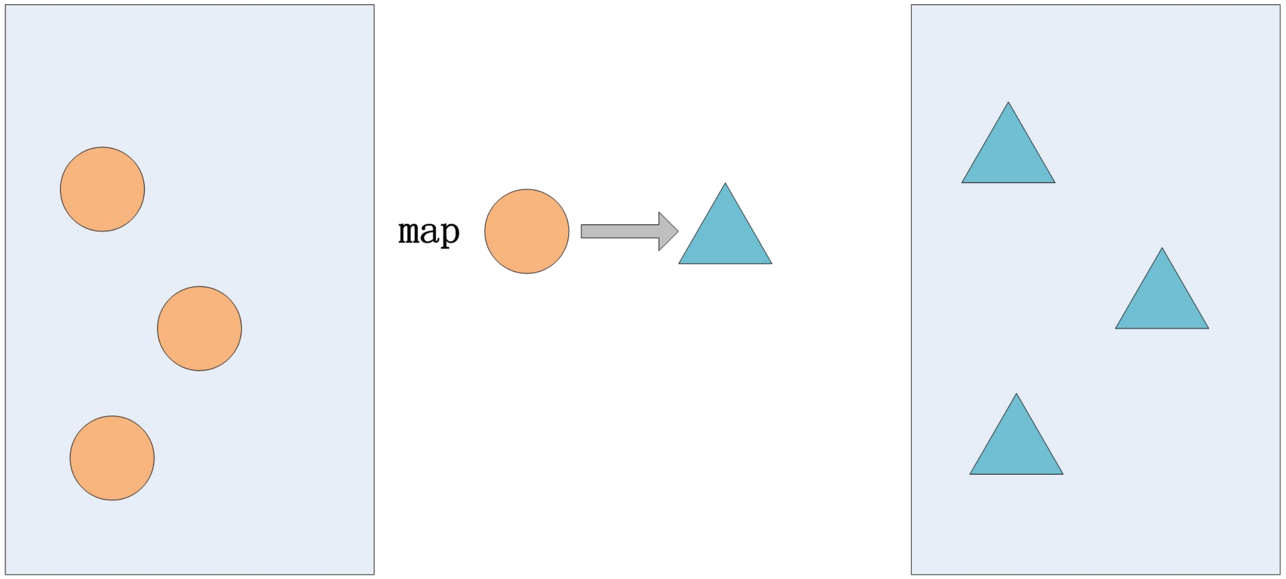
Scala集合库详解(部分)

