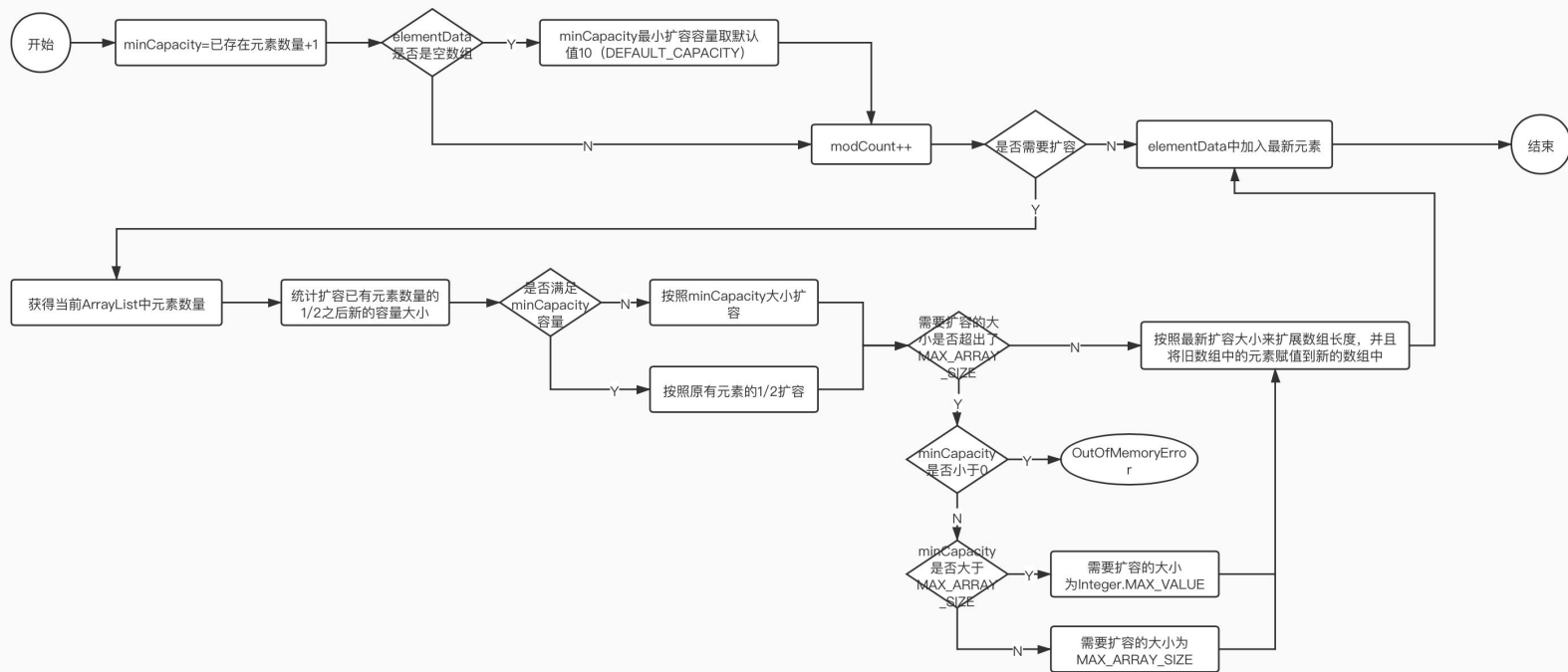
集合容器

ArrayList源码解析

```
ArrayList arrayList = new ArrayList();
```

```
elementData={}
```

