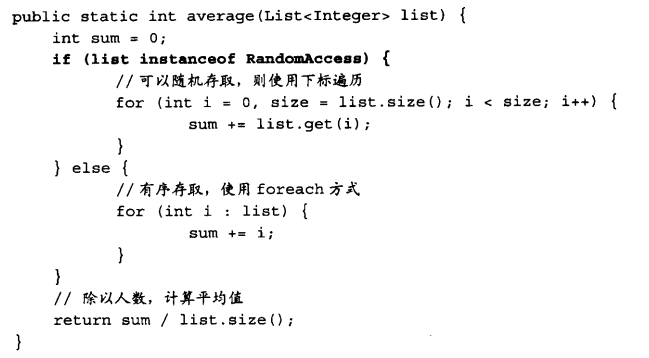
实现了RandomAccess接口（随机存取接口）表示数据元素之间没有关联，可以随机访问，比next的访问速度快些

ArrayList 用 get方式遍历快，LinkedList用迭代器（foreach）遍历快些



通过上面的分析我们也会发现如何构造一个优良的散列函数是一件非常重要的事情，我们构造散列函数的基本原则就是：尽可能的减少冲突，尽可能的将元素“散列”在存储空间中

　　下面是我从维基上找到的一些方法，之后如果有好的想法再做补充：

1. [直接定址法](http://zh.wikipedia.org/w/index.php?title=%E7%9B%B4%E6%8E%A5%E5%AE%9A%E5%9D%80%E6%B3%95&action=edit&redlink=1)：取关键字或关键字的某个线性函数值为散列地址。即hash(k)=k或hash(k)=a\cdot k + b，其中a\,b为常数（这种散列函数叫做自身函数）
2. [数字分析法](http://zh.wikipedia.org/w/index.php?title=%E6%95%B0%E5%AD%97%E5%88%86%E6%9E%90%E6%B3%95&action=edit&redlink=1)：假设关键字是以*x*为基的数，并且哈希表中可能出现的关键字都是事先知道的，则可取关键字的若干数位组成哈希地址。
3. [平方取中法](http://zh.wikipedia.org/wiki/%E5%B9%B3%E6%96%B9%E5%8F%96%E4%B8%AD%E6%B3%95)：取关键字平方后的中间几位为哈希地址。通常在选定哈希函数时不一定能知道关键字的全部情况，取其中的哪几位也不一定合适，而一个数平方后的中间几位数和数的每一位都相关，由此使随机分布的关键字得到的哈希地址也是随机的。取的位数由表长决定。
4. [折叠法](http://zh.wikipedia.org/w/index.php?title=%E6%8A%98%E5%8F%A0%E6%B3%95&action=edit&redlink=1)：将关键字分割成位数相同的几部分（最后一部分的位数可以不同），然后取这几部分的叠加和（舍去进位）作为哈希地址。
5. [随机数法](http://zh.wikipedia.org/w/index.php?title=%E9%9A%8F%E6%9C%BA%E6%95%B0%E6%B3%95&action=edit&redlink=1)
6. [除留余数法](http://zh.wikipedia.org/w/index.php?title=%E9%99%A4%E7%95%99%E4%BD%99%E6%95%B0%E6%B3%95&action=edit&redlink=1)：取关键字被某个不大于散列表表长m的数p除后所得的余数为散列地址。即hash(k)=k \,\bmod \,p, p\le m。不仅可以对关键字直接取模，也可在[折叠法](http://zh.wikipedia.org/w/index.php?title=%E6%8A%98%E5%8F%A0%E6%B3%95&action=edit&redlink=1)、[平方取中法](http://zh.wikipedia.org/wiki/%E5%B9%B3%E6%96%B9%E5%8F%96%E4%B8%AD%E6%B3%95)等运算之后取模。对p的选择很重要，一般取素数或m，若p选择不好，容易产生碰撞。