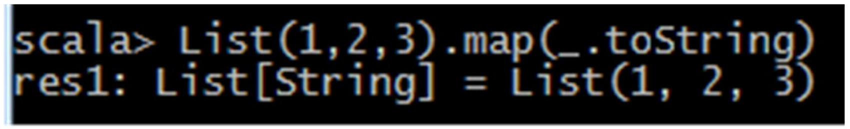
Thoughtworks 杨云
@诺铁

四种基本运算

- map
- filter
- fold/reduce
- flatten

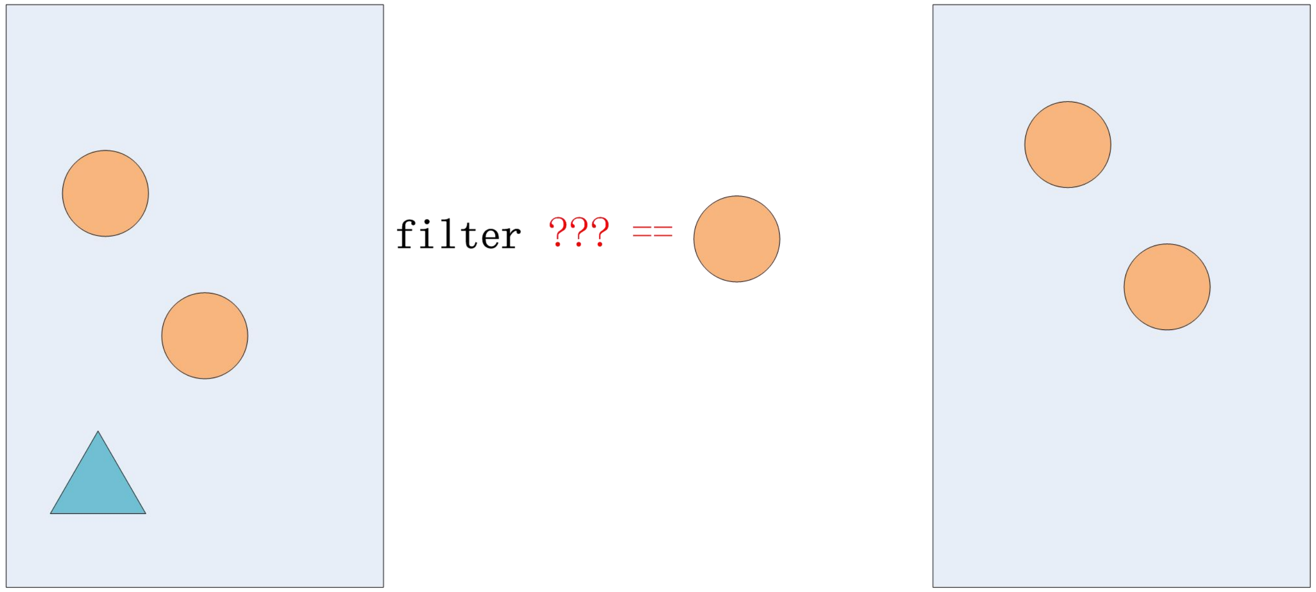
map运算

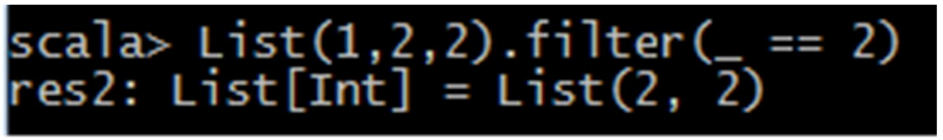




```
scala> List(1,2,3).map(_.toString)
res1: List[String] = List(1, 2, 3)
```

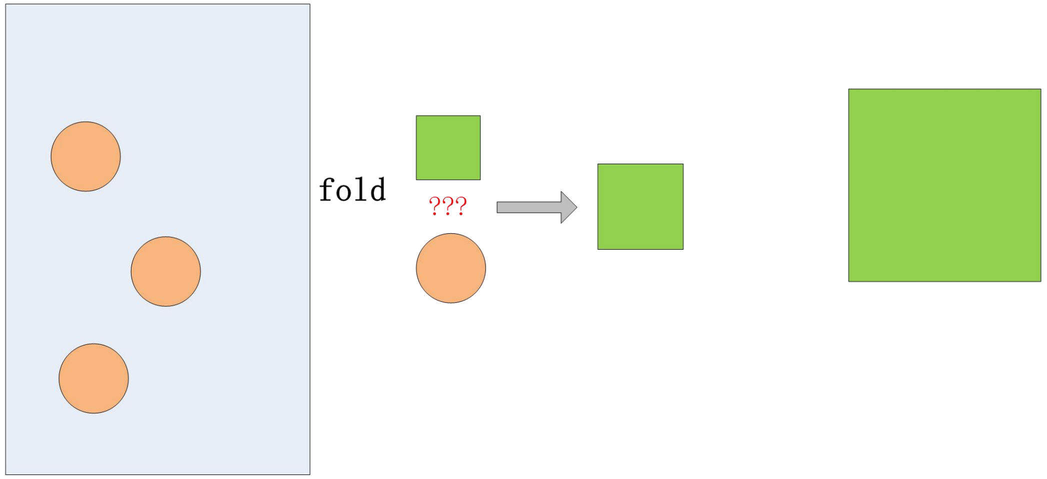
filter运算

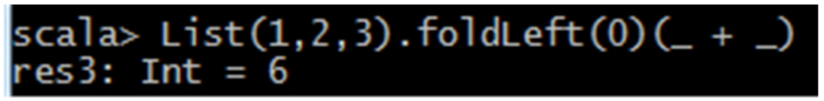




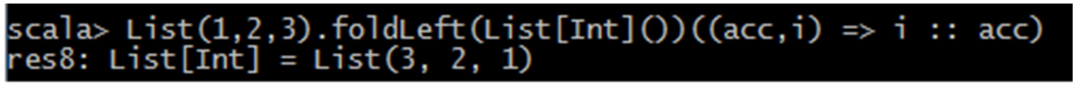
```
scala> List(1,2,2).filter(_ == 2)
res2: List[Int] = List(2, 2)
```

fold运算

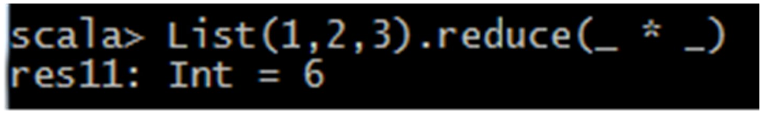




```
scala> List(1,2,3).foldLeft(0)(_ + _)
res3: Int = 6
```

