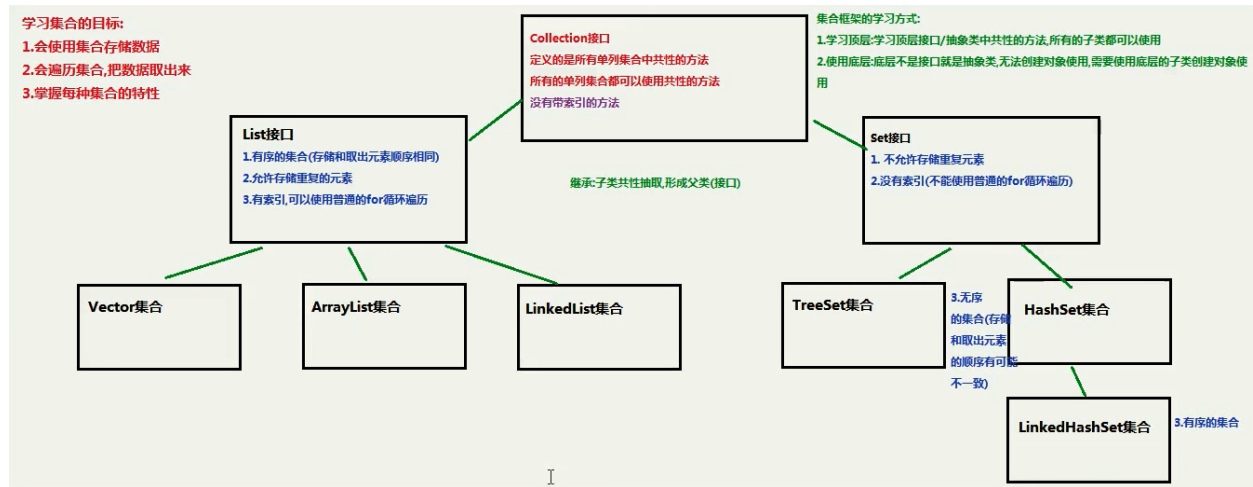


Collection概述



迭代器

1.Iterator<E> iterator(): 获取集合对应的迭代器,用于遍历集合中的元素。关于元素返回的顺序没有任何保证(除非collection是某一个能提供保证顺序的类实例)

2.迭代: 即Collection集合元素的通用获取方式,在取出元素之前先要判断集合中有没有元素,如果有,就把这个元素取出来,继续在判断,如果还有就再取出来,一直把集合中的所有元素全部取出来,这种去厨房是专业术语就叫做迭代。

3.Iterator接口的常用方式如下:

`public E next():` 返回迭代的下一个元素

`public boolean hasNext():` 如果仍有元素可以迭代,则返回true

4.使用步骤:

1.使用集合中的方法iterator()获取迭代器的实现类对象,使用Iterator接口接收(多态)。

2.使用Iterator接口中的方法hasNext判断还有没有下一个元素

3.使用Iterator接口中的方法next取出集合中的下一个元素

5.代码示例

```
public class CollectionIterator {
    public static void main(String[] args) {
        Collection<String> coll = new ArrayList<String>();
        coll.add("姚明");
        coll.add("科比");
        coll.add("麦迪");
        coll.add("詹姆斯");
        coll.add("艾弗森");
        //使用多态
        Iterator<String> it = coll.iterator();

        while(it.hasNext()){
            String n = it.next();
            System.out.println(n);
        }
    }
}
```

```
}
```

运行结果：

```
"E:\Program Files\Java\jdk1.8.0_171\bin\java" ...
姚明
科比
麦迪
詹姆斯
艾弗森

Process finished with exit code 0
```

6.迭代器实现原理：

```
//创建一个集合对象
Collection<String> coll = new ArrayList<>();
//往集合中添加元素
coll.add("姚明");
coll.add("科比");
coll.add("麦迪");
coll.add("詹姆斯");
coll.add("艾弗森");

/*
1.使用集合中的方法iterator()获取迭代器的实现类对象,使用Iterator接口接收(多态)
注意:
    Iterator<E>接口也是有泛型的,迭代器的泛型跟着集合走,集合是什么泛型,迭代器就是什么泛型
*/
//多态 接口      实现类对象
Iterator<String> it = coll.iterator(); 获取迭代器的实现类对象,并且会把指针(索引)指向集合的-1索引

/*
发现使用迭代器取出集合中元素的代码,是一个重复的过程
所以我们可以使用循环优化
不知道集合中有多少元素,使用while循环
循环结束的条件,hasNext方法返回false
*/
while(it.hasNext()){ 判断集合中还有没有下一个元素
    String e = it.next();
    System.out.println(e); 做了两件事:
    1.取出下一个元素      "姚明"  "科比"  "麦迪"  "詹姆斯"  "艾弗森"
    2.把指针向后移动一位
}
```

new ArrayList<>();

"姚明"	"科比"	"麦迪"	"詹姆斯"	"艾弗森"
------	------	------	-------	-------

-1 0 1 2 3 4

↑ next ↑ next ↑ next ↑ next ↑ next ↑ next

✦

jdk1.9对集合天真的优化of()方法

jdk1.9新特性：

List接口，Set接口，Map接口里面增加了一个静态的方法of,可以给集合一次性添加多个元素 `static <E> list<E> of(E... elements)`

使用前提：

- 1.of方法只适用于List接口，Set接口，Map接口，不适用于接口的实现类
- 2.of方法的返回值是一个不能改变的集合，集合不能再使用add,put方法添加元素，会抛出异常
- 3.Set接口和Map接口 在调用of方法的时候，不能有重复的元素，否则会抛出异常

```
public class Demo {
    public static void main(String[] args) {
        List<String> list = List.of("a", "b", "v", "a", "v"); //允许有重复值

        System.out.println(list); // [a, b, v, a, v]

        // Set<String> set = Set.of("a", "b", "v", "a"); //不允许重复, 不然直接报错 duplicate
        // element: a
        Set<String> set = Set.of("a", "b", "v");

        System.out.println(set); // [a, b, v]
    }
}
```

```
Map<String,String> map = Map.of("aa","vv","bb","cc");  
System.out.println(map); //{aa=vv, bb=cc}  
}  
  
}
```