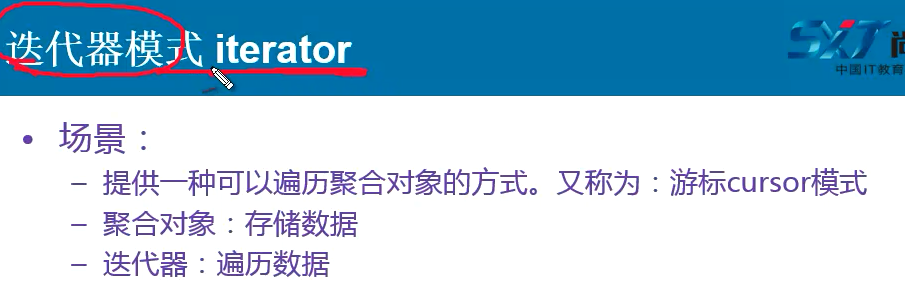
迭代器模式 --- 也叫**游标模式**

这个模式一直在用 仅仅是JDK已经提供好了

实际工作中 自己实现迭代器 不多见

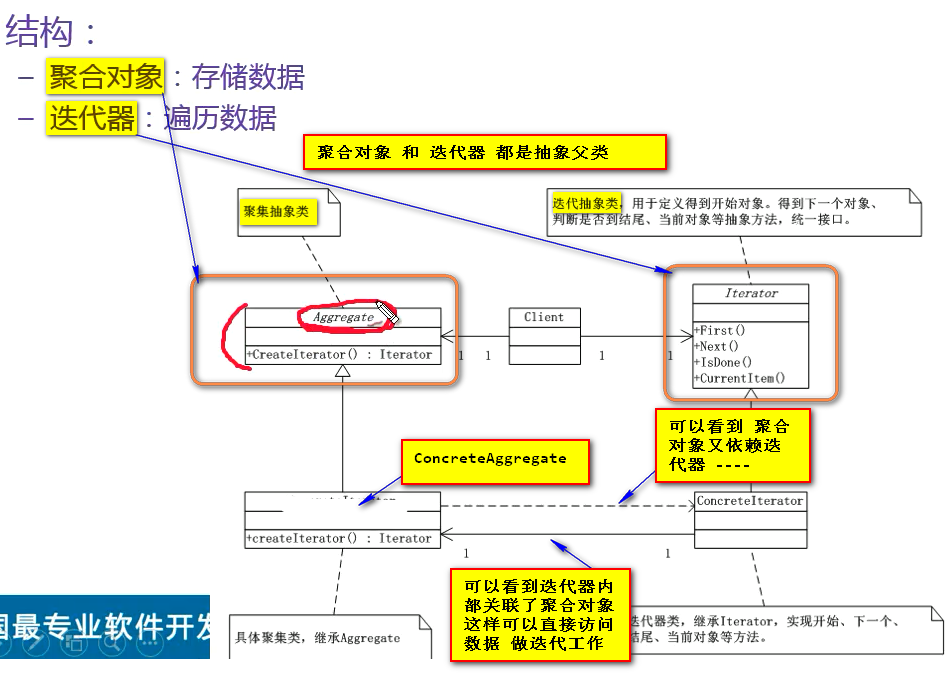


提供一种可遍历局和对象的方式

理解迭代器分成两个点 ---- 整个模式有两个部分 聚合对象 和迭代器

聚合对象 ---- 存储数据

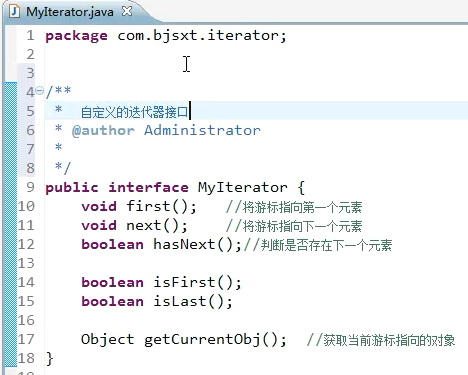
迭代器 ---- 遍历数据



Aggregate ---- 聚集

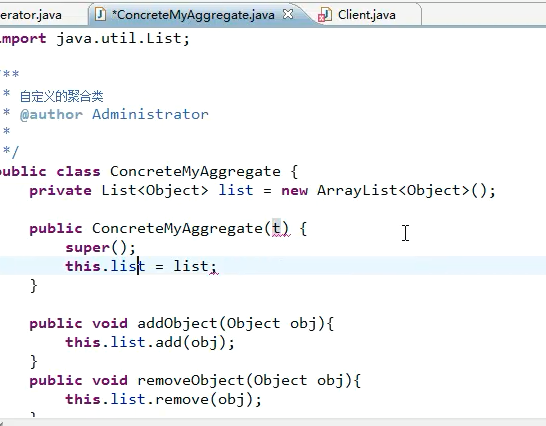
定义迭代器 通常会把迭代器定义到聚合类的内部类 ---- 内部类的好处就是可以直接操作聚合数据集的对象