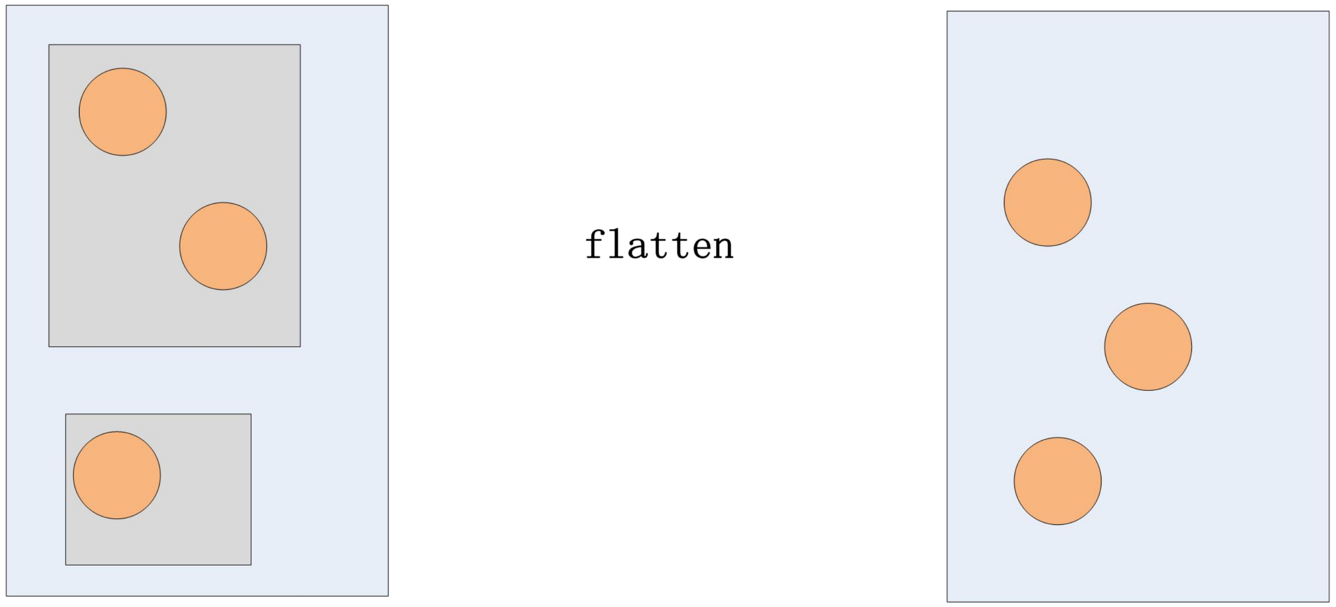


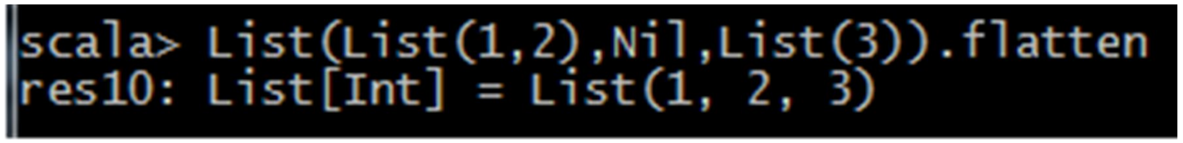
```
scala> List(1,2,3).foldLeft(List[Int]())((acc,i) => i :: acc)
res8: List[Int] = List(3, 2, 1)
```



```
scala> List(1,2,3).reduce(_ * _)
res11: Int = 6
```