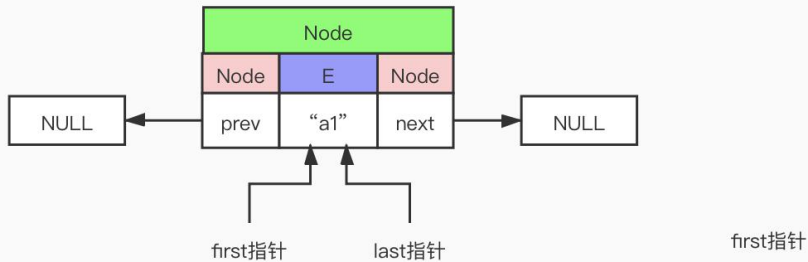
```
arrayList.add("a1");
```



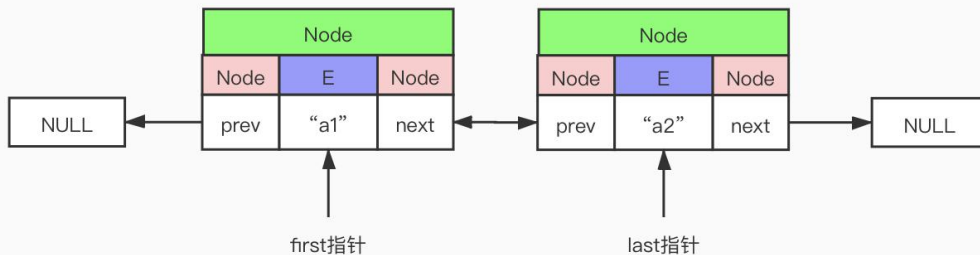
LinkedList源码解析

```
LinkedList linkedList = new LinkedList();
```