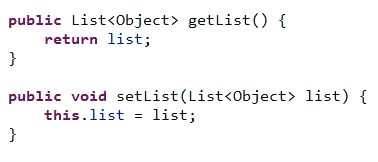
通常情况 迭代器就是为某一个聚集类来服务的 别的地方未必适用 所以 选择用内部类实现 JDK中的迭代器 通常都是这么做的 但是 还是要有一个接口



写一个聚合类ConcreteMyAggregate

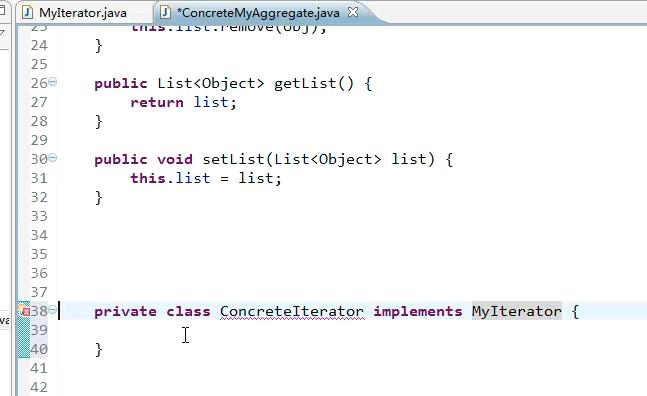
有一个容器 存储我们的对象 然后操作这个容器中的对象

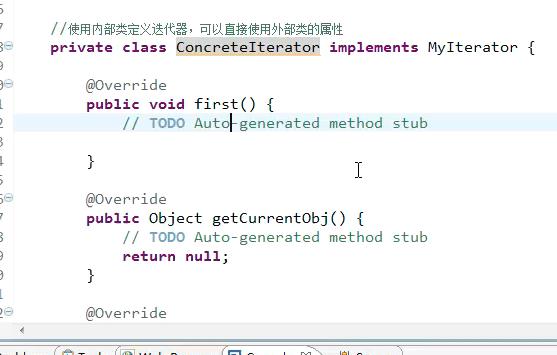




还有就是获取迭代器

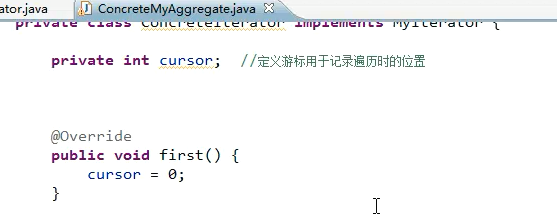
迭代器内部类实现 我们就使用内部类

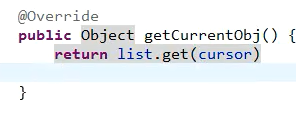


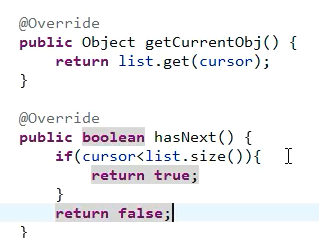


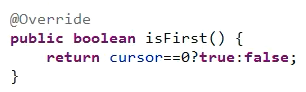
这个迭代器需要一个游标 通过这个指针 指向其他的内容

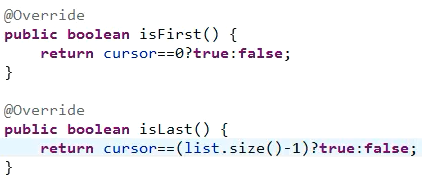
定义游标 用于记录遍历时的位置

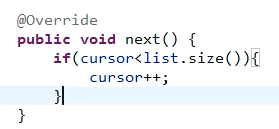