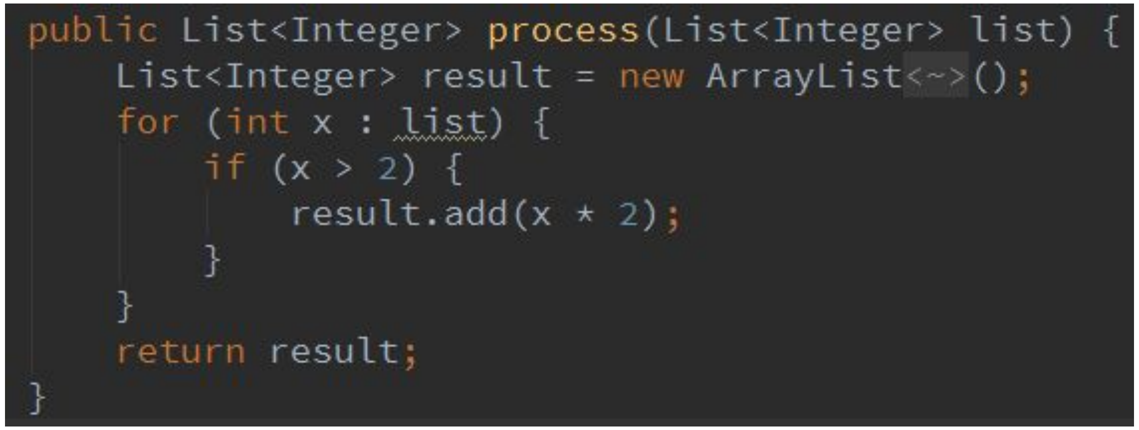
flatten运算



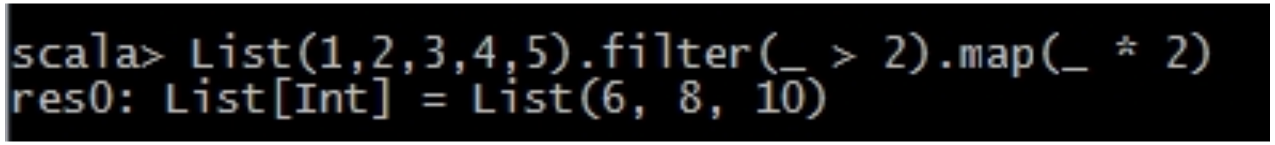


```
scala> List(List(1,2),Nil,List(3)).flatten
res10: List[Int] = List(1, 2, 3)
```

一生二、二生三、三生万物

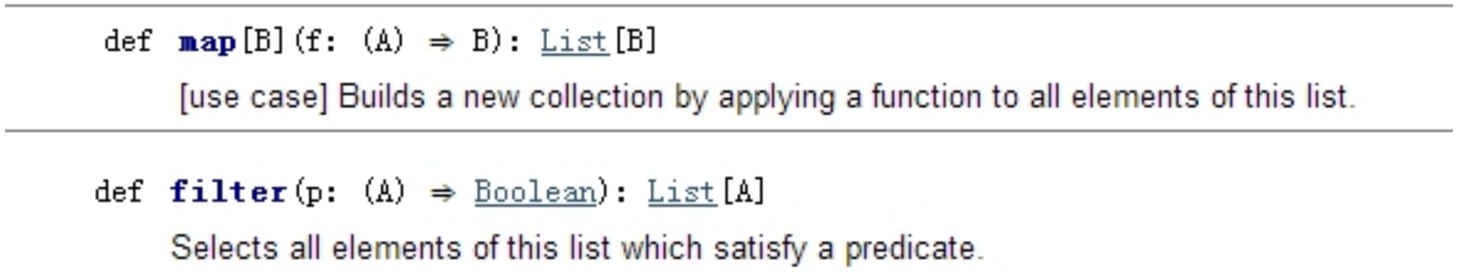


```java
public List<Integer> process(List<Integer> list) {
    List<Integer> result = new ArrayList<~>();
    for (int x : list) {
        if (x > 2) {
            result.add(x * 2);
        }
    }
    return result;
}
```



```
scala> List(1,2,3,4,5).filter(_ > 2).map(_ * 2)
res0: List[Int] = List(6, 8, 10)
```

高阶函数的妙用

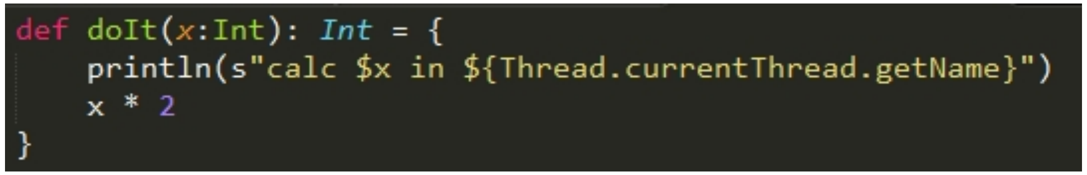


```
def map[B](f: (A) ⇒ B): List[B]
```