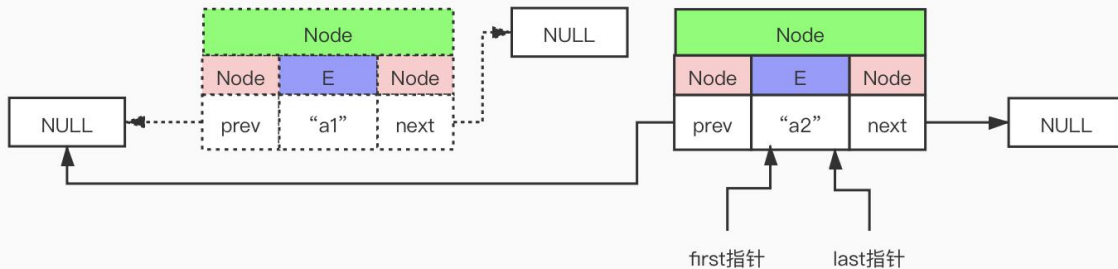
```
linkedList.add("a1");
```



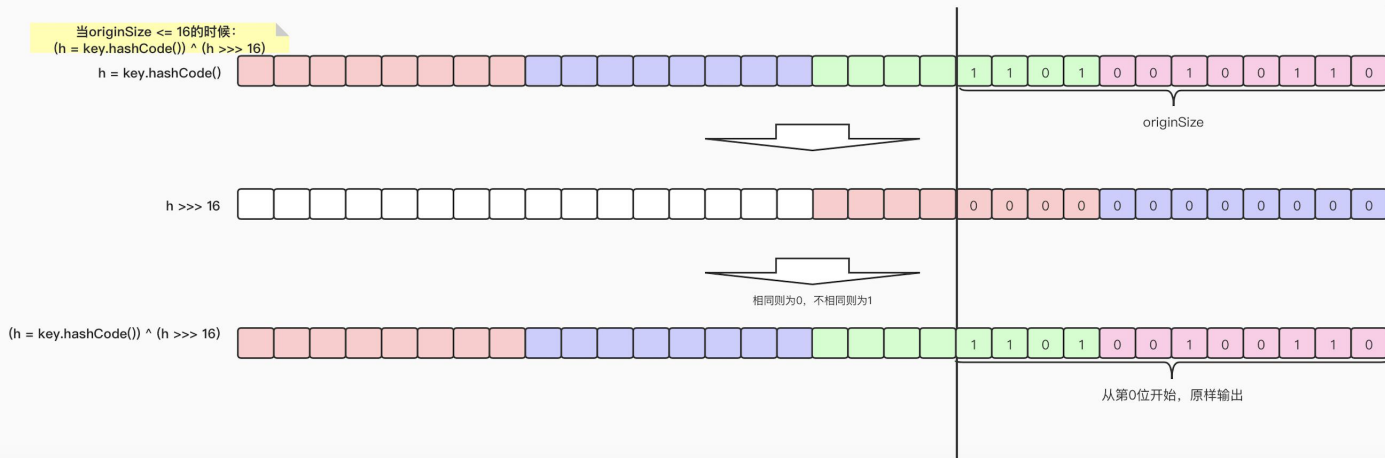
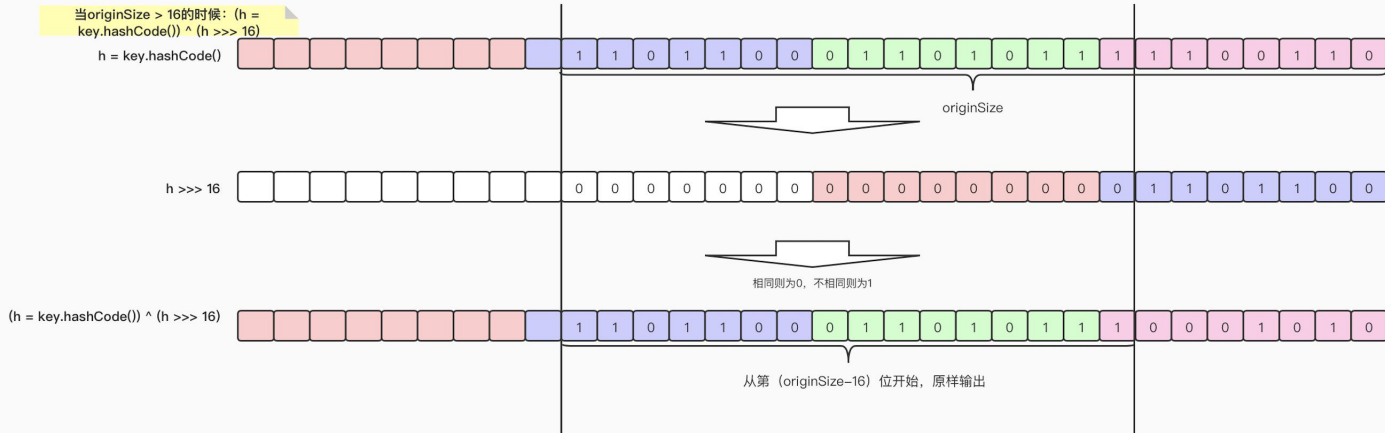
```
linkedList.add("a2");
```