 list直接被内部类调用







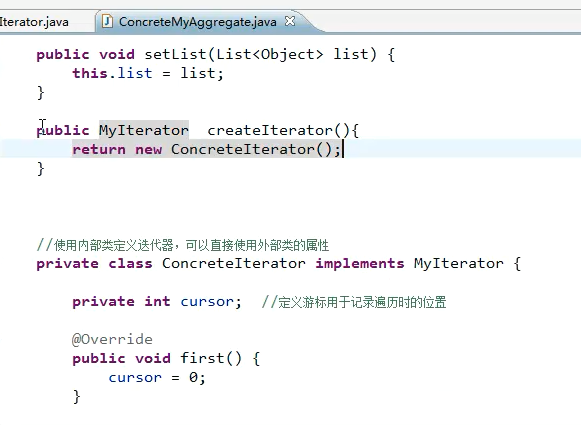


在外部类中提供获取迭代器的方法

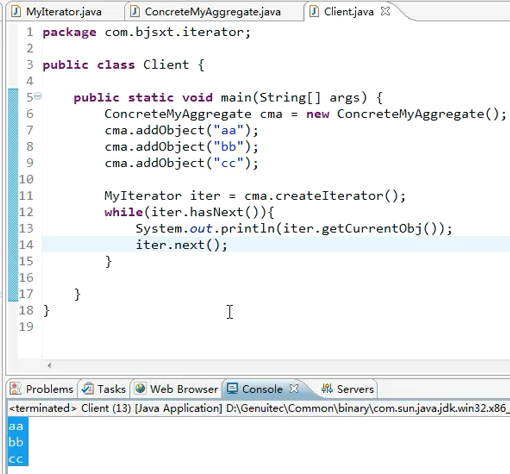
这样外部的聚集类就是依赖了内部的迭代器类

内部的迭代器类实际上也是依赖了外部的集合对象列表

所以类图中是双向箭头【你中有我 我中有你】



createIterator就相当于工厂方法

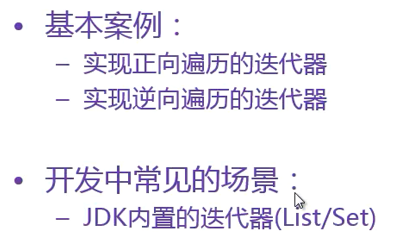


起到了

Jdk中的next()方法是这里面的getCurrentObj和next的合并 这里面的next()就是让游标向下走一个

实际上还是分工 ---- 遍历集合 和 存储集合本身的工作分开了

如果做一个逆向迭代器 ------ 开始游标就不是0了



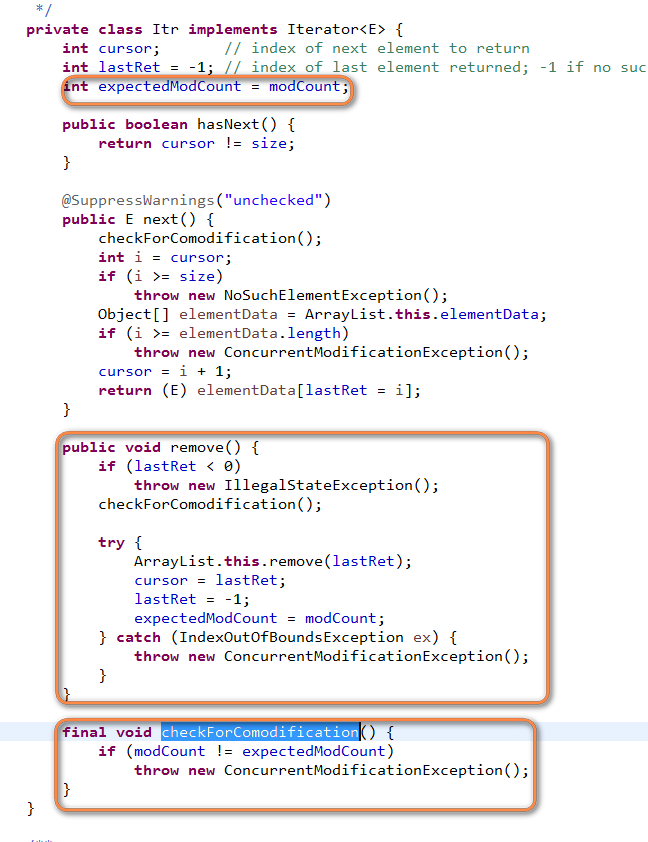
List和Set实现了Collection Collection实现了Iterator

ArrayList Set都采用内部类实现

这样了解了迭代器的底层构造

迭代器模式经常用 自己设计不多见

【记得之前是不能并发删除list中的数据 但是 好像迭代模式可以 跟踪了一下ArrayList内部的代码



】