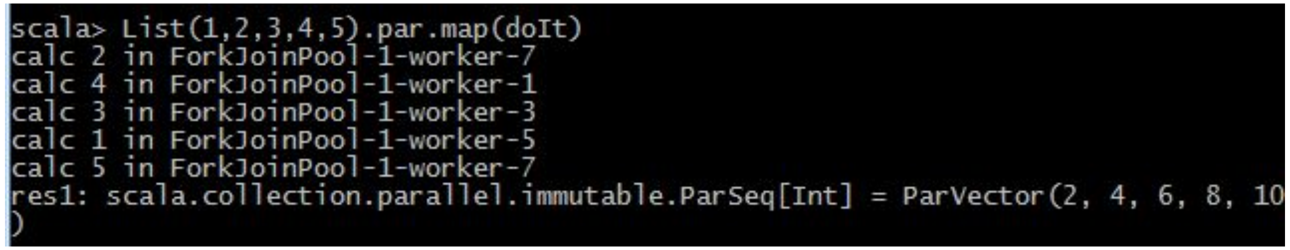
[use case] Builds a new collection by applying a function to all elements of this list.

```
def filter(p: (A) ⇒ Boolean): List[A]
```

Selects all elements of this list which satisfy a predicate.



```scala
def doIt(x:Int): Int = {
    println(s"calc $x in ${Thread.currentThread.getName}")
    x * 2
}
```



```
scala> List(1,2,3,4,5).par.map(doIt)
calc 2 in ForkJoinPool-1-worker-7
calc 4 in ForkJoinPool-1-worker-1
calc 3 in ForkJoinPool-1-worker-3
calc 1 in ForkJoinPool-1-worker-5
calc 5 in ForkJoinPool-1-worker-7
res1: scala.collection.parallel.immutable.ParSeq[Int] = ParVector(2, 4, 6, 8, 10
)
```