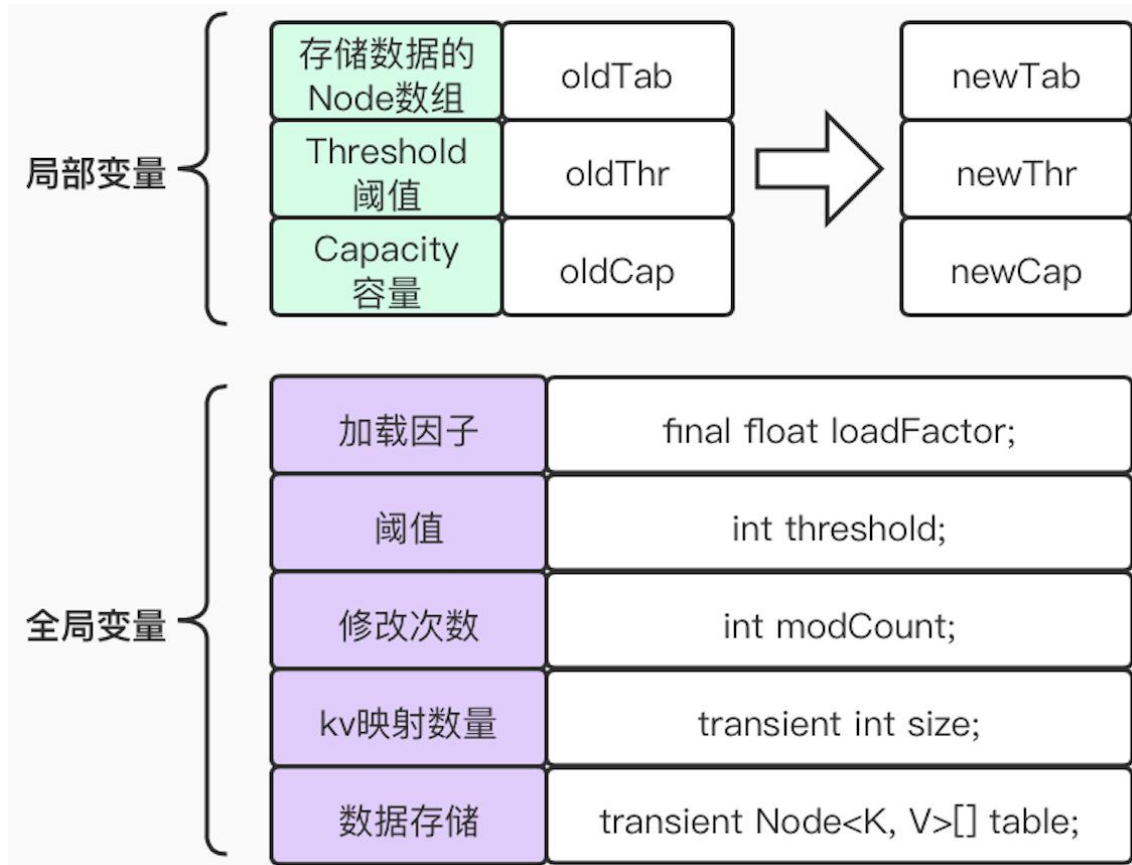
```
linkedList.remove(1);
```



HashMap源码解析——HashMap.hash(Object key)

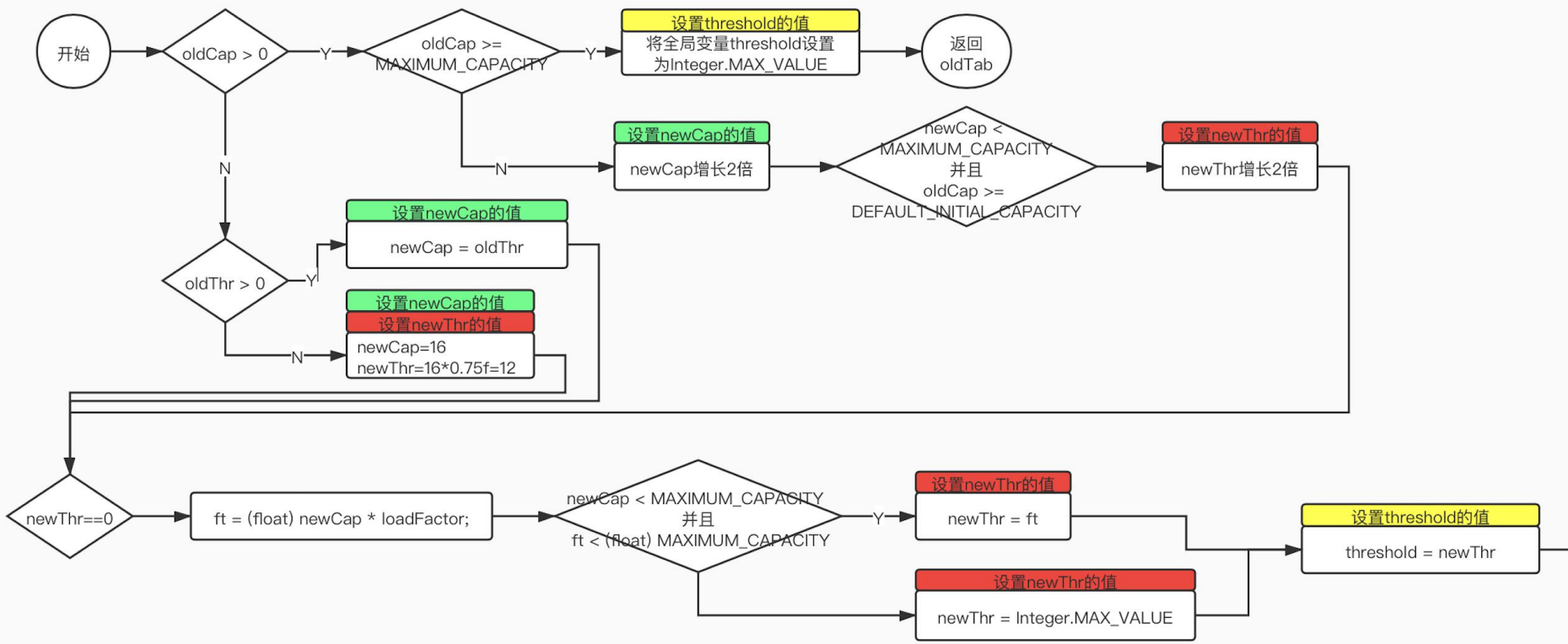


HashMap源码解析——关键变量



HashMap源码解析——resize

resize()解析——第一阶段：参数newCap, newThr, threshold的解析计算



HashMap源码解析——resize