集合库的分类

- 可变集合 vs 不可变集合
- 即时计算集合 vs 延迟计算集合
- 顺序计算集合 vs 并行计算集合

继承层次

引用自scala in depth

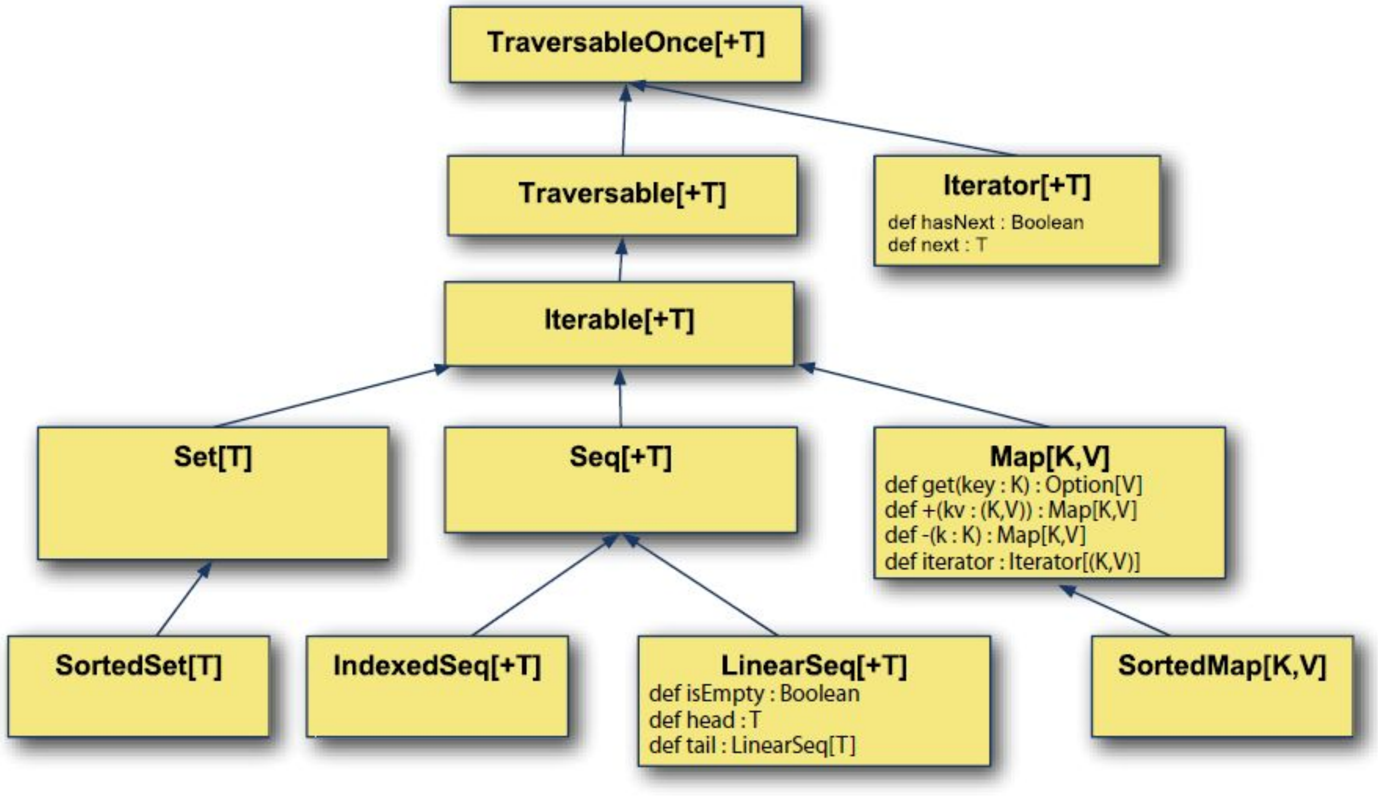
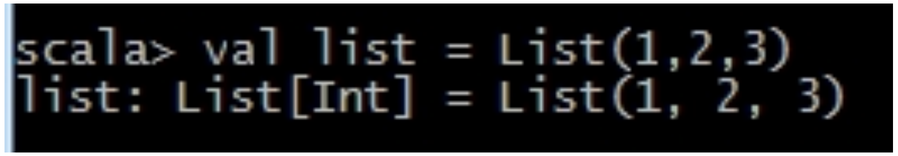
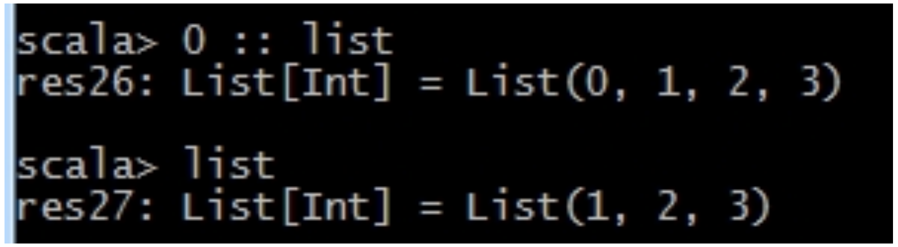


Figure 8.1 Generic collections hierarchy

不可变集合



```
scala> val list = List(1,2,3)
list: List[Int] = List(1, 2, 3)
```



```
scala> 0 :: list
res26: List[Int] = List(0, 1, 2, 3)

scala> list
res27: List[Int] = List(1, 2, 3)
```

不可变集合之Vector

- Vector是个由元素的下标组成的前缀树（trie）。

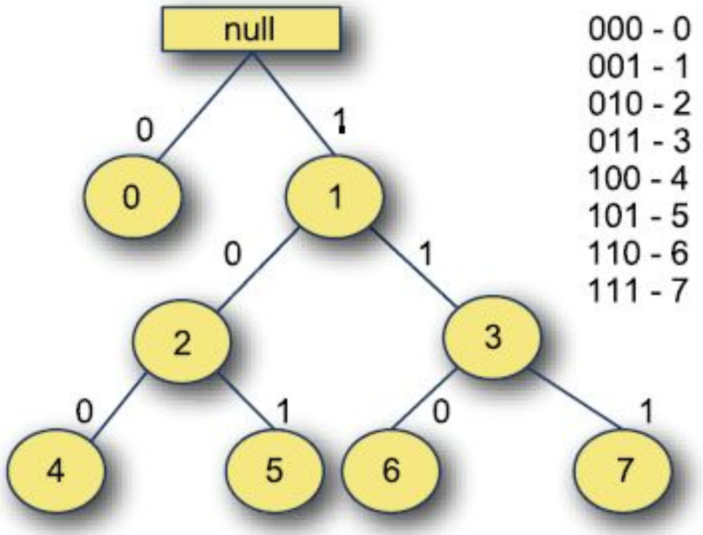


Figure 8.2 Example index trie with a branching factor of two

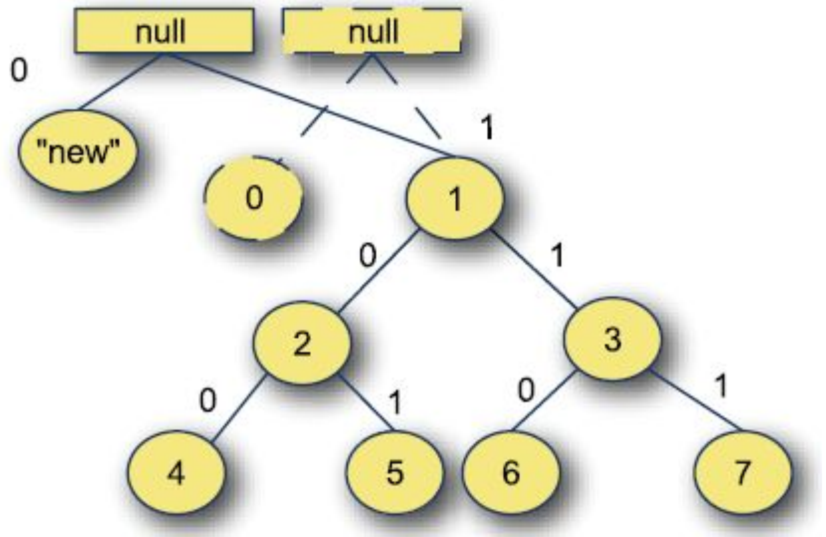


Figure 8.3 Update to trie with sharing

Vector续

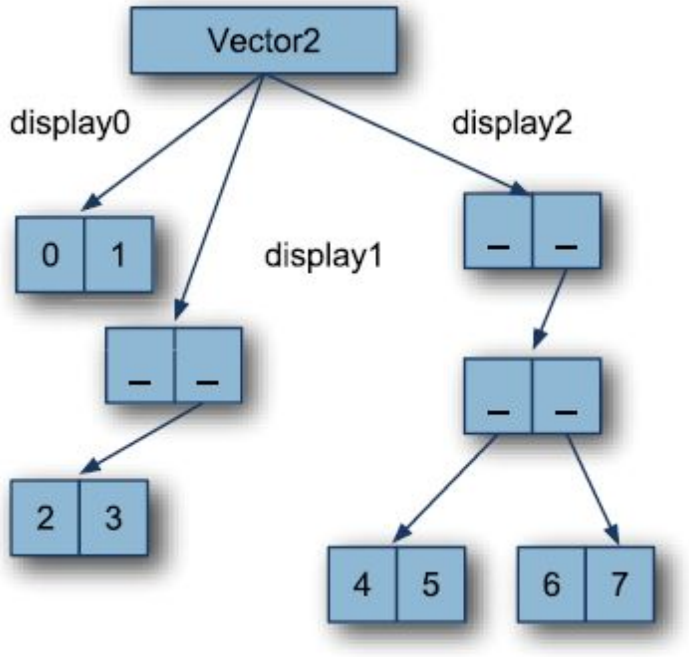


Figure 8.4 **Vector's array structure with branching factor of 2**

不可变集合之List

- List是个单链表

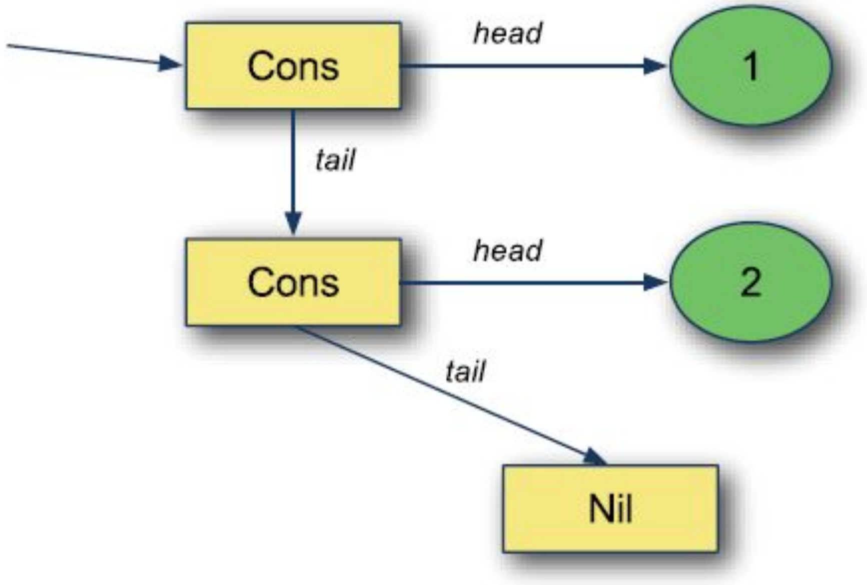
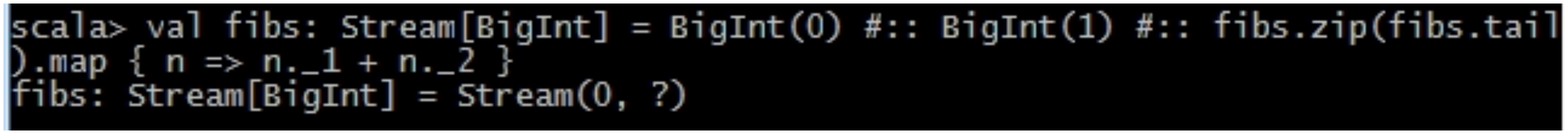


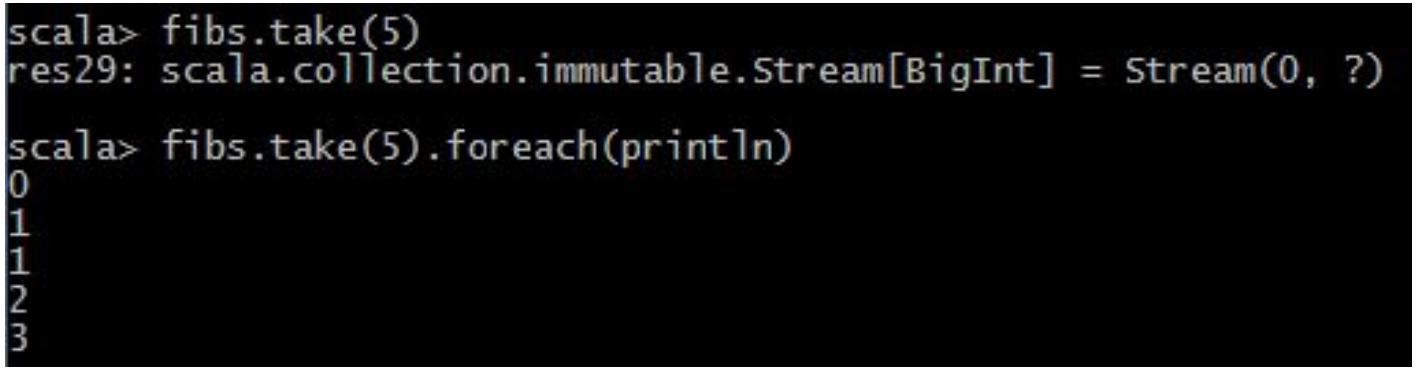
Figure 8.5 Internal structure of a list

不可变集合之Stream

- Stream是一种延迟持久（lazy persistent）的集合

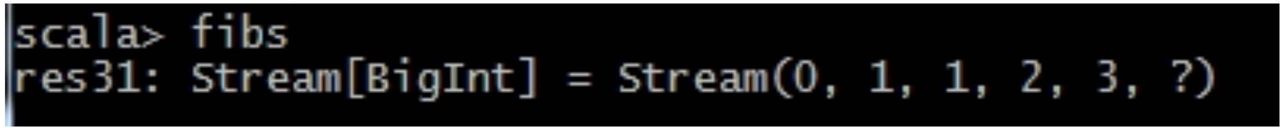


```
scala> val fibs: Stream[BigInt] = BigInt(0) #:: BigInt(1) #:: fibs.zip(fibs.tail
).map { n => n._1 + n._2 }
fibs: Stream[BigInt] = Stream(0, ?)
```



```
scala> fibs.take(5)
res29: scala.collection.immutable.Stream[BigInt] = Stream(0, ?)

scala> fibs.take(5).foreach(println)
0
1
1
2
3
```



```
scala> fibs
res31: Stream[BigInt] = Stream(0, 1, 1, 2, 3, ?)
```

可变集合